Estimating the Complexity of Software Services Using an Entropy Based Metric

George Feuerlicht[1,2,3(✉)] and David Hartman[1]

[1] Unicorn College, V Kapslovně 2767/2, 130 00 Prague 3, Czech Republic
george.feuerlicht@gmail.com,
david.hartman@unicorncollege.cz
[2] Prague University of Economics,
W. Churchill Square. 4, 130 67 Prague 3, Czech Republic
[3] University of Technology, Sydney,
Broadway, P.O. Box 123, Ultimo, NSW 2007, Australia

Abstract. Poor design of software services results in unnecessarily complex and inflexible SOA applications that are difficult to maintain and evolve. There is also some evidence that the quality of service-oriented applications degrades and as complexity increases with new service versions that include modifications to rectify problems and improve functionality. Design quality metrics play an important role in identifying software quality issues early in the software development lifecycle. The concept of software entropy has been used in literature to express decline in the quality, maintainability and understandability of software through its lifecycle. In this paper we propose a Service Entropy Metric (SEM) that estimates the complexity of service design based on the complexity of the XML message structures that form the basis for service interfaces. We illustrate the application of the SEM metric using the Open Travel Alliance specification and show that the complexity of the specification as measured by SEM increases over time as new versions of the specification are released.

Keywords: Service design metrics · Service complexity · Software entropy

1 Introduction

Increasing size and complexity of SOA (Service Oriented Architecture) applications presents a challenge to the developers of software services. Large-scale SOA projects typically involve thousands of software services (SOAP and REST [1] services), making the maintenance and evolution of applications implemented using these services highly challenging. It is widely accepted that excessive complexity leads to reduced quality of software and consequently to an increase in the maintenance effort. Predicting the quality of software during the design stage of the SDLC (Software Development Life Cycle), i.e. before the software is implemented, allows early rectification of design defects and can lead to a significant reduction of maintenance costs [2]. Early detection of service design issues requires reliable metrics that access software quality. Software quality can be measured by assessing structural properties of software (i.e. size, complexity, cohesion, coupling, etc.), and various design metrics for

A. Norta et al. (Eds.): ICSOC 2015 Workshops, LNCS 9586, pp. 15–23, 2016.
DOI: 10.1007/978-3-662-50539-7_2

object-oriented and component-based software development have been proposed and experimentally verified. [3–5]. Such metrics mostly rely on estimating cohesion and coupling, as maximizing cohesion and minimizing coupling reduces interdependencies between software components allowing individual components to be maintained independently without causing undesirable side effects [6, 7]. However, most of the proposed metrics were originally designed for fine-grained object-oriented software components and are of limited applicability for assessing the quality of coarse-grained, document-centric services that are used extensively in SOA applications. In order to develop a more suitable metric for estimating the quality of software services we focus on analysing the underlying XML message schemas that constitute the basis for service interfaces and directly impact on the quality of SOA applications. In our previous work we have proposed metrics that evaluate interdependencies among a set of XML message schemas by estimating the level of data coupling. Unlike most of the existing service metrics that were derived from metrics for object-oriented software, the DCI (Data Coupling Index) [8] and MDCI (Message Data Coupling Index) [9] analyse the underlying XML data structures to estimate the level coupling and to predict the impact of schema changes on existing SOA applications.

In this paper we focus on estimating the complexity of software services, as a measure of the effort required to develop and maintain software. The concept of *entropy* has been used to measure software complexity for over two decades [10], and more recently it was applied to estimating the complexity of XML schemas [11]. The proposed Software Entropy Metric (SEM) estimates the complexity of a software service based on the complexity of the XML message structures that form the service interface. In the following section (Sect. 2) we review related work that addresses the problem of measuring software quality and complexity. We then describe our approach to estimating the complexity of services and the XSD Analyzer tool that we have developed to compute the SEM complexity metric (Sect. 3). In Sect. 4 we present experimental results obtained by analysing the Open Travel Alliance schemas, and in Sect. 5 we presents our conclusions and outline directions for further work.

