SOA and EDA: A Comparative Study
Similarities, Differences and Conceptual Guidelines on their usage

Zaharah Allah Bukhsh, Marten van Sinderen, P. M. Singh
University of Twente, Drienerlolaan 5, 7522 NB Enschede, the Netherlands
z.allahbukhsh@student.utwente.nl,{m.j.vansinderen,p.m.singh}@utwente.nl

Keywords:
Service oriented architecture, SOA, Event driven architecture, EDA, Event driven SOA, SOA 2.0, Learning Management System, LMS

Abstract:
Changing business requirements and new technologies trigger the business stakeholders to shift their approach from many small isolated systems to a single connected system. Integration of isolated systems is partially supported by service oriented architecture (SOA) and event driven architecture (EDA), each of which provides a set of system design guidelines. Since the purpose of both architectures is similar, the stakeholders have to make a choice on which architecture to use. The objective of this paper is to investigate the differences between SOA and EDA and provide conceptual guidelines on which architecture to consider for a given set of requirements. Apart from literature, we have considered various online resources (blogs, forums) that argue about differences and similarities between SOA and EDA. To clarify the design principles of both architectures, we present a case study of a learning management system (LMS).

1 INTRODUCTION

Emergence of web services has brought the service oriented architecture (SOA) into limelight (Natis, 2003). Features of loose coupling, flexibility, time to market, distributed nature and reusability of legacy applications gave the SOA a competitive advantage over the object oriented paradigms (He, 2003). The primary focus of SOA is to enable exploitation of services by accessing to remote components' interfaces through request and response methods (e.g. RPC). The concept of SOA is similar to the traditional client-server architecture of systems. But with changing the way of doing business, an IT system is not only required to be reactive but also proactive. Shortly after SOA, another architecture, known as event driven architecture (EDA), emerged which is able to sense and respond to real-time events. EDA is able to detect situations based on monitoring events and react to these situations (i.e. be proactive). From the business perspective, SOA mimics the business function (e.g. stock management) and EDA accommodates the real time business events (e.g. stock is low).

There have been discussions about the similarities and differences between SOA and EDA (Cramon, 2013; Dubray, 2014; van Hoof, 2006b).Moreover, there is an abundance of literature which emphasizes the relationship between SOA and EDA such as the combining of SOA and EDA features, and the interaction of such features (Maréchaux, 2006; Malekzadeh and Pessi, 2010; Zagarese et al., 2013). With the on-going discussions, Oracle proposed the combination of SOA and EDA under the title of SOA 2.0 (Krill, 2006).The terminology and idea of SOA 2.0 was highly criticized(McKendrick, 2006; Little, 2006), but, at the same time it suggest the event-driven approach to business in order to capture the real-time business events.

With many architecture patterns and varying business requirements, business architects and IT professionals need to make a decision on which architecture to consider for the design of a specific system. So, the motivation of this work is to facilitate the business stakeholders in their architectural choices while keeping the requirements of the system in mind. The contribution of this work lies in its role to clarify the major differences between SOA and EDA. It will also provide the the conceptual guidelines to business stakeholders for the choice of a particular architecture. We have chosen a case study of a learning management system (LMS), for the discussion of design principles of SOA and EDA. The reason for considering the

LMS case study is twofold: on one hand, it is easy to consider the LMS's design problems even from an academic setting and, on the other hand, a relatively large audience (students and researchers) has a basic understanding of e-learning environments.

This paper is structured as follows: section 2 presents the basic background knowledge of SOA and EDA along with sample scenarios. Major differences between SOA and EDA are discussed in section 3. Section 4 outlines the design problems of a learning management system and their solution using the SOA and EDA architectural approach. Section 5 provides the set of guidelines for stakeholders in making architectural choices. Finally, section 6 present our conclusion.

2 BACKGROUND

In this section, we present background of SOA and EDA. The brief introduction of SOA and EDA is provided in section 2.1 and 2.2 respectively. Section 2.3 presents sample scenarios on the use of SOA and EDA.

2.1 Service Oriented Architecture

SOA is a way of designing software system where independent software components provide services to end-users or other software components(Papazoglou, 2003). A service can be defined as a unit of functionality that is self-contained, discoverable and can be dynamically invoked(Bianco et al., 2007). SOA has three basic participants: service consumer, services provider and service registry. The interaction among these participants involve publish, find and bind operations(Champion et al., 2002). Service provider defines the service description and publishes it to a service registry. Using the find operation, service requester discovers the service description on service registry. Finally, the service requester invoke the service from service provider using the bind operation.

