FIT for SOA? Introducing the F.I.T.-Metric to Optimize the Availability of Service Oriented Architectures

Sebastian Frischbier, Alejandro Buchmann, Dieter Pütz

Abstract The paradigm of service-oriented architectures (SOA) is by now accepted for application integration and in widespread use. As an underlying key-technology of cloud computing and because of unresolved issues during operation and maintenance it remains a hot topic. SOA encapsulates business functionality in services, combining aspects from both the business and infrastructure level. The reuse of services results in hidden chains of dependencies that affect governance and optimization of service-based systems. To guarantee the cost-effective availability of the whole service-based application landscape, the real criticality of each dependency has to be determined for IT Service Management (ITSM) to act accordingly. We propose the FIT-metric as a tool to characterize the stability of existing service configurations based on three components: functionality, integration and traffic. In this paper we describe the design of FIT and apply it to configurations taken from a production-strength SOA-landscape. A prototype of FIT is currently being implemented at Deutsche Post MAIL.

1 Introduction

A company's IT Service Management (ITSM) has to fulfill conflicting demands: while minimizing costs, IT solutions have to support a wide range of functionality, be highly reliable and flexible [22]. The paradigm of service-oriented architectures (SOA) has been proposed to solve this conflict. SOA was intended to facilitate the integration of inter-organizational IT-systems, thus becoming a key enabler of cloud computing [12]. At present, it is used mostly for intra-organizational application

Sebastian Frischbier, Alejandro Buchmann
Databases and Distributed Systems Group, Technische Universität Darmstadt,
e-mail: `lastname@dvs.tu-darmstadt.de`

Dieter Pütz
Deutsche Post AG, Bonn, e-mail: `d.puetz@deutschepost.de`

integration. Especially large companies, such as Deutsche Post DHL, use SOA to integrate and optimize their historically grown heterogeneous application landscape.

From an architectural point of view, the SOA paradigm reduces complexity and redundancy as it restructures the application landscape according to functionality and data-ownership. Basic entities within a SOA are services organized in domains without overlap. Each service encapsulates a specific function with the corresponding data and is only accessible through an implementation-independent interface. Services are connected according to a given workflow based on a business process [29].

From an infrastructure point of view, services are usually provided and consumed by applications. With the term 'application' we refer to large-scale complex systems, themselves quite often consisting of multi-tier architectures running on server clusters serving thousands of clients. These applications have to be available according to their business criticality. The desired level of availability is specified in service level agreements (SLA) in terms of service level targets (SLT). These differ according to the characteristics of the individual application and the means necessary to meet the desired level of availability [7]. Usually, the different service levels can be grouped into three classes: *high* availability (gold), *medium* availability (silver) and *low* availability (bronze). Effort and means needed for the operations provider to guarantee a given level of availability are reflected in the costs.

Deciding on the proper level of availability and defining adequate service levels is a difficult task, which becomes even more complex in service-based and distributed environments. The SOA paradigm drives the reuse of existing services by enabling their transparent composition within new services. As functionality and data are encapsulated, services have to rely on other services in order to run properly. This results in a network of hidden dependencies, since each service is only aware of its own direct dependencies on the consumed services. These dependencies affect the availability of applications directly as applications rely on other services to communicate and access information in service-based environments. Chains of interdependent services can lead to an application with higher availability becoming dependent on an application with lower availability even if the applications have no direct semantic relationship.

IT Service Management has to decide on the criticality of such a relationship and act accordingly. Criticality in this context does not refer to the *probability* of a breakdown actually taking place but to the *impact* on the application landscape once it occurs. The following approaches are possible to cope with disparities in service levels of depending applications: i) all participating applications are operated on the highest level of availability present in a chain of dependencies; ii) the configuration stays unchanged; iii) the SLA of single participants is changed to minimize the expected impact. As the "methods used are almost always pure guesswork, frequently resulting in drastic loss or penalties" [38, 43], the first approach is often favored. Although it succeeds, it is surely inefficient and expensive. Even hosting services alternatively in cloud environments rather than on-premise does not solve this problem. It rather shifts the risk of availability management to the cloud provider who will charge for it.

Due to the lack of proper decision support, both the second and third approach are usually avoided as they may result in serious breakdowns and loss of revenue. Therefore, ITSM requires methods and tools to: i) model all relevant dependencies; ii) identify hotspots and; iii) decide on their criticality. In particular, deciding on the criticality is important as this allows for ranking hotspots (e.g. as preparation for closer inspection) and simulating changes.

