SOA Model for High Availability of Services

Tayyaba Anees and Heimo Zeilinger
Institute of Computer Technology
Vienna University of Technology
Gußhausstraße 27-29, 1040 Vienna, Austria
e-mail: {anees, zeilinger}@ict.tuwien.ac.at

Abstract— **Service-oriented Architecture (SOA) provides reusability and enables easy functionality integration. Service availability in SOA is important as it is used by safety critical systems, telecommunication systems and business systems. Service unavailability can result in reduced profits, reputation damage and reduced safety. Machine virtualization, clusters and group communication systems are used to increase availability, but they are not very much applied to SOA-based systems. This paper focuses on service-orientation and a model for increasing service availability in SOA is proposed. The proposed model improves failure detection process using monitoring. Use of heartbeat mechanism is proposed for failure detection instead of timeout mechanism as it can provide more accuracy and also it can reduce failure detection time. Model is emulated in LAN and WAN environments to investigate impact of different network configurations on service availability. Results indicate that service availability is increased and failure detection process is improved by monitoring.**

Keywords- service-oriented architecture; availability; high availability; monitoring; failover; safety critical systems; business systems; telecommunication systems

I. INTRODUCTION

Service-oriented architecture [1] is for flexibility and reuse, and enables organizations to easily integrate systems [2]. The term SOA followed in the paper is defined by the organization for advancement of structured information standards (OASIS) [1] as "…a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" [1]. Services are reusable, which can work autonomously as well as in service compositions. SOA follows a standard-based development approach [3]. Service availability is seen in the paper from service consumer's point of view. In our opinion, standards based development makes SOA-based systems more acceptable to service consumers and they can be trusted for quality.

According to authors', availability of services in SOA needs attention as unavailability of services can result in dissatisfaction among service consumers, lost revenues, damaged reputation for service providers and loss of human lives. One of the most important issues for SOA is to assure availability [4]. Business services today are not only doing more work but also have more users, often spread out across the globe and require 24/7 availability and availability is one of the important factors to be considered for business-driven IT service management [5]. The fundamental characteristic of SOA, loose coupling and on-demand integration, enable organizations to seek more flexibility and responsiveness from their business IT systems, but this brings challenges to assure QOS, especially availability, which should be considered in an integrated way in SOA [6].

SOA adoption is increasingly seen in the latest trends where safety critical systems, telecommunication systems and business systems are using it ([7], [8] and [9]). This tendency is due to reduced expected costs due to reusability, which is achievable by using SOA. Service availability is becoming a requisite for such systems as profits can only be earned if service functionality is available to service consumers. Many of these systems require not only availability, but instead high availability for safety as unavailability of service can cause information loss, which can put system into hazardous state, thus reducing system safety. In the paper, the term availability describes the probability that a service in a given time is available.

Availability is dependent on mean time to failure (MTTF) and mean time to repair (MTTR). In the paper, the qualitative description of high availability [11] is followed as highly available systems are those systems which are expected to operate correctly in the presence of multiple failures, using a subset of original components, with reduced capacity and system should be able to self-heal and reconstitute itself, without the loss of data or application services. The system must detect failures and reconfigure system operations dynamically. In our opinion, high availability is expensive and it is not required for all applications or services. Requirements for availability and high availability are dependent on area of application and also on a specific solution. In SOA, requirements for availability and high availability are generally specified in service level agreements (SLA).

This work focuses on increasing service availability by reducing failover time and failure detection time through monitoring. Failover means a backup module taking the workload, when the primary module has failed [12]. Failover time includes the failure detection time and the recovery time [12].

Clusters, group communication systems (GCS) and machine virtualization are solutions, which are used to increase availability. These solutions also use monitoring but they differ in concept, and they are not very much applied within the domains of SOA. In machine virtualization, several operating systems share the resources of same physical machine. Shared physical machine can increase performance overhead and it can become a single point of failure for virtualized solutions. Clusters use GCS and GCS based solutions require coordination activities between group members, which can impose performance overhead. SOA-based solutions for increasing availability are mostly focusing on service compositions. Middleware based solutions also exist, which use an enterprise service bus. These solutions can also add performance overhead, which can increase failover time.