2.2 Event Driven Architecture

EDA is an architecture design where services of independent software components communicate through event notifications(Woolf, 2006).Maréchaux (2006) defines EDA as "a methodology for designing and implementing applications and systems in which events transmit between decoupled software components and services". EDA uses publish-subscribe architecture to enable the communication among services and to end-users. It has three main components: event emitter, event broker and event subscriber. An event emitter detects events and posts an announcement to the event broker. The event broker collects all the triggered events and forwards them to interested subscribers. Finally, event subscribers receive event notifications and respond accordingly. Event subscribers are able to further trigger the event to other services.

2.3 Sample Scenarios

To elaborate SOA and EDA, we are considering a simple example of an 'authentication process'. This process verifies the login credentials of users in order to ensure that right person get access to computer system.

In Figure 1, we show the 'authentication process' scenario from the SOA respective, where a client, which is a service consumer, requests a login service to get access to the system through the service interface. In response, the login service, which is a service provider, can invoke other services through service interfaces in order to send a random code to an e-mail/mobile for further verification. The service consumer is bounded to wait for and act on the response/reply from the service provider before access is possible. Thus, even though, the service consumer and service provider do not have interdependencies yet both are bounded during a communication.

Figure 2 presents the 'authentication process' from the EDA perspective. We have modified the login scenario in order to demonstrate the event-based communication among services. The client request the login service by the providing login credentials(1). In case of incorrect credentials, the login service will trigger the event to security service(2) through the enterprise service bus (ESB). In addition to providing the connection among services, ESB is also able to function as an event broker. Event broker analyse the triggered event and forward

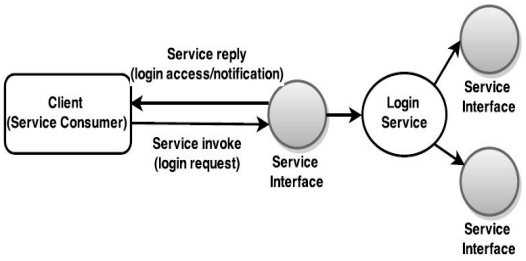

Figure 1: Service Oriented Architecture of 'Authentication Process'

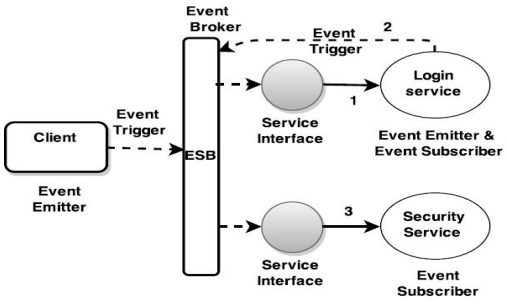

Figure 2: Event driven Architecture of 'Authentication Process'

it to security service(3). In this scenario, login service is event emitter while the security service is event subscriber. On the other hand, the security service (i.e. event subscriber) might take further security action by temporary blocking the account. In this scenario, the event emitter (login service) is not bounded to listen to the event subscriber's action/response(security service). Moreover, the role of event emitter and event subscriber are not mutually exclusive. A service(e.g. login service) can be event subscriber as well as event emitter at the same time.

From high level of abstraction, it can be said that the pattern to invoke the services in SOA and EDA are different as in SOA user command and other services can invoke the service while in EDA the service can be invoked with real-time event.

3 COMPARATIVE STUDY OF SOA AND EDA

In literature, we found two divergent points of views about SOA and EDA. According to one, SOA and EDA complements each other, while according to other, SOA and EDA are inverse of each other. In section 3.1 and 3.2, we have highlighted some of the basic commonalities and differences between SOA and EDA.