We introduce the FIT-metric to aid ITSM in these tasks, especially in deciding on the criticality of dependencies in existing service-oriented architectures and simulating changes. Our metric consists of the three components: functionality, integration and traffic. The necessary data can be cost-effectively obtained from end-to-end monitoring and existing documentation. FIT is the result of our analysis conducted at Deutsche Post MAIL and is currently implemented there.

The contributions of this paper are: i) we identify the need for applications and their dependencies to be ranked according to their criticality; ii) we propose a metric taking into account functionality, integration and traffic of the services involved to aid ITSM in assessing the appropriate service level in interdependent SOA-based systems; iii) we evaluate our approach by applying it to actual service configurations taken from a product-strength SOA-landscape.

The structure of this paper is as follows: we present a production-strength SOA in Sect. 2 to point out the need for a criticality-metric for service-dependencies. In Sect. 3 we present the design of our FIT-metric in detail. We use a case study to evaluate our approach in Sect. 4. We conclude our paper by reviewing related work on the topic of metrics for service-oriented architectures in Sect. 5 followed by a summary of our findings and a short outlook on future work in Sect. 6.

2 A Production-Strength SOA Environment

Deutsche Post AG is the largest postal provider in Europe, with Deutsche Post MAIL division alone delivering about 66 million letters and 2.6 million parcels to 39 million households in Germany each working day. Furthermore, Deutsche Post MAIL increases the level of digitalization in its product-portfolio (e.g. online and mobile based value-added services) since 2009 [8]. In 2010 Deutsche Post MAIL started to offer the *E-Postbrief* product to provide consumers and business users with a secure and legally compliant form of electronic communication [9].

The application landscape supporting these processes and products was transformed to apply the SOA-paradigm in 2001 [14]. Today, applications communicate across a distributed enterprise service bus (ESB) by consuming and providing SOA-services that are grouped in mutually exclusive domains. The initially used SOA-framework has been a proprietary development by Deutsche Post MAIL called

Service-oriented Platform (SOP). Today, SOP's open-source successor SOPERA is at the heart of both *Eclipse SOA*[1] and *Eclipse Swordfish*[2].

The results discussed here are based on our analysis of the Deutsche Post MAIL SOP/SOPERA application landscape using a Six Sigma-based [11] approach. This included conducting interviews, reviewing documentation as well as assessing monitoring capabilities to identify dependencies and business criticalities. All data presented in this paper is anonymized due to confidentiality requirements.

The main findings of our analysis are: i) long chains of dependencies between services affect the availability of applications in matured production-strength SOA-landscapes as the SOA-paradigm itself fosters the reuse of services; ii) SOA-dependencies are hard to uncover for ITSM at runtime as they are hidden in the SOA-layer itself; iii) the data necessary to identify and analyze them at runtime may be already existing but is often not available at first as it is spread across different heterogeneous applications; iv) to the best of our knowledge, there is no metric available for ITSM to decide on the criticality of service-relationships based on the data usually available at runtime. Based on these findings: i) we initiated a specialized but extensive end-to-end monitoring of the SOA-application landscape to allow for dependencies and their usage to be quantified automatically in the future; ii) we defined a cost-effective criticality-metric based on the available data; iii) we built a prototypic software tool named *FIT-Calculator* to allow for automated graph-based analysis and simulation based on the monitored data.

The availability-'heat map' of the SOA-landscape as illustrated in Fig. 1 is automatically generated based on the monitoring data currently available to us. It gives an overview of 31 participating applications and their 69 service-relationships. Each node represents an application providing and consuming SOA-services. The desired level of availability for each node x is expressed by the node's color as well as by an abbreviation in brackets ([g]old, [s]ilver and [b]ronze). Edges denote service-relationships between two applications with an edge pointing from the consuming application to the application providing the service (direction of request). Edge weights refer to the number of requests processed over this dependency within a given period. Dependencies of consuming SOA-services on providing SOA-services within an application are modeled as an overlay for each application (not shown).