The proposed approach focuses on the use of monitoring and heartbeat mechanism for failure detection instead of using timeouts for failure detection and retrying in case of failures. The proposed approach simplifies the management of failures by focusing only on the necessary participants of SOA-based systems including service consumers, service providers and service registry. Additionally, the focus is on atomic services instead of service compositions for simplification. In the proposed approach, service provider's availability is seen from service consumer's perspective because they are the ones who ultimately use the service and in this context failover time becomes important, which is the time when service is unavailable. Failures are covered in the approach through redundancy and service provider's availability is determined through failover time. The proposed approach improves the process of failure detection and failover by using heartbeat mechanism and service provider availability is improved by reducing failure detection time through monitoring service provider's failures and by selection of an optimal heartbeat interval.

The remainder of the paper is organized as follows: Section II describes the state of the art research work. In Section III we present and propose a SOA model for improving service availability. It also includes a discussion about the availability parameters considered in the proposed solution and describes the approach for analyzing service availability in SOA. Section IV presents experimental results and discussion. Section V contains concluding remarks.

II. STATE OF THE ART

SOA eases development efforts due to reusability and standards based development approach [3]. As stated by Li [4], SOA is an emerging approach addressing the requirements of loosely coupled, standards-based, and protocol independent distributed systems. Costs are reduced by reusability of components which turns out to be an advantage orchestrating large scale distributed applications [13]. These cost reductions lead to upcoming adoptions of SOA in business computing environments [8].

Recently, a shift in trends is seen and there is a move towards SOA adoption by safety critical systems. SOA is being adopted by military organizations such as the United States Department of Defense, The North Alliance Treaty Organization and the UK's Ministry of Defense [9]. Telecommunication networks are service centric and use service composition techniques in accordance to SOA principles [14].

Platforms that are supposed to form the core of mission critical service-oriented applications need mechanisms that can regulate the reliability and availability of the core services in changing conditions [15].

For increasing availability of services different solutions are proposed. In [16], a solution for improving availability of service compositions or complex services is proposed. Another solution is to pool multiple services that provide the same functionality by different service providers [4]. If a service fails, another service in the pool is selected to process the request again. In this approach, an appropriate size of service pool has to be selected otherwise resources can be underutilized. In the proposed approach, the focus is on reducing failure detection time and failover time by monitoring for increasing service provider availability. If failure detection time is reduced, the time for which the service is unavailable or MTTR is reduced and consequently availability is increased as it is dependent on MTTR. The proposed approach focuses on monitoring failures of individual services and not composite services. In service compositions, wrong execution order of services or failure of one service in a service composition can reduce service availability of all services in a service composition but mostly the focus of service compositions is on a single solution and how services are invoked in that solution. In our opinion, the probability of reuse of individual services is higher than the reuse of service compositions. A single service can be reused in many different business solutions so availability of individual services can be more beneficial as it can increase service availability in different solutions.

The current solutions for increasing availability include machine virtualization [17], clusters [4] and group communication systems [18]. The emergence of machine virtualization has significantly reduced system setup time, coupled with the ability to migrate services and the flexibility to consolidate multiple underutilized servers into a smaller number of machines [19]. These solutions aim at reducing MTTR by using redundancy and failover mechanism. In any reliability work in general, a decrease in MTTR contributes a proportional reduction in unavailability [20]. In most of the high availability distributed systems, redundancy is used to increase availability and redundant servers appear to reduce MTTR [19]. Failover can be used to ensure availability [4]. Failover is realized by heartbeat detection and automatically takeover of functions [21].

Failover requires failure detection and service migration to a redundant service provider. For failure detection, heartbeat mechanism can be used and timeouts can also be used. For failure identification at application level keep-alive probing can be used and applications can set own timeouts [22]. Another common method to detect failures is the error prone approach of timeouts in order to overcome inaccurate failure detections in software [23]. Another possibility to introduce fault tolerance to applications are self checking mechanisms in the code. The application verifies that it is healthy on its own. Through such tactics, most failures can be detect accurately [23].

In the proposed approach, heartbeat mechanism is used for failure detection and for better accuracy in comparison to the timeout approach. Monitoring service constantly checks for service failures and in timeout approach failure is only detected when service consumer sends a request to the service provider. As monitoring is done on a constant basis failures can be detected earlier than the timeout approach where the process of failure detection begins after consumer sends a service request. In case of monitoring, failed service can be restored using failover before a service consumer sends a request. In failover process, after failure detection, recovery is done by switchover process in which a redundant service provider takes over the work of failed service provider. Runtime monitoring can be used for analyzing and recovering from detected faults [24]. Monitoring is used in the proposed solution for failure detection and recovery.