3.1 SOA and EDA :View from Literature

According to van Hoof (2008), EDA differs from SOA in its focus. SOA has services at the centre of its model while EDA has real-time events. SOA and EDA also differ in their communication style, where SOA approach is more focused on synchronous communication while EDA is focused on asynchronous communication. He summarise this difference as "both styles focus on the same architecture but from different viewpoints"(van Hoof, 2007a). According to him, request-and-reply pattern and publish-and-subscribe pattern, are inverse of each other(van Hoof, 2006b). However, he also agrees that the fusion of EDA and SOA will enhance loose coupling and bring agility to business. Debnath, vice president of Oracle server technologies, names the EDA as the known cousin of SOA (Rich, 2006). Contrary to van Hoof, Debnath acknowledges that SOA and EDA overlap at certain points but are very different at some. According to him, both architectures require an underlying infrastructure, a bus to carry requests in applications network, and some business processing rules. He notes that SOA and EDA are different only from the perspective of how a company wants to solve a problem. He refers to an analogy of the human body, where eyes and ears are similar to EDA as sensing the events and sending it to brain, while hands and feet are similar to SOA as providing movement on request of sense neurons.

Cramon (2013) considers the EDA as the solution to problems caused by SOA. According to him, SOA doesn't solve the integration problem, it lacks agility and the layered SOA is hard coupled. He proposes EDA as the solution to enhanced agility of business and solves the integration problems. According to him, events can drive the business processes. Dubray (2014) disagrees with Cramon on this point, according to him the asynchronous communication pattern doesn't work in the business world.

The explanation by Candy bridges the gap between two different points of views about SOA and EDA (Chandy, 2009). According to him, SOA and EDA are inverse of each other for those who approach them based on their communication pattern. While, SOA and EDA are complimentary for those who approach it from the structural point of view where both architecture enhance modularity and support tight-to-loose and loose to very loose coupling respectively.

3.2 SOA and EDA :Similar or Different

SOA and EDA has many features in common. Table 1. outlines the detailed differences between SOA, EDA and SOA 2.0/event driven SOA. From Table 1, it is worth noting that there doesn't exist much difference between EDA and SOA 2.0/event driven SOA. We believe a statement (in Table 1) with greater number of citations is more credible as compare to the one with lesser citations.

In addition to a difference in communication style, discussed in section 4.1, SOA and EDA are different in management of data. SOA do not allow the data replication at the functional level while multiple copies of data can be maintained at technical level

Category	SOA	EDA	SOA 2.0
Basic approach	Reactive approach: takes action on command (service pulling) (Malekzadeh and Pessi, 2010; Kong, 2013)	Proactive approach: detect events (which is change in state) and take action (events pushing) (Malekzadeh and Pessi, 2010; Kong, 2013)	SOA provide the design approach and infrastructure while EDA provide communication approach in SOA2.0. (Levina and Stantchev, 2009; Zicari, 2011; Rich, 2006)
Difference in SOA, EDA and SOA 2.0 features			
Business support	Services driven approach orchestrate the business functions and business processes (Sriraman and Radhakrishnan, 2005)	Event driven approach orchestrate the business events along with business processes (Sriraman and Radhakrishnan, 2005)	Based on specific scenarios, business events and business processes are dealt with SOA and/or EDA (Levina and Stantchev, 2009; Rich, 2006)
Level of coupling	Loosely coupled in technical domain but not in functional domain (Malekzadeh and Pessi, 2010; van Hoof, 2006a, 2007b; Juric, 2010)	Provide functional level decoupling (Malekzadeh and Pessi, 2010; Maréchaux, 2006; van Hoof, 2007b; Juric, 2010)	Provide loose coupling to decoupling due to events driven approach (Maréchaux, 2006; Levina and Stantchev, 2009; Hanson, 2005)
Data dependency	Data inconsistency is avoided through data isolation concept but it introduces data dependencies (van Hoof, 2007b; Dahan, 2009).	Data redundancy is employed to avoid dependencies and event provide the data synchronization function (van Hoof, 2007b; Dahan, 2009)	Violets the atomicity and consistency property of data (similar to EDA)(Dahan, 2009)
Business/IT alignment	Services become responsible for certain part of business domain.(Dahan, 2009)	Business events doesn't mimics real-time tasks of certain business domains and IT. (Dahan, 2009)	(similar to EDA)
Reusability	Loosely coupled services provide the reusability (van Hoof, 2006a)	Reusability is enhanced due to decoupled ends and fine-grinded services (Clark and Barn, 2012)	(similar to EDA)
Fault tolerance/availability	In case back-end system is down, processing of whole system is halted and user has to wait for response.(Dahan, 2009)	If the back-end system is down, user's request will still be accepted and responded due to data redundancy. (Dahan, 2009)	(similar to EDA)
Difference in Communication approach between SOA, EDA and SOA 2.0			
Communication style	Synchronous service invocation/remote procedural calls (request and response)(Woolf, 2006; van Hoof, 2007b; Juric, 2010; Luckham, 2007)	Asynchronous service invocation (Publish and subscribe) (Woolf, 2006; van Hoof, 2007b; Juric, 2010; Luckham, 2007)	Synchronous, Asynchro-nous communication between users and service is performed by events (Levina and Stantchev, 2009; van Hoof, 2007b; Luckham, 2007)
Interaction approach	Service need to be available when service consumer request it (Woolf, 2006; van Hoof, 2007b)	Event's subscriber doesn't need to be available when event is triggered (Woolf, 2006; van Hoof, 2007b)	Service doesn't need to be available (Similar to EDA)(Hanson, 2005)
Invocation approach	One service consumer can initiate one service at a time (Maréchaux, 2006; Yuan and Lu, 2009)	Event can trigger many subscribers at a time.(Maréchaux, 2006; Yuan and Lu, 2009)	An event can trigger many services at a time (Maréchaux, 2006; Hanson, 2005) (similar to EDA)
Interaction pattern (from requester)	Service consumer request specific service and wait for response (Woolf, 2006; van Hoof, 2007b; Luckham, 2007)	Event emitter triggers the event and doesn't wait for response. Emitter doesn't have knowledge on who are its subscribers(Woolf, 2006; van Hoof, 2007b; Luckham, 2007)	(Similar to EDA)
Interaction pattern (from provider)	Service provider response back with services/notification(van Hoof, 2007b; Luckham, 2007)	Event subscriber takes action but event emitter is not necessarily aware of it(van Hoof, 2007b; Luckham, 2007)	(Similar to EDA)