This visualization allows ITSM to identify hotspots easily. Hotspots, in this context, refer to applications that cause potentially critical relationships by providing SOA-services to applications with higher levels of availability. In the given example, 8 hotspots (A1, A5, A10, A11, A16, A18, A20, A23) cause 11 potentially critical relationships. On the heat map in Fig. 1, these relationships are marked bold red with the hotspots being drawn as rectangles.

---

[1] `http://www.eclipse.org/eclipsesoa/`

[2] `http://www.eclipse.org/swordfish/`

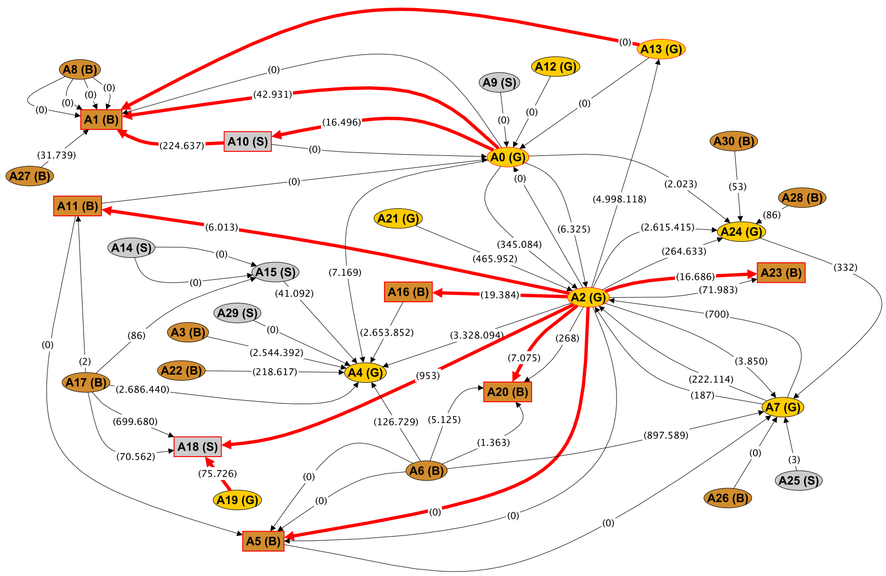


Fig. 1: Graph representing applications and their direct SOA-relationships.

3 Introducing the F.I.T.-Metric

The criticality of a single hotspot a depends on the criticality of each relationship between a and the depending applications with higher SLA. To us, the criticality of a single relationship $e(a,x)$ in general is primarily influenced by: i) the business relevance F_x of the application x directly depending on a via $e(a,x)$; ii) the impact of $e(a,x)$ on other applications in the SOA-landscape due to the integration $I_{a,x}$ of x (i.e. x serving as a proxy); iii) the actual usage $T_{a,x}$ of the relationship $e(a,x)$ by the depending application x.

F_x and $I_{a,x}$ refer to independent aspects of x's importance to the system landscape and the business users. An application's core function alone can be highly relevant to business users (e.g. business intelligence systems) while it may be unimportant for other applications from an integration point of view. In turn, an application serving mainly as a proxy for other applications can be relatively unimportant to business by its own. As these two aspects of a relationship are rather static (i.e. an application's core functionality is seldom altered completely over short time and dependencies between applications change only infrequently), they have to be weighted by an indicator for the actual usage of this relationship.

Therefore, we define the criticality $eFIT_{e(a,x)}$ of the relationship $e(a,x)$ as the sum of F_x and $I_{a,x}$, weighted by $T_{a,x}$ in Eq. (1).

$$eFIT_{e(a,x)} = (F_x + I_{a,x}) \cdot T_{a,x} \qquad (1)$$

In turn, the sum of these relationship-criticalities for all relationships to a defines the criticality FIT_a of hotspot a as defined in Eq. (2).

$$FIT_a = \sum_{\forall e(a,x)} eFIT_{e(a,x)} \tag{2}$$

In this setting, uncritical relationships and applications have a FIT-score of 0 while critical hotspots are ranked ascending by their criticality with FIT-scores > 0.