III. PROPOSED MODEL

A. *Availability Parameters*

The tendency of SOA adoption raises the need for investigation of availability issues related to SOA. Nowadays, there is a tough competition in every field of life. Business systems and telecommunications systems need to increase customer satisfaction for acquiring higher profits. Safety critical systems need to provide more confidence to service consumers for getting more profits and in worst case for retaining profits. As these systems are using SOA, this can be achieved by improving quality of service attributes, such as by improving service availability in SOA.

This paper proposes a SOA-based model, which can be used by such systems for increasing service provider availability. Basic SOA model includes service providers, service consumers and service registry. In modified SOA model we have added a monitoring service to basic SOA model as shown in Figure 1. In Figure 1, service providers publish service descriptions (1.publish) in registry. Service consumers find (2.find) required service from registry. Service provider has service implementation and service registry holds service description. Service consumers bind (3.bind) to service provider. Service consumers and service provider start interacting (4.interact) with each other. Next section explains the role of monitoring service.

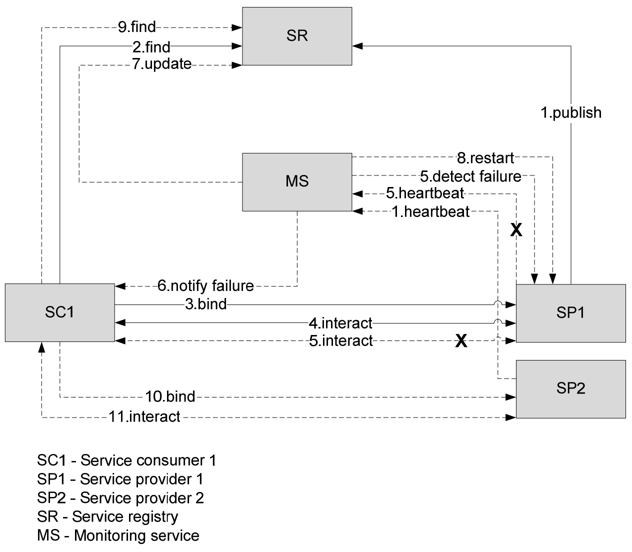

Figure 1. Modified SOA model.

Availability of service provider is essential as they provide functionality to service consumers. In general, availability decreases due to failures. Availability can be increased by increasing MTTF or by reducing MTTR. MTTF is the time until a failure happens and it can be increased by reducing failures from the system. For increasing MTTF statistical data is needed. MTTR is the time to repair the system. As the model proposed in the paper is based on a newly developed system, statistical data is not available for it and statistical data is not always accurate and complete as well. The focus in the proposed approach is on reducing MTTR by reducing failover time through monitoring for increasing service availability. Failover time is the downtime of service. In our opinion, by reducing failover time, availability can be increased.

The requirements for availability and high availability vary for different systems. High availability can be analyzed quantitatively as well as qualitatively. In the proposed solution, availability and high availability are qualitatively analyzed. Qualitative descriptions are used for analyses because SOA is not specific to an area of application and different areas of application can have different requirements for availability and high availability. Quantitative requirements differ for different areas of application whereas qualitative description is applicable in a generic way such as highly available systems should be able to detect failures and restore operations dynamically. Failures of some services can be tolerable and some are not depending on requirements and cost of high availability. Reliable communication is an essential service for many distributed applications, some of which require very fast recovery from failures, while others can tolerate slower failure recovery [25].

In our opinion, in SOA, several services work together to perform a business task and not all services used in a service-centric solution require high availability. For instance, customer interaction services may require high availability because their unavailability can result in monetary losses but services which are used for internal communication between employees may not need high availability as in this case monetary losses are not expected. In our opinion, high availability is expensive and it should not be added without considerable thought.

In SOA, service level agreements are used to describe quality of service parameters [26]. In our opinion, for SOA-based systems requirements of availability and high availability should be specified in SLA. Authors believe that highly available systems have certain features such as redundancy, use of automated means for recovery, minor failures and the ability of failure detection. A system can be considered highly available on the basis of these features. Redundancy, failure detection and automated recovery using monitoring are used for high availability in the proposed approach.