Table 1: Difference between SOA, EDA and event-driven SOA/SOA 2.0

(van Hoof, 2006a). On the other hand, EDA proposes to maintain certain level of data replication and data inconsistency, at functional level, which makes the adoption of this architecture a controversial debate(Dahan, 2009). With the expense of data inconsistency, data replication reduce the dependencies among services, and ultimately bring loose coupling in services network. Moreover, by managing the multiple copies of data, the system availability time and fault tolerance can also be improved. However, in case of system failure, the data cleaning is a challenging task as data can be inconsistent at times.

Besides differences, there are many similarities between SOA and EDA. The integral parts of both architectures are services. Both architectures aim to provide agility to business. The basic SOA design principles (service level agreement, service discoverability, service atomicity, service abstraction, service composability and service statelessness) are also similar for EDA. The connection between SOA and EDA can be better understood as the EDA being the upper layer of SOA, where the later invokes the services through events instead of commands.

4 CASE STUDY

This section provides the discussion of SOA and EDA design principles for a learning management system (LMS). In section 4.1, the introduction to LMS and its design problems are given. The discussion of the solution to the design problems of LMS from SOA and EDA perspective is presented in section 4.2 and 4.3 respectively. Finally, section 4.4 provides our insights on which architecture to choose over the other.

4.1 Learning Management System

LMS is a web-based application that provides the platform for interaction between students and instructors. These days, almost every educational institute uses a LMS to facilitate the communication among students and instructor. Most popular LMSs are Edmodo, Moodle and Blackboard (Dunn, 2012). However, sometimes the features provided by LMS are not adequate which leads to underutilization of LMS or not using it at all.

Major limitations of many LMSs include unavailability of required features, limited customization, and interoperability issues (Fathema and Sutton, 2013; Bickford, 2013). The changing requirements from instructors and students cause a mismatch of provided features of LMS and required features of the LMS. Similarly, few LMSs provide very limited customization ability to end-users e.g. browsers' provided file viewer, inability to add deadlines to calendar. To overcome such limitations, an educational institute has to use number of different systems for various purposes, e.g. along with the LMS, a calendar management tool, result report generation tools and other systems could be required. However, it must be possible for these system to be interoperable with LMS. We can define interoperability as the ability of system to talk to other systems. Most of the vendor's LMSs provide zero interoperability and integration for other systems at the learner's side(Forment et al., 2009).