3.1 Component I: Functionality

Functionality refers to quantifying an application's relevance to business. As the business impact of IT systems is hard to determine and even harder to quantify from the IT point of view [38], these categorizations are often based on subjective expert knowledge and individual perception. Although this makes an unbiased comparison between applications difficult, we suggest reusing already existing information as an approximation. For example, we turn to data regarding business continuity management (BCM). In this context, applications have to be categorized according to their recovery time objective (RTO) in case of disaster [7]. The RTO-class $RTOC_x = 1, \ldots, n$ increases with the duration allowed for x to be unavailable. The *economic* $RTOC_x$ ($econRTOC_x$) is the RTOC the users are willing to pay and is inversely proportional to the quality level of the SLA. Therefore we define the assumed business relevance F_x of application x as:

$$F_x = \frac{1}{econRTOC_x} \tag{3}$$

3.2 Component II: Integration

Integration quantifies the impact of an inoperative application on all other applications based on dependencies between SOA-services. Information about these dependencies can be drawn at runtime from workflow-documentation (e.g. BPEL-workflows or service descriptions) or service monitoring.

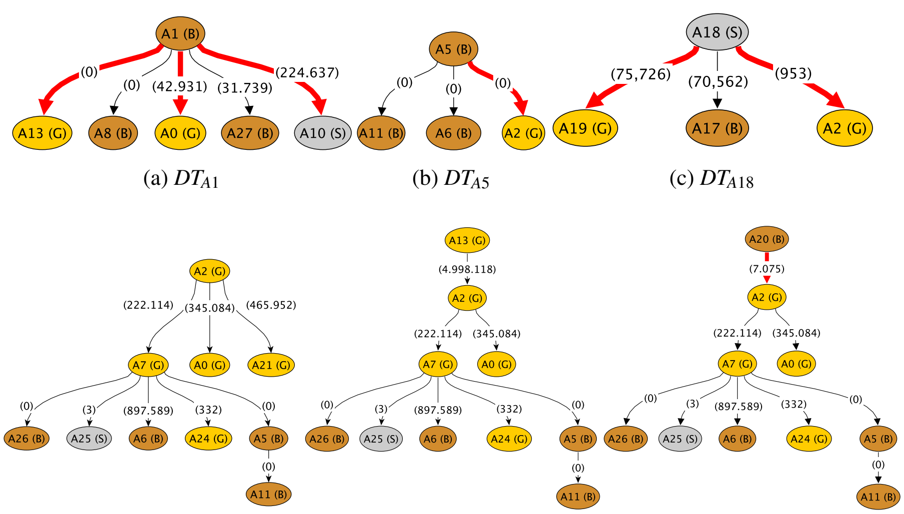
We define the dependency tree DT_a for each application a as the first step. The tree's root node is the initial application a itself. All direct consumers $tSC_{1,1}, \ldots,$ $tSC_{1,m}$ of this application's services are added as the root's children. On the following levels $i = 2, \ldots, h$ only applications that are *indirectly dependent* on services provided by a are added as $tSC_{i,1}, \ldots, tSC_{i,n}$. Thus, DT_a is not identical to the simple graph of *a's* predecessors in Fig. 1 as the nodes on level i depend on the internal dependencies inside applications (services depending on services).

Figures 2a- 2f show the dependency trees for the applications A1, A2, A5, A13, A18 and A20 based on the relationship graph shown in Fig. 1 and the overlay modeling internal dependencies of services inside applications. Edge weights denote the number of requests processed over a service-dependency alone as well as bold red edges representing possibly critical relationships. Here, edges point from service provider to service consumer (direction of response).

The weighted dependency tree wDT_a^x quantifies the direct and indirect dependencies on a over $e(a,x)$ by weighting the sub-tree of DT_a with application x as root. For DT_{A2} (cf. Fig.2d), the corresponding wDT_{A2}^{A7} would consist of A7 (root), A26, A25, A6, A24, A5 and A11.

Deep dependency trees containing long chains of indirect dependencies have a far-reaching impact on the landscape once the root node breaks down. They have to be emphasized as they are far less obvious to ITSM than a huge number of direct dependencies on an application. Therefore, wDT_a^x takes into account the assumed business relevance of each node $tSC_{i,j}$ in DT_a as well as the length of the dependency chain to $tSC_{i,j}$. The occurrence of each node $tSC_{i,j}$ is weighted with its functionality $F_{tSC_{i,j}}$ and its level of occurrence in the dependency tree. We define the integration of x as depending on a as:

$$I_{a,x} = wDT_a^x = \sum_{i=2}^{h} i \cdot \sum_{j=1}^{m} F_{tSC_{i,j}} \tag{4}$$



(a) DT_{A1} (b) DT_{A5} (c) DT_{A18}

(d) DT_{A2} (e) DT_{A13} (f) DT_{A20}

Fig. 2: Dependency trees DT of selected nodes (c.f. Fig. 1).