B. *Service Availability in SOA*

In the proposed solution, for increasing service provider availability in SOA, monitoring service is added to the basic SOA model as shown in Figure 1. Monitoring service detects failures, notifies service consumers about failures and initiates the failover process. Monitoring service deletes the information of failed service from service registry to reduce failures by reducing chances of requests being sent to failed services. Monitoring service restarts the failed service provider as well. UDDI based service registry is used in the test environment as it fulfils the requirements. In proposed solution, availability of service provider is improved by reducing failure detection time and failover time through monitoring. Failover time includes failure detection time and switchover time. Switchover time is the time that the primary and the backup are switching over the roles [12]. In the model, switchover time is the time for finding the backup service provider from service registry. In the proposed model, redundant service providers are used who send heartbeat messages to monitoring service at periodic intervals. Monitoring service detects failures on the basis of 3 consecutive missed heartbeat messages from the service provider.

In the proposed approach, heartbeat mechanism is used as failure detection time can be reduced with it and it provides greater accuracy in comparison to timeout approach. In case of heartbeat approach, failures can be detected earlier than timeout approach as failures are constantly monitored and service an be restored before a service consumer sends a request to service whereas in timeout approach service failure is detected only after the service consumer sends request to the service. In the approach of timeouts, service consumers are blocked for a certain time to get a response from service. If they do not get a response within a specified time interval, they retry the service and wait for ensuring service failure. In heartbeat mechanism service consumers are not blocked for a certain time and another service can be used as soon as the failure is detected.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

The following experiment is conducted to evaluate the proposed approach to increase service availability in service-oriented systems. Test environment for evaluating the proposed model is shown in Figure 2. WANem [27], an emulator is used for evaluating the model in test environment under different network conditions. Four nodes are used in the test setup. All traffic between service consumers, service providers and monitoring service passes through the emulator. Service consumers send requests in parallel to service providers and they are placed on one node. Service registry and service providers are placed on another node. Monitoring service is placed on a separate node. Service consumers and service providers are deployed on Glassfish server [28]. Service registry which is used is jUDDI [29] and it is deployed on Jakarta-Tomcat server [30]. Service registry uses mySQL [31] database for storing information. Synchronous web services are used in the implementation.

In experiments in Figure 3, different number of service consumers and different number of service providers are used with no packet loss or delay. In experiments shown in Figure 4, different number of service consumers and 10 service providers are used with different rates of packet loss 0 %, 1 % and 5 %. Different rates of packet loss are used for analyzing the impact of packet loss on service provider's availability. Packet loss is chosen to analyze the applicability of the proposed model to all kind of services as for some services such as VOIP services high rate of packet loss is expected whereas for other services rate of packet loss can be low. In the experiment, different heartbeat intervals are used to analyze optimal failure detection time and to reduce failover time to increase service provider availability. Model is emulated with 100 ms, 500 ms and 1000 ms heartbeat interval for sending heartbeats. Different heartbeat intervals are used to analyze how fast monitoring service can detect failures accurately. In the experiment, measurements are taken to see the impact of different redundancies and different number of service consumers on failover time.

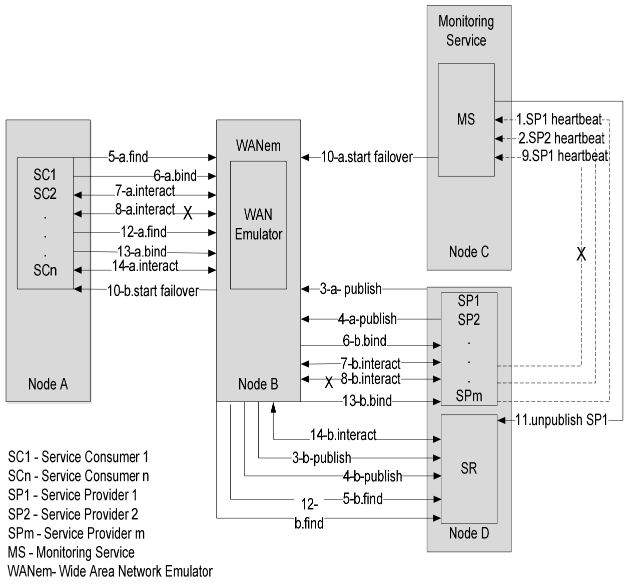

Figure 2. Test environment of modified SOA model.

Failover time is chosen in measurements as it is the time when service is unavailable. Different redundancies are used because overall performance can decrease or failover time can increase by adding more redundancy. In the proposed approach, performance is determined with respect to failover time. Failover time or performance should be acceptable to service consumers. Performance can also decrease with more service consumers as failover time can increase due to more number of service requests with higher number of service consumers.