Based on these design problems, the e-learning consortium proposed certain standards(Masie, 2002) for the development of LMSs. These standards are *interoperability, re-usability, manageability* and *accessibility*. According to these standards, a LMS should be able to connect with relevant other systems, its components should be reusable, the content and activities of a learner should be tracked, and the system should be accessible from anywhere, anytime and from any device.

4.2 SOA guidelines to overcome LMS design problems

In this section, we discuss on how SOA supports the standards proposed by e-learning consortium (Masie, 2002). Instead of single black box which is providing all the functionalities to clients, SOA proposes a layered approach which divides the whole infrastructure into layers. Based on the SOA reference architecture (Arsanjani et al., 2007), figure 3 shows a partial LMS architecture. A simple request-reply architecture between the presentation layer and the services layer is presented.

Using SOA, other systems, independent of platforms and languages, can connect to the LMS by following the description of the service interfaces. Figure 3 highlights the concept of provided interfaces from services and required interfaces from the presentation layer to enable communication. Any service interface that complies to the standard (e.g. WSDL) is discoverable by services and other systems. A change in implementation of service doesn't affect the service interfaces, which makes communication among clients, LMS and other systems transparent. The problem of *interoperability* is reduced through the services and standard service interfaces of SOA. The independent nature of services enhances the *reusability* of the system. Each service is responsible to provide certain functionalities of the LMS to the

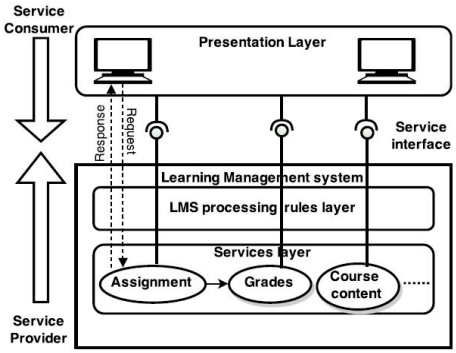

Figure 3: LMS model based on SOA reference architecture

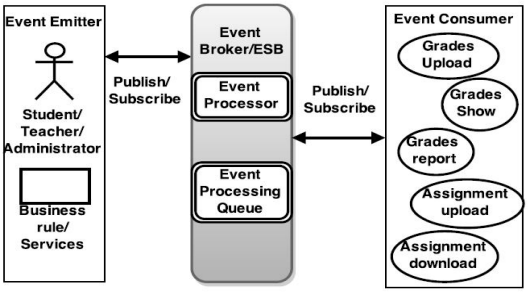

Figure 4: LMS model based on EDA reference architecture

users, e.g. a 'Grades' service is responsible to upload, save, maintain and show the grades of registered users. Services are supported by the underlying lower level layer. Since, our discussion is limited to design issues of services we have omitted lower level layers from figure 3. Independent services and standard service interfaces reduce the complexities of interoperability, integration and collaboration posed by traditional systems. However, interoperability aspect of SOA-based systems is sometimes challenged, due to coupled request-reply architecture(Taylor et al., 2009).

The functionality of the LMS is visible only through the services. The activities of students and instructors on LMS are easy to track with services invocation calls, which makes the system more transparent and *manageable*. Since the services infrastructure is not tightly coupled, *accessibility* of services are much enhanced as compared to procedures based object oriented scenarios where procedures are dependent on each other.

4.3 EDA guidelines to overcome LMS design problems

In this section, we discuss how EDA support the standards proposed by the e-learning consortium(Masie, 2002). In Figure 4, we have presented a partial LMS architecture based on EDA reference architecture (Moxey et al., 2010).

EDA takes the SOA layered approach one step further by introducing the publish-subscribe architecture, which brings end-to-end decoupling among services. The integral part of the system is the service layer. However, EDA enables the decoupling on the services ends(Yuan and Lu, 2009). The decoupling of services improves the *reusability* of the system. In EDA, the services are meant to be fine-grained. We illustrate this (in Figure 4) by showing the 'Grades Upload' and 'Grades Show' as separate services compared to the 'Grades' service represented in Figure 3. The Grades Upload services can trigger the Grade Show service by publishing an event of 'grades available'. Fine-grained services enhance the reusability of system but on the other hand it also increases the network load because a client may need to trigger an event number of times in order to get the complete data. The decoupled services and asynchronous communication pattern of EDA improves the *interoperability* of LMS. Moreover, fine-grained services enable the LMS to deal with the changing requirements of end-users. The event-driven processing characteristic of EDA allows the system to be easily *manageable* as all the triggered events by users/services can be tracked and responded through the event processor and event processing queue. Due to the decoupled services, an EDA implementation can improve the availability and *accessibility* of the system. For instance, even if the back-end repository of the LMS is down due to network load, the Grades Upload service can still upload the grades and provide the response message (Arsanjani et al., 2007).