3.3 Component III: Traffic

Traffic quantifies the real usage of a given relationship between two applications a and x. As both F_x and $I_{a,x}$ refer to the worst-case impact of a breaking down, we need to balance this with an approximation for the current utilization of the relationship $T_{a,x}$. In order to get such an approximation from the data available to us, we relate the number of requests by x to a over a given critical edge $e(x,a)$ to the total number of requests by x:

$$T_{(a,x)} = \frac{cREQ_{e(x,a)}}{\sum_{\forall e(x,i)} REQ_{e(x,i)}} \tag{5}$$

4 Case Study: Applying FIT to a Real Application Landscape

We test our approach with data taken from the SOA environment presented in Sect. 2. We discuss the criticality of the initial 8 hotspots identified on the heat map in Fig. 1 (rectangular nodes). We simulate two alternative SLA-structures (*scenario 1* and *scenario 2*) aimed at eliminating the two most critical hotspots and discuss the effects. The FIT-scores of the 8 initial hotspots are listed in Table 1 in descending order. The detailed values to retrace how the initial FIT-scores for A1, A5, A18 and A20 were obtained are also shown in Table 1. We discuss selected applications.

A18 (silver) is deemed the most critical hotspot as it causes a relationship with criticality $eFIT_{e(A18,A19)} = 3$ to A19 (gold). As can be seen in Fig. 2c and Table 1, the link $e(A18,A19)$ carries 100% of the requests of service that A19 makes to A18. In contrast only 0.00008% or 953/11,332,472 requests of service of A2 on A18 occur along $e(A18,A2)$. Therefore, even though both A19 and A2 have SLA gold and depend on A18 with SLA silver, only $e(A18,A19)$ is critical and will require adjustment of the SLA of A18 or A19.

A1 (bronze) is ranked the second critical hotspot mainly because two of its three critical relationships to applications with higher SLAs are in heavy use (cf. Fig. 2a). A10 (silver) relies fully (100%) on A1 while A0 (gold) processes 10% of its total traffic over the critical relationship $e(A1,A0)$. As the most impacted application has only SLA silver, this relationship is ranked lower than the relationship $e(A18,A19)$ discussed before where A19 has gold level.

A20 (bronze) is ranked relatively uncritical in spite of the large body of important applications depending indirectly on it (c.f. Fig. 2f). This is mainly due to the low usage of its relationship to A2, accounting for only 0.1% of the total traffic produced by A2. Nevertheless, the heavy dependency tree of A2 weights this relationship $e(A20,A2)$ more critical than the relationship $e(A18,A2)$ as discussed earlier.

A5 (bronze) is ranked with criticality 0, putting it on the same level as an uncritical configuration (e.g. application A2, c.f. Fig. 2d and Table 1). Although there is a potentially critical relationship to an application with high availability, there is no traffic across this relationship within the measured period (c.f. Fig. 2b). Therefore,

Initial		**Scenario 1**		**Scenario 2**		**FIT-score components for A1, A2, A18, A20**							
h	FIT_h	h	FIT_h	h	FIT_h	a	x	$eFIT_{e(a,x)}$	F_x	$I_{a,x}$	$T_{a,x}$	$cREQ_{e(x,a)}$	$\sum REQ_{e(x,y)}$
A18	3	A18	0	A18	0	A1	A0	0.307	3	0	0.102	42,931	420,028
A1	2.307	A1	2.307	A1	0	A1	A8	0	1	0	0	0	0
A10	0.118	A10	0.118	A13	18.083	A1	A10	2	2	0	1	224,637	224,637
A23	0.063	A23	0.063	A23	0.06	A1	A13	0	3	0	0	0	0
A20	0.027	A20	0.027	A20	0.026	A1	A27	0	1	0	0	0	31,739
A11	0.021	A11	0.021	A11	0.02	A2	A0	0	3	0	0	0	420,028
A16	0.005	A16	0.005	A16	0.005	A2	A7	0	3	19	0	0	346,735
A5	0	A5	0	A5	0	A2	A21	0	3	0	0	0	583,490
						A18	A2	0	3	0	0	953	11,332,472
						A18	A17	0	1	0	0	0	3,456,770
						A18	A19	3	3	0	1	75,726	75,726
						A20	A2	0.027	3	40	0.001	7,075	11,332,472

Table 1: FIT-scores for hotspots h and component values for selected applications.

the importance of this relationship can be discarded as it seems to be unlikely that the relationship should be used exactly in the time of a downtime of A5. In addition, no other applications are indirectly depending on A5 over this relationship. Therefore, A5 can be assumed to be non-critical.