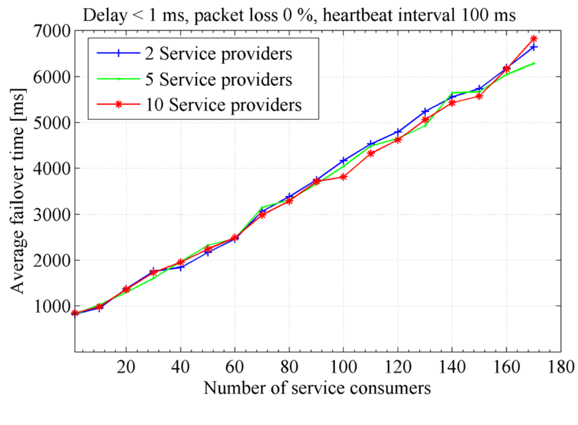

Figure 3. Average failover time with different redundancies.

Results shown in Figure 3, with 0 % packet loss show that by increasing number of service consumers, failover time is increased irrespective of redundancies. Results indicate that failover time with 130 service consumers is 5 s. In our opinion, failover time up to 5 s should be acceptable for most of the service consumers unless there are critical services which have higher requirements for failover time. In our opinion, for business systems and telecommunication systems 5 s failover time can be tolerable in most of the cases as loss of human lives is not expected with the services provided by business systems. Results indicate that service provider can tolerate failover requests from 130 service consumers at the same time which is quite acceptable according to the capacity of one system. However, by improving the capacity of system using better hardware the limitation of 130 service consumers can be removed. 5 s‘s failover time is high for critical services used by safety critical systems as they have high requirements for availability and high availability but for non-critical services they can also consider 5 s’s failover time.

Results indicate that by increasing redundancy and due to sending of more heartbeat messages, bandwidth consumption is not changed considerably, due to which performance or failover time stays similar with different redundancies. In our opinion, redundancy can be added in the system to increase availability of service according to requirement as performance is not deteriorated with it. We recommend that, for systems where frequency of failures is high or reputation damage is important for a certain business, redundancy can be added for increasing availability because results indicate that performance is not decreased with redundancy. However, if frequency of failures in a system is low, adding more redundancy to the system is not recommended as it will be expensive and it can result in underutilized resources.

Results in Figure 4, indicate that monitoring service can detect failures in less time when a small heartbeat interval is used such as 100 ms. Results show that with a small heartbeat interval, failure detection time is reduced. Failure detection time with 1000 ms heartbeat interval is 3000 ms, with 500 ms heartbeat interval failure detection time is 1500 ms whereas with 100 ms heartbeat interval failure detection time is reduced to 300 ms.

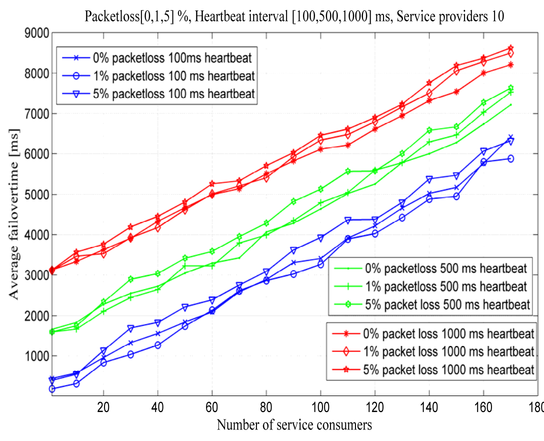

Figure 4. Average failover time with different rates of packet loss and with different redundancies.

Results in Figure 4 indicate that by selecting an appropriate heartbeat interval monitoring service can detect failures quickly which reduces failure detection time. As failure detection time is reduced, failover time is reduced as well and service availability is increased. The results indicate that service provider can handle 130 service consumers with 5 s’s failover time which can be acceptable for non-critical services and if the requirements of availability and high availability are not high for a specific system.

We have also analyzed the impact of packet loss on service provider availability in these measurements. Results in Figure 4, indicate that 1 % packet loss has insignificant impact on service provider availability and 5 % packet loss can reduce service provider availability, but the difference is not very high. The results indicate that services which can tolerate packet loss to some extent can be accommodated by using the model. Results indicate that sustainable work load can be identified by using the model. By avoiding peak load on the system, service provider availability can be increased.