4.4 Which architecture to choose?

Both architectures, SOA and EDA, are able to solve the design problems of LMS. However, the solutions provided by both architecture has some limitations. Standard service interface of SOA bring the interoperability aspect to LMS but has a tight communication architecture. On the other hand, EDA enhance the accessibility and reusebility by decoupling but it is unable to respond to client's awaiting request efficiently.

The combinational approach of SOA and EDA, can be called SOA 2.0, will enable a LMS to sense the real-time events as well as deal with client's awaiting requests. SOA can provide the service infrastructure with standard service interface while the EDA is able to build a smart and self-aware LMS. The resulting LMS will have enhanced system interoperability and accessibility for clients while at the same time can sense the real-time events. (e.g. LMS can sense

the system failure and can automatically initiates the actions to resolve them).

5 DISCUSSION

The basic purpose of SOA and EDA is to bring the agility to business systems in order to deal with changing day-to-day scenarios. The infrastructure of both architectures is based on services with difference in service invocation method. The main differentiation point is request-reply architecture of SOA and the events based publish-subscribe architecture of EDA. Based on literature study, we can conclude that SOA and EDA have different capabilities. In the following, we will provide some conceptual guidelines, derived from literature study, on which architecture approach to adapt considering the system's requirements.

SOA is a architectural solution if a (a) system is of transactional nature, (b) high data integrity need to be maintained, (c) client's awaiting requests need to be handled in timely manner, and (d) strong cohesion is required in services implementation. By strong cohesion of services, we mean a service should be able to provide the functionality of certain business domain. Example of SOA based system is banking system where business operations are set of atomic transactions (e.g. deposit, withdraw) which are required to maintain data integrity all the time.

With different capabilities than SOA, EDA is suitable for those (a) systems which is of analytical nature, (b) automated execution of task is required based on conditions, (c) client requests can be entertained later in time, and (d) decoupling is required in services implementation. Decoupling enhance the system's ability to sense and respond to events. Example of EDA based system is inventory manager where a business event must be sensed and responded in case of stock is low, order accepted is greater than available stock, etc.

Except their difference in service invocation method, SOA and EDA complement each other(Maréchaux, 2006; Dahan, 2009; Rich, 2006). The varying design principles of both architectures are able to provide a system that is enriched in functionality. SOA provide the EDA with distributed setting of services which makes the sensing and responding of events possible. On the other hand, EDA replace the hard-coupled remote procedural calls with flexible sense-and-publish architecture. This combinational approach of SOA and EDA is collectively referred as event-driven SOA or SOA 2.0. Event-driven SOA is able to sense the analytical events and can also deal with transactional requests.

6 CONCLUSION

SOA and EDA are architecture designs which assist in system implementation. Both of these architectures are different in their communication pattern yet maintain similarities in basic service infrastructure. Requirements of today's business demand a system that is smart and self-aware in dealing with real-time situations and, at the same time, can manage customers' requests. With these requirements, events and services are both needed. SOA has to borrow the event-driven approach from EDA, and EDA has to base on SOA services and standard interfaces.

Our research contributes to the discussion on the similarities and differences between SOA and EDA. This discussion serves a higher purpose, namely to be able to decide which (combination of) architectural patterns is best to fulfil given requirements. We provided guidelines to help making such decisions. We have concluded that SOA and EDA are different in their communication styles and in some service design principles but these differences doesn't make these architecture mutually exclusive. As a future direction, further comparative studies based on the practical implementation of SOA, EDA and their combination can bring more insights into their capabilities, differences and benefits.

REFERENCES

Arsanjani, A., Zhang, L.-J., Ellis, M., Allam, A., and Channabasavaiah, K. (2007). Design an SOA solution using a reference architecture. *IBM DeveloperWorks*.

Bianco, P., Kotermanski, R., and Merson, P. F. (2007). Evaluating a service-oriented architecture.