In *scenario 1* we now try to cost-effectively eliminate the hotspots top-down. We start with A18 by lowering the depending application A19's SLA to silver. Simulating the resulting SLA-structure shows that A18's criticality drops to 0 with no further negative impact on the surrounding applications. Based on this setting, we try to also eliminate hotspot A1 in *scenario 2* by leveling the SLAs of A0, A1, A10 and A13 to silver. Nevertheless, simulating this structure shows that A13 becomes a new hotspot with criticality $FIT_{A13} = 18$, beating A1 in its criticality. Therefore leveling all four applications should be discarded for being ineffective and other structures have to be simulated instead.

The examples discussed here in detail show the importance of assessing the potential hotspots identified on the heat map in Fig.1. Especially to balance static information (i.e. business-criticality of applications and their relationships) with factual usage is crucial to focus on the really critical hotspots. Simulating different structures based on this approach aids ITSM optimizing the availability of service-based application landscapes.

5 Related Work

Related work published over the past years deals with several types of metrics for SOA. To categorize these contributions we mapped them to the phases of the application management lifecycle: requirements specification, design, build, deploy, operate and optimize [7]. Most of the reviewed work deals with aspects and metrics to support the first four phases based on design-time data: Metrics to measure business alignment of SOA implementations [1, 30], procedures to model [6, 16] and implement SOA [3], including the prediction of development effort and implementation complexity early in the design phase [37]. Metrics to measure granularity, complexity and reuse [15, 35, 36], performance [5, 13] and QoS [28, 31] of SOA-based services also rely on design-time data. Most work on operation and optimization has been done on how to handle service level agreements primarily based on design-time data: how to formally describe them [19, 34, 39, 40], technically implement, test and enforce them [4, 10, 15, 17, 18, 23, 25, 26, 32, 33, 42, 44, 45] or how to monitor them [2, 20, 21]. Contributions available on SLA design deal with isolated approaches: Sauvé et al. [38] and Marques et al. [27] are in favor of deriving the service level targets directly from the business impact of the given service (i.e. taking into account the risk of causing revenue loss on the business layer). Li et al. [24] and Smit et al. [41] focus on infrastructure aspects of specific applications.

Most of these contributions require customized frameworks or rely massively on design-time data and services being designed as glass-boxes. None of these contributions propose a solution how to characterize the criticality of an existing service configuration in historically grown heterogeneous application landscapes based on runtime data provided by end-to-end monitoring. Nevertheless, this is crucial for ITSM to decide on changes in the SLA-structure cost-effectively.

6 Conclusion and Outlook

SOA reduces the complexity of system integration. However, it increases the problems of governance and availability management on the infrastructure level because of hidden dependencies among services. As services are transparently reused, applications with higher SLAs can become dependent on applications with lower SLAs, thus creating hotspots in the SLA structure. To guarantee overall cost-effective availability in such a setting, ITSM has to identify these hotspots and decide on their criticality.

In this paper, we proposed the FIT-metric based on the three components: function, integration and traffic. Our metric allows ranking hotspots and their relationships according to their criticality for ITSM. Based on this ranking, different alternative SLA-structures and their impact can be simulated. Therefore, the contributions of this paper are threefold: i) we showed the need for a criticality metric in a historically grown production-strength SOA-landscape; ii) we presented the cost-effective FIT-metric to rank hotspots and their relationships according to their criticality for ITSM to optimize SLA levels; iii) we demonstrated our approach by applying it to actual service configurations.

We are about to conclude the implementation of our prototype at Deutsche Post MAIL. This includes finishing the rollout of our service monitoring and reporting to allow for more extensive analysis in the future (e.g. include data about latencies in FIT). As part of our future work we want to apply our findings to other loosely coupled systems such as event-based systems (EBS). Today, SOA is mostly used intra-organizationally to implement given workflows within a single organization. Thus, critical knowledge about participants, their interdependencies and corresponding business impact are available in principle. Tomorrow's systems tend to become even more federated, distributed and loosely coupled. In those service-based inter-organizational systems availability management is even more difficult.