2 Related Work

Predicting maintainability of SOA applications using variants of existing metrics has been the subject of recent research interest [2, 12, 13]. Both coupling and cohesions have been used in traditional software design as indicators of software quality [14, 15, 16, 17, 18]. For example Chidamber et al. proposed the Lack of Cohesion in Methods (LCOM) metric for object-oriented software based on evaluating the similarity of methods. This original work has been used as the basis for developing metrics for software services [19, 13, 12]. While there are similarities between object-oriented software and software services, there are also significant differences and that make it difficult to apply similar metrics to both approaches. The underlying assumption for such metrics is that each service has a number of operations and that the interfaces of these operations are formed by input and output messages [12]. 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 difficult to apply. Another significant difference is that services are often based on pre-existing XML message schemas developed by various consortia and standards organizations. For example, the travel domain web services are based on the OTA (Open Travel Alliance) specification [20] that defines the structure of messages for travel applications. As the message schemas constitute the basis for service interfaces, it follows that the quality of service design is closely related to the quality of design of the underlying XML schemas. There is evidence that such schemas frequently contain overlaps and inconsistencies and suffer from excessive complexity [21, 22]. Standard XML specifications (e.g. OTA [23], UBL [24], etc.) typically contain hundreds of XML message schemas and thousands of schema elements. As these specifications evolve over time incorporating new requirements, their complexity increases even further. Design of such XML schemas typically follows document engineering [25–27] or a similar methodology that produces XML documents by aggregating data elements based on pre-defined simple and complex types [28]. For example, OTA message level schemas are constructed by aggregation of simple (OpenTravel Simple Types) and complex (OpenTravel Common Types, and Industry Common Types) schema elements [23]. This design approach results in overlapping message schemas and high levels of data coupling reducing the maintainability of services implemented using these specifications [4, 29].

Ensuring XML schema design quality in the context of SOA applications presents a particularly difficult problem as the schemas are often developed in the absence of a domain data model [30]. Current work in this area includes research that focuses on identifying dependencies among schema elements and developing tools for automating the propagation of these changes to all dependent schema components. 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 is proposed. Composite operations are constructed from atomic operations to implement complex schema changes [21, 31, 32].

Evaluation of the quality of design of XML schemas is another research direction that has attracted recent attention [2, 30, 33]. Numerous XML schema quality metrics have been proposed primarily with the objective to measure various aspects of schema complexity. McDowell et al. proposed eleven metrics and two composite indexes to measure the quality and complexity of XML schemas. These metrics are based on counts of complex type declarations, derived complex types, number of global type declarations, number of simple types, and element fanning (*fan-out* – number of child elements that an element has, and *fan-in* – number of times that an element is referenced) [30]. The authors formulate a *Quality Index* and a *Complexity Index* that estimate the quality and complexity of XML schemas based on weighted values of the metrics. A metric analysis tool is provided for developers to verify the validity of the metrics in the context of specific projects.

The concept of entropy [34] has been adapted for the measurement of complexity of software, and was initially applied to procedural software [35], and later to object-oriented design [36, 37]. Ruellan [38] used an entropy measure to assess the amount of information contained in XML documents (information density) with the objective to reduce the size of XML documents and to improve processing speed of XML messaging applications. Thaw et al. [39] proposed entropy-based metrics to measure reusability, extensibility, understandability of XML schema documents. Basci et al. [2] 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. The authors used the concept of Schema Entropy (SE) to assess XML schema complexity. SE is evaluated based on the complexity of schema elements as measured by fan-in and fan-out, and the number of simple elements that constitute individual schema elements. The SE metric was empirically validated using publicly available XML schemas, and the authors concluded that the metric provides a useful feedback when comparing schemas with equal number of elements [11]. In [40] Tang et al. apply an entropy-based measure to assessing the structural uniformity (*structuredness)* of XML documents. Two metrics are defined: Path-Based Entropy and Subtree-Based Entropy metrics that attempt to measure the level of diversity of a set of XML documents. Unlike Basci et al. [2, 11], the authors base the entropy calculation on XML documents, rather than XML schemas. Pichler et al. [22] developed a set of metrics to analyse the complexity of business documents with the objective of estimating the effort involved in data element mapping between different business document standards.