V. CONCLUSION

Applicability of service-oriented architecture is increasing in safety critical systems, telecommunication systems and business systems. Requirements of availability and high availability for every application area are different. Even the best practices cannot be utilized properly to fulfil the requirements. The solutions for increasing availability, such as machine virtualization, clusters and group communications systems are not very much applied within the domains of SOA. In this paper, a SOA-based model has been proposed and monitoring is used for increasing service availability. Clusters, machine virtualization, group communication systems and middleware based solutions can increase availability but they can also increase complexity, performance overhead, installation requirements and maintenance costs due to which they are not chosen in the proposed approach. In the paper, it is investigated that how different network conditions can impact or reduce service availability. Proposed SOA model focuses on reducing failure detection time by using heartbeat mechanism. Heartbeat mechanism is chosen for more accuracy in failure detections in comparison to the timeouts approach. In case of timeouts, failure is detected once and with heartbeat mechanism failure is ensured repeatedly. Experimental results show the effectiveness of the approach and indicate that by using heartbeat mechanism failures can be detected earlier than the timeout approach. Results indicate that a small heartbeat interval can reduce failure detection time and failover time and by selecting an optimal heartbeat interval service availability can be increased. Availability is also increased by adding redundancy as a redundant system can cover more failures than a non-redundant system. Results indicate that redundancy does not reduce performance and it can be used according to requirement. The next step in the research work is to extend the model with redundant monitoring services as a single monitoring service can become a single point of failure for the system. Diverse monitoring services can be introduced in model to avoid failures of same kind. Model can be extended by analyzing availability of service compositions or by analyzing availability of asynchronous services. Also, a middleware can be added to the model and availability can be analyzed for middleware based service-oriented systems.

REFERENCES

[1] Reference Model for Service Oriented Architecture 1.0, OASIS Committee Specification 1, Aug 2006.

[2] OASIS, "Advancing open standards for information society", [retrieved: Feb, 2013] from http://www.oasis-open.org/.

[3] W. D. Yu and C. H. Ong, "A SOA-based Software Engineering Design Approach in Service Engineering," Proc. IEEE International Conf. on e-Business Engineering (ICEBE), China, Oct. 2009, pp. 409 - 416.

[4] B. Li, "Research and application of SOA standards in the integration on web services," Proc. 2nd International Workshop on Education Technology and Computer Science (ETCS), China, vol. 2, Mar. 2010, pp. 492–495.

[4] M. Chen, C. Wu, and k. Wu, "Staging adjust service pool to assure availability in SOA title," Proc. International Conf. on Complex, Intelligent, and Software Intensive Systems (CISIS), Korea, Jun. 2011,pp. 409-413.

[5] J. Qiu, J. A. Pershing, Y. Li, L. Xie, J. Luo, and Y. Chen, "Availability weak point analysis over an SOA deployment framework," IEEE Symposium on Network Operations and Management (NOMS), Brazil, Apr. 2008, pp. 473 - 480.

[6] J. A Pershing, L. Xie, J. Luo, Y. Li, and Y. Chen, "A methodology for analyzing availability weak points in SOA deployment frameworks," IEEE Transactions on Network And Service Management (TNSM), vol. 6, Mar. 2009, pp. 31-44.

[7] J. Niemöller, I. Fikouras, K. Vandikas, R. Levenshteyn, R. Quinet, and E. Freiter, "Blending the telecommunication domain with Web 2.0 services," Proc. 14th International Conf. on Intelligence in Next Generation Networks (ICIN), Berlin, Oct. 2010, pp. 1-6.

[8] J. He, E. Castro-Leon, and M. Chang, "Scaling down SOA to small businesses," Proc. IEEE International Conf. on Service-Oriented Computing and Applications (SOCA), Jun. 2007, pp. 99 - 106.

[9] J. Fenn, A. Brown, and C. Menon, "Issues and considerations for a modular safety certification approach in a service-oriented architecture," Proc. 5th IET International Conf. on System Safety, United kingdom, Oct. 2010, pp.1-6.

[10] M. Haberkorn, and K. Trivedi, "Availability monitor for a software based system," Proc. 10th IEEE High Assurance Systems Engineering Symposium (HASE), USA, Nov. 2007, pp. 321–328.