Acknowledgements We would like to thank Irene Buchmann, Jacqueline Pranke, Achim Stegmeier, Alexander Nachtigall and the two anonymous reviewers for their valuable input and discussions on this work. Part of this work is funded by German Federal Ministry of Education and Research (BMBF) under research grants ADiWa (01IA08006) and Software-Cluster project EMERGENT (01IC10S01), and by Deutsche Post MAIL. The authors assume responsibility for the content.

References

1. Aier, S., Ahrens, M., Stutz, M., Bub, U.: Deriving SOA evaluation metrics in an enterprise architecture context. In: ICSOC 2007 Workshops, pp. 224–233 (2009)
2. Ameller, D., Franch, X.: Service level agreement monitor (SALMon). In: ICCBSS 2008, pp. 224–227 (2008)
3. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., Holley, K.: SOMA: a method for developing service-oriented solutions. IBM Systems Journal **47**(3), 377–396 (2010)
4. Bause, F., Buchholz, P., Kriege, J., Vastag, S.: Simulation based validation of quantitative requirements in service oriented architectures. In: WSC 2009, pp. 1015–1026 (2009)
5. Brebner, P.C.: Performance modeling for service oriented architectures. In: ICSE Companion '08, pp. 953–954 (2008)
6. Broy, M., Leuxner, C., Fernndez, D.M., Heinemann, L., Spanfelner, B., Mai, W., Schlör, R.: Towards a Formal Engineering Approach for SOA. Techreport, Technische Universität München (2010). Available online at http://www4.informatik.tu-muenchen.de/publ/papers/TUM-I1024.pdf. Accessed on 21.06.2011.
7. Cannon, D.: ITIL Service Operation: Office of Government Commerce. The Stationery Office Ltd (2007)
8. Deutsche Post DHL: Deutsche Post CEO Frank Appel presents Strategy 2015. http://www.dp-dhl.com/en/investors/investor_news/news/2009/dpwn_strategie_2015.html (2010). Accessed on 18.03.2011.
9. Deutsche Post DHL: MAIL Division. http://www.dp-dhl.com/en/about_us/corporate_divisions/mail.html (2011). Accessed on 18.03.2011.
10. Di Modica, G., Regalbuto, V., Tomarchio, O., Vita, L.: Dynamic re-negotiations of SLA in service composition scenarios. In: EUROMICRO 2007, pp. 359–366 (2007)
11. Eckes, G.: Six SIGMA for Everyone, 1 edn. Wiley & Sons (2003)
12. Frischbier, S., Petrov, I.: Aspects of data-intensive cloud computing. In: From Active Data Management to Event-Based Systems and More, pp. 57–77. Springer (2010)
13. Her, J.S., Choi, S.W., Oh, S.H., Kim, S.D.: A framework for measuring performance in Service-Oriented architecture. In: NWeSP'07, pp. 55–60 (2007)
14. Herr, M., Bath, U., Koschel, A.: Implementation of a service oriented architecture at deutsche post MAIL. Web Services pp. 227–238 (2004)
15. Hirzalla, M., Cleland-Huang, J., Arsanjani, A.: A metrics suite for evaluating flexibility and complexity in service oriented architectures. In: ICSOC 2008, pp. 41–52 (2009)
16. Hofmeister, H., Wirtz, G.: Supporting Service-Oriented design with metrics. In: EDOC 2008, pp. 191–200 (2008)
17. Hsu, C., Liao, Y., Kuo, C.: Disassembling SLAs for follow-up processes in an SOA system. In: ICCIT 2008, pp. 37–42 (2008)