In summary, different XML schema features, including the number of schema elements, number of complex types, fan-in and fan-out have been used to measures schema complexity. Our approach (described in the next section) is an adaption of the Class Definition Entropy (CDE) metric for object-oriented design developed by Bansiya et al. [36] that measures schema complexity by estimating entropy based on complex schema elements.

3 Service Entropy Metric (SEM)

In this section we describe our approach to estimating the complexity of services using an entropy-based metric. Software services typically use the request/response message exchange pattern and have an interface that can be described by:

S(M_RQ, M_RS)

where **S** is a service and **M_RQ** and **M_RS** are the request and response messages, respectively. For example, the OTA flight booking service has the following interface:

AirBook(OTA_AirBookRQ, OTA_AirBookRS)

where OTA_AirBookRQ and OTA_AirBookRS are the request and response messages, respectively.

We estimate the complexity of the service **S** as the sum of the entropies of the request and response messages, based on the Message Schema Entropy (MSE) [41]. In our formulation of the MSE metric we adapt the approach of Bansiya et al. [36] who developed the Class Definition Entropy (CDE) metric for object-oriented design.

The CDE metric evaluates the frequency of occurrence of name strings for a given class; our MSE metric computes the frequency of occurrence of complex schema elements in a given XML message schema (e.g. the OTA_AirAvailRQ schema). MSE entropy is computed as:

$$\mathrm{MSE} = -\sum_{j=1}^{N} (P_i \log_2 P_i)$$

where:

N = total number of unique complex elements in the message schema

$\mathbf{n_i}$ = number of occurrences of the i^{th} complex element in the message schema

M = total number of (non-unique) complex elements in the message schema

$P_i = n_i/M$

MSE calculation is based on counting complex schema elements (i.e. elements based on complex types) and represents an approximation, as the complexity of individual element substructures is not taken into account. OTA differentiates between *complex types* that contain multiple data elements, and *simple types* that contain a single data element. In addition to globally defined schema elements (common data types), individual message schemas include locally defined elements. For example, the OTA_AirAvailRQ message that is used to implement the (web) service for flight availability inquiry includes 428 elements with multiple levels of nesting. OTA defines common data types (OTA_AirCommonTypes) for the airline messages that form a global type repository of XML Schema components used in the construction OTA Air messages. While the level of nesting and the number of simple elements that constitute the messages contribute to complexity of the services, their inclusion into the metric calculation involves assigning arbitrary weights and can make the interpretation of the metric more difficult.

As entropy values are additive, we can calculate SEM as the sum of the entropies of request and response messages:

$$\mathrm{SEM(S)} = \mathrm{MSE(M\_RQ)} + \mathrm{MSE(M\_RS)}$$

We have developed a prototype XSD Analyzer tool that calculates the values of the MSE and SEM metrics. XSD Analyzer allows the selection of message schemas and produces an output that includes the total number of non-unique schema elements (M), the number of unique (distinct) schema elements (N), counts of occurrences of individual schema elements, and the values of MSE and SEM, for the selected service interface.

4 Experimental Results Using the OTA Air Message Schemas

In this section we use the SEM metric to estimate the complexity of services based on the OTA Air messages specification. OTA Air messages are a subset of the OTA specification for services that support various business functions related to airline travel, such as checking flight availability, flight booking, etc. For example, the Search and Availability of flights service is implemented using the Air_AvailabilityRQ/RS request/response message pair [42].

Table 1 shows SEM values for a subset of eight OTA Air services for the period of 2006 to 2014. It is evident that complexity of the services as measured by SEM increases as the specification evolves over time. The increase in SEM ranges from 11.2 % for the AirBook service to 24 % for the AirDemandTicket service over the nine-year period.