[11] A. Apon and L. Wilbur,"Ampnet - a highly available cluster interconnection network," 17th International Symposium on Parallel and Distributed Processing (IPDPS), France, Apr. 2003, pp. 201.2.

[12] M. Yin, "Assessing availability impact caused by switchover in database failover," Proc. Annual Reliability and Maintainability Symposium (RAMS), USA, Jan. 2009, pp. 401 - 406.

[13] T. G. Papaioannou, N. Bonvin, and K. Aberer, "An economic approach for scalable and highly-available distributed applications," Proc. IEEE 3rd International Conf. on Cloud Computing, USA, Jul. 2010, pp. 498 - 505.

[14] R. Levenshteyn, R. Quinet, J. Niemöller, I. Fikouras, K. Vandikas, and E. Freiter, "Blending the telecommunication domain with web 2.0 services," Proc. 14th International Conf. on Intelligence in Next Generation Networks (ICIN), Berlin, Oct. 2010, pp.1-6.

[15] S. Ahlfeld, S. Schulte, J. Eckert, A. Papageorgiou, T. Krop, and R. Steinmetz, "Enhancing availability with self-organization extensions in a SOA platform," Proc. 5th International Conf. on Internet and Web Applications and Services (ICIW), Barcelona, May. 2010, pp. 161 - 166.

[16] A. Grzech and P. Swiatek, "Complex services availability in service oriented systems," Proc. 21st International Conf. on Systems Engineering (ICSEng), USA, Aug. 2011, pp. 227 - 232.

[17] C. Lin, X. Zhang, and X. Kong, "Model-driven dependability analysis of virtualization systems," Proc. 8th IEEE/ACIS International Conf. on Computer and Information Science, China, Jun. 2009, pp. 199 - 204.

[18] Y. Krasny, A. Krits, E. Farchi, G. Kliot, and R. Vitenberg, "Effective testing and debugging techniques for a Group Communication System," Proc. International Conf. on Dependable Systems and Networks (DSN), Japan, Jun. 2005, pp. 80-85.

[19] H. Y. Chan, B. Y. Ooi, and Y. Cheah, "Dynamic service placement and redundancy to ensure service availability during resource failures," Proc. International Symposium in Information Technology (ITSim), Kuala Lumpur, Jun. 2010, pp. 715 - 720.

[20] W. D. Grover and A. Sack, "High availability survivable networks: When is reducing MTTR better than adding protection capacity?," Proc. 6th International Workshop on Design and Reliable Communication Networks (DRCN), France, Oct. 2007, pp. 1-7.

[21] L.Yue, Y. Shengsheng, G. Hui, and Z. Jingli, "Design of a dual-computer cluster system and availability evaluation," Proc. IEEE International Conf. on Networking, Sensing and Control (ICNSC), Taiwan, Mar. 2004, pp. 355 - 360.

[22] R. Wolski, J. S. Plank, and M. Allen, "The effect of timeout prediction and selection on wide area collective operations," Proc. IEEE International Symposium on Network Computing and Applications (NCA), USA, Oct. 2001,pp. 320 – 329 .

[23] K. P. Birman, "Reliable Distributed Systems Technologies, Web Services and Applications," Springer, 2005.

[24] A. Q. Gates, N. Delgado, and S. Roach, "A taxonomy and catalog of runtime software-fault monitoring tools," Proc. IEEE Transactions on Software Engineering, vol. 30, Dec. 2004, pp. 859 - 872.

[25] S. Han and K. G. Shin, "Experimental evaluation of failure-detection schemes in real-time communication networks," Proc. 27th Annual International Symposium on Fault-Tolerant Computing, USA, Jun 1997, pp. 122 - 131.

[26] P. Merson, L. O'Brien Lero, and L. Bass, "Quality attributes for service-oriented architectures," Proc. Int. Workshop Systems Development in SOA Environments (SDSOA), USA, May 2007, p. 3.

[27] WANem, "Wide Area Network Emulator," [retrieved: Feb 14, 2013] from http://wanem.sourceforge.net/.

[28] Glassfish homepage, [retrieved: Feb 16, 2013] from http://glassfish.java.net/.

[29] jUDDI, "An Apache Project Homepage," [retrieved: Feb 11, 2013] from http://juddi.apache.org/.

[30] Apache Tomcat, [retrieved: Feb 14, 2013] from http://tomcat.apache.org/.

[31] MySQL, [retrieved: Feb 22, 2013] from http://www.mysql.com/.