18. Kotsokalis, C., Winkler, U.: Translation of service level agreements: A generic problem definition. In: ICSOC/ServiceWave 2009, pp. 248–257. Springer (2010)
19. Kotsokalis, C., Yahyapour, R., Gonzalez, M.R.: Modeling service level agreements with binary decision diagrams. Service-Oriented Computing pp. 190–204 (2009)
20. Kunz, M., Schmietendorf, A., Dumke, R., Rud, D.: SOA-capability of software measurement tools. ENSUR A p. 216 (2006)
21. Kunz, M., Schmietendorf, A., Dumke, R., Wille, C.: Towards a service-oriented measurement infrastructure. In: SMEF 2006, pp. 10–12 (2006)
22. Kütz, M.: Kennzahlen in der IT. Werkzeuge für Controlling und Management, 2nd edn. Dpunkt Verlag (2007)
23. Lam, T., Minsky, N.: Enforcement of server commitments and system global constraints in SOA-based systems. In: APSCC 2009, pp. 126–133 (2009)
24. Li, H., Casale, G., Ellahi, T.: SLA-driven planning and optimization of enterprise applications. In: WOSP/SIPEW '10, pp. 117–128 (2010)
25. Liu, L., Schmeck, H.: Enabling Self-Organising service level management with automated negotiation. In: IEEE/WIC/ACM 2010, pp. 42–45 (2010)
26. Liu, L., Zhou, W.: A novel SOA-Oriented federate SLA management architecture. In: IEEC 2009, pp. 630–634 (2009)
27. Marques, F., Sauvé, J., Moura, A.: Service level agreement design and service provisioning for outsourced services. In: LANOMS 2007, pp. 106–113 (2007)
28. Mayerl, C., Huner, K.M., Gaspar, J., Momm, C., Abeck, S.: Definition of metric dependencies for monitoring the impact of quality of services on quality of processes. In: IEEE/IFIP 2007, pp. 1–10 (2007). DOI 10.1109/BDIM.2007.375006
29. McGovern, J., Sims, O., Jain, A.: Enterprise service oriented architectures: concepts, challenges, recommendations. Kluwer Academic Pub (2006)
30. O'Brien, L., Brebner, P., Gray, J.: Business transformation to SOA. In: SDSOA '08, pp. 35–40 (2008)
31. O'Brien, L., Merson, P., Bass, L.: Quality attributes for service-oriented architectures (2007)
32. Palacios, M., Garcia-Fanjul, J., Tuya, J., de la Riva, C.: A proactive approach to test service level agreements. In: ICSEA 2010, pp. 453–458 (2010)
33. Parejo, J.A., Fernandez, P., Ruiz-Corts, A., Garca, J.M.: SLAWs: towards a conceptual architecture for SLA enforcement. In: SERVICES-1 2008, pp. 322–328 (2008)
34. Raibulet, C., Massarelli, M.: Managing non-functional aspects in SOA through SLA. In: DEXA 2008, pp. 701–705 (2008)
35. Rud, D., Schmietendorf, A., Dumke, R.: Product metrics for service-oriented infrastructures. In proceedings of IWSM/MetriKon 2006 pp. 161–174 (2006)
36. Rud, D., Schmietendorf, A., Dumke, R.: Resource metrics for service-oriented infrastructures. SEMSOA 2007 pp. 90–98 (2007)
37. Salman, N., Dogru, A.: Complexity and development effort prediction models using component oriented metrics. ENSUR A (2006)
38. Sauvé, J., Marques, F., Moura, A., Sampaio, M., Jornada, J., Radziuk, E.: SLA design from a business perspective. DSOM 2005 pp. 73–84 (2005)
39. Schulz, F.: Towards measuring the degree of fulfillment of service level agreements. In: ICIC 2010, pp. 273–276 (2010)
40. Skene, J., Lamanna, D.D., Emmerich, W.: Precise service level agreements. In: ICSE 2004, pp. 179–188 (2004)
41. Smit, M., Nisbet, A., Stroulia, E., Edgar, A., Iszlai, G., Litoiu, M.: Capacity planning for service-oriented architectures. In: CASCON 2008, pp. 144–156 (2008)
42. Strunk, A.: An algorithm to predict the QoS-Reliability of service compositions. In: SERVICES 2010, pp. 205–212 (2010)
43. Taylor, R., Tofts, C.: Death by a thousand SLAs: a short study of commercial suicide pacts. Hewlett-Packard Labs (2005)
44. Thanheiser, S., Liu, L., Schmeck, H.: SimSOA: an approach for agent-based simulation and design-time assessment of SOC-based IT systems. In: SAC 2009, pp. 2162–2169 (2009)
45. Theilmann, W., Winkler, U., Happe, J., de Abril, I.: Managing On-Demand business applications with hierarchical service level agreements. In: FIS 2010, p. 97 (2010)