All services shown in Table 1 increase in complexity over time, indicating that as new elements are added to extend the functionality of the services, existing elements are retained to maintain compatibility with previous versions of the specification. Figures 1 and 2 show the values of SEM for the AirAvailability (AirAvail) and AirBook (AirBook) services for the period of 2006 to 2014. Complexity of both services increases over time, in particular in the period around 2011 that most likely corresponds to a major revision of the specification.

Table 1. Values of SEM for OTA Air Services (versions 2006-2014)

OTA Air Service	2006	2007	2008	2009	2010	2011	2012	2013	2014
AirAvail	10.66	10.82	11.04	11.12	11.13	11.24	12.60	12.60	12.60
AirBook	13.62	13.84	14.13	14.30	14.59	14.54	15.26	15.19	15.14
AirCheckIn	12.45	12.49	12.74	13.01	13.06	13.06	14.24	14.15	14.10
AirDemandTicket	9.25	9.45	9.52	9.62	9.81	10.88	11.53	11.50	11.47
AirFareDisplay	11.33	11.33	11.54	11.54	11.57	11.57	12.94	12.94	12.94
AirLowFareSearch	11.95	12.15	12.57	12.57	12.57	12.53	14.59	14.59	14.59
AirPrice	11.60	11.83	12.45	12.63	12.63	12.63	13.42	13.42	13.42
AirSeatMap	10.05	10.05	10.05	10.05	10.14	10.43	11.35	11.35	11.35

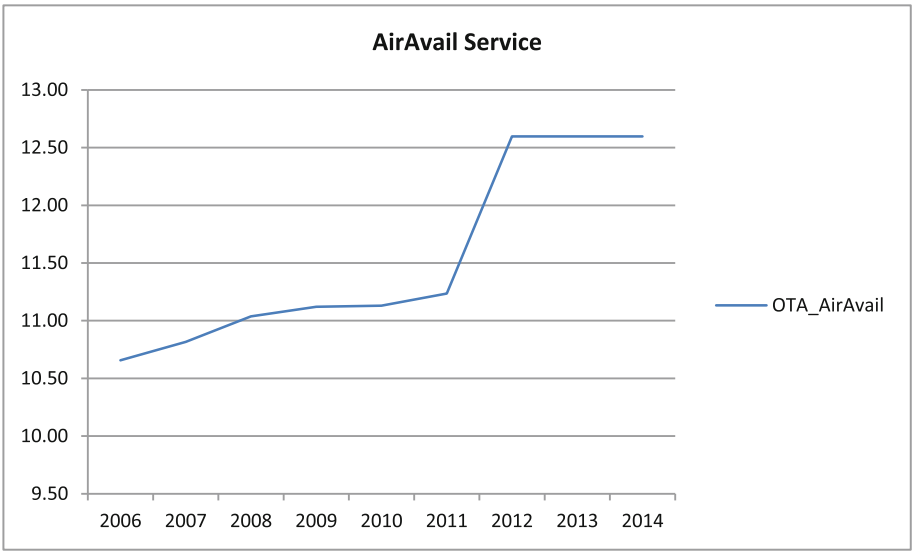

Fig. 1. Increase in complexity of the AirAvail service

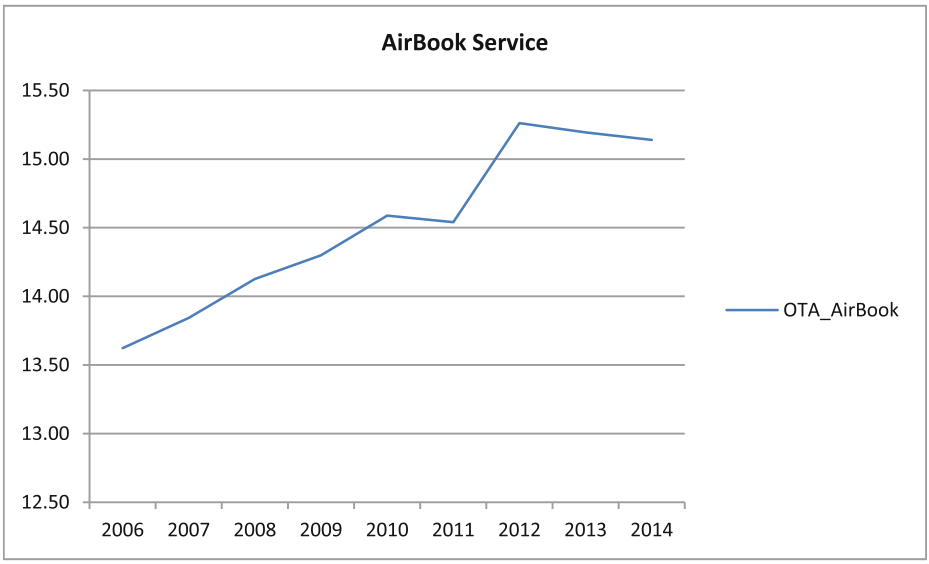

Fig. 2. Increase in complexity of the AirBook service

5 Conclusions and Further Work

The focus of this paper is the estimation of complexity of services based on the analysis of the underlying message schemas. Following a review of related literature in Sect. 2, we have defined an entropy-based service complexity metrics and used the XSD Analyzer tool to compute the values of SEM for eight OTA Air services over a period of nine years (2006–2014). The results indicate an almost monotonic increase in service complexity over the nine-year period, with the increase in SEM ranging from 11.2 % (for the AirBook service) to 24 % (for the AirDemandTicket service). While the exact significance of this increase is difficult to determine without further investigation, it can be argued that an increase in the complexity of the specification leads to a reduction in application development productivity. We note here that we use the OTA specification as an example of a domain specification designed using the document engineering approach, and that we do not imply any criticism of the quality of the OTA specification. Some researchers (in particular in the REST community) argue that large complex schemas are not problematic as long as they have a well-designed extension model and that the resulting coarse-grained services simplify the interaction dialog and have performance advantages in unreliable network environments. We argue that as the reliability of the Internet improves, the impact of excessive complexity on the maintainability of the services will outweigh such performance advantages.

The current version of the SEM metric is purely based on complex element counts and it does not take into account the internal complexity of individual elements (i.e. the sub-structure of complex elements). This makes it relatively easy to interpret the metric, but it also reduces the accuracy of the estimates of service complexity. We are currently working on improving the *sensitivity* of the SEM metric by incorporating a

range of structural features (i.e. number of schema levels, number of simple elements, etc.) into the calculation of the metric.

References

1. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures (2000). http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
2. Basci, D., Misra, S.: Measuring and evaluating a design complexity metric for XML schema documents. J. Inf. Sci. Eng. **25**(5), 1405–1425 (2009)
3. Bansiya, J., Davis, C.G.: A hierarchical model for object-oriented design quality assessment. IEEE Trans. Softw. Eng. **28**(1), 4–17 (2002)
4. Etzkorn, L.H., et al.: A comparison of cohesion metrics for object-oriented systems. Inf. Softw. Technol. **46**(10), 677–687 (2004)
5. Eder, J., Kappel, G., Schrefl, M.: Coupling and cohesion in object-oriented systems. Technical report, University of Klagenfurt, Austria (1994)
6. 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)
7. Papazoglou, M.P., Heuvel, W.V.D.: Service-oriented design and development methodology. Int. J. Web Eng. Technol. **2**(4), 412–442 (2006)
8. Feuerlicht, G.: Simple metric for assessing quality of service design. In: Maximilien, E., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 133–143. Springer, Heidelberg (2011)
9. Feuerlicht, G.: Evaluation of quality of design for document-centric software services. In: Ghose, A., Zhu, H., Yu, Q., Delis, A., Sheng, Q.Z., Perrin, O., Wang, J., Wang, Y. (eds.) ICSOC 2012. LNCS, vol. 7759, pp. 356–367. Springer, Heidelberg (2013)
10. Gonzalez, R.R.: A unified metric of software complexity: measuring productivity, quality, and value. J. Syst. Softw. **29**(1), 17–37 (1995)
11. Basci, D., Misra, S.: Entropy as a measure of quality of XML schema document. Int. Arab J. Inf. Technol. **8**(1), 75–83 (2011)
12. 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)
13. Perepletchikov, M., Ryan, C., Frampton, K.: Cohesion metrics for predicting maintainability of service-oriented software. In: QSIC, pp. 328–335 (2007)
14. Vinoski, S.: Old measures for new services. IEEE Internet Comput. **9**(6), 72–74 (2005)
15. 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, China (2008)
16. 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, Spain (2009)
17. Stevens, W.P., Myers, G.J., Constantine, L.L.: Structured design. IBM Syst. J. **38**(2&3), 115–139 (1999)
18. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-Oriented Modeling and Design. Prentice Hall, New Jersey (1991)
19. Chidamber, S., Kemerer, C.: A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**(6), 476–493 (2002)

20. OTA: OTA Specifications (2010). http://www.opentravel.org/Specifications/Default.aspx. Accessed 6 May 2010
21. Necaský, M.: Conceptual modeling for XML. Dissertations in Database and Information Systems Series. IOS Press/AKA Verlag (2009)
22. Pichler, C., Strommer, M., Huemer, C.: Size matters!? Measuring the complexity of xml schema mapping models. In: 2010 6th World Congress on Services (SERVICES-1). IEEE (2010)
23. OTA: Open Travel Aliance Specification (2014). http://www.opentravel.org/Specifications/Default.aspx. (cited 6 May 2014)
24. OASIS Universal Business Language (2014). https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl
25. Glushko, R., McGrath, T.: Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services. MIT Press Books, Cambridge (2008)
26. Glushko, R. McGrath, T.: Patterns and reuse in document engineering. In: XML 2002 Proceedings (2002)
27. 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)
28. ebXML - Enabling A Global Electronic Market (2007). http://www.ebxml.org/. (cited 9 December 2007)
29. Feuerlicht, G., Lozina J.: Understanding service reusability. In: 15th International Conference on Systems Integration 2007. VSE Prague, Prague, Czech Republic (2007)
30. McDowell, A., Schmidt, C., Yue, K.B.: Analysis and metrics of XML schema (2004)
31. 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 2010, pp. 38–49 (2010)
32. Necaský, M. Mlýnková, I.: Five-level multi-application schema evolution. In: Proceedings of the Databases, Texts, Specifications, and Objects, DATESO 2009, pp. 213–217 (2009)
33. Visser, J.: Structure metrics for XML Schema. In: Proceedings of XATA (2006)
34. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mobile Comput. Commun. Rev. **5**(1), 3–55 (2001)
35. Mohanty, S.N.: Entropy metrics for software design evaluation. J. Syst. Softw. **2**(1), 39–46 (1981)
36. Bansiya, J., Davis, C., Etzkorn, L.: An entropy-based complexity measure for object-oriented designs. Theory Pract. Object Syst. **5**(2), 111–118 (1999)
37. Olague, H.M., Etzkorn, L.H., Cox, G.W.: An entropy-based approach to assessing object-oriented software maintainability and degradation-a method and case study. In: Software Engineering Research and Practice. Citeseer (2006)
38. Ruellan, H.: XML Entropy Study. In: Balisage: The Markup Conference (2012)
39. Thaw, T.Z., Khin, M.M.: Measuring qualities of XML schema documents. J. Softw. Eng. Appl. **6**, 458 (2013)
40. Tang, R., Wu, H., Bressan, S.: Measuring XML structured-ness with entropy. In: Wang, L., Jiang, J., Lu, J., Hong, L., Liu, B. (eds.) WAIM 2011. LNCS, vol. 7142, pp. 113–123. Springer, Heidelberg (2012)
41. Feuerlicht, G., et al.: Measuring complexity of domain standard specifications using XML schema entropy. In: SOFSEM 2015. CEUR (2015)
42. Alliance, O.T: OpenTravel™ Alliance XML Schema Design Best Practices (2010). http://www.opentravel.org/Resources/Uploads/PDF/OTA_SchemaDesignBestPracticesV3.06.pdf. Accessed 1 Sept 2010