Bickford, A. (2013). 12 common complaints about learning management systems (LMS). [Online] Available at: http://bit.ly/N6C2k3 [Accessed: 10th Nov 2014].

Champion, M., Ferris, C., Newcomer, E., and Orchard, D. (2002). Web services architecture. *W3C working draft*, 14.

Chandy, M. (2009). One day SOA and EDA will be used in all aspects of daily life: Dr. K. Mani Chandy explains. [Online] Available at: http://bit.ly/1Du2uay [Accessed: 23 Oct 2014].

Clark, T. and Barn, B. S. (2012). A common basis for modelling service-oriented and event-driven architecture. In *Proceedings of the 5th India Software Engineering Conference, ISEC'12*, pages 23–32.

Cramon, J. (2013). SOA and event driven architecture (SOA 2.0). [Online] Available at: http://slidesha.re/1MrLNEf [Accessed: 10 Oct 2014].

Dahan, U. (2009). EDA: SOA through the looking glass. [Online] Available at: http://bit.ly/1vQqwiL [Accessed: 10 Nov 2014].

Dubray, J.-J. (2014). SOA vs EDA. [Online] Available at: http://slidesha.re/1FjQU7u [Accessed: 15 Oct 2014].

Dunn, J. (2012). The 20 best learning management systems. [Online] Available at: http://bit.ly/1vzRnd9 [Accessed: 28 Nov 2014].

Fathema, N. and Sutton, K. L. (2013). Factors influencing faculty membersâĂŹ learning management systems adoption behavior: An analysis using the technology acceptance model. *International Journal of Trends in Economics, Management and Technology, USA*.

Forment, M., Guerrero, M., GonzÃąlez, M., PeÃśalvo, F., and Severance, C. (2009). Interoperability for LMS: The missing piece to become the common place for elearning innovation. In *Visioning and Engineering the Knowledge Society. A Web Science Perspective*, volume 5736, pages 286–295. Springer Berlin Heidelberg.

Hanson, J. (2005). Event-driven services in SOA.

He, H. (2003). What is service-oriented architecture. *PublicaÃğÃčo eletrÃŧnica em*, 30:50.

Juric, M. B. (2010). WSDL and BPEL extensions for event driven architecture. *Information and Software Technology*, 52(10):1023–1043.

Kong, X. (2013). A financial services case study of SOA based on CEP. *Journal of Theoretical and Applied Information Technology*, 48(1):595–599.

Krill, P. (2006). Make way for SOA 2.0. [Online] Available at: http://bit.ly/1zIZCEb [Accessed: 5 Nov 2014].

Levina, O. and Stantchev, V. (2009). Realizing event-driven SOA. *ICIW*, 9:37–42.

Little, M. (2006). SOA 2.0 ignorance. [Online] Available at: http://bit.ly/1EFhDaH [Accessed: 7 Oct 2014].

Luckham, D. (2007). SOA, EDA, BPM and CEP are all complementary. [Online] Available at: http://bit.ly/1AKtTbi [Accessed: 7 Nov 2014].

Malekzadeh, B. and Pessi, K. (2010). Event-driven architecture and SOA in collaboration-a study of how event-driven architecture (EDA) interacts and functions within service-oriented architecture (SOA). Master's thesis, University of Gothenburg.

Maréchaux, J.-L. (2006). Combining service-oriented architecture and event-driven architecture using an enterprise service bus. *IBM Developer Works*, pages 1269–1275.

Masie, E. (2002). Making sense of learning specifications & standards: A decision makerâĂŹs guide to their adoption. *The Masie Center, evaluation*.

McKendrick, J. (2006). Please, no SOA 2.0. [Online] Available at: http://zd.net 17IJvAQ [Accessed: 7 Oct 2014].

Moxey, C., Edwards, M., Etzion, O., Ibrahim, M., Iyer, S., Lalanne, H., Monze, M., Peters, M., Rabinovich, Y., and Sharon, G. (2010). A conceptual model for event processing systems. *IBM Redguide publication*.

Natis, Y. V. (2003). Service-oriented architecture scenario. Gartner Research, Stamford.

Papazoglou, M. P. (2003). Service-oriented computing: concepts, characteristics and directions. In *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, pages 3–12.

Rich, Seeley, N. W. (2006). Oracle's debnath on making an event-driven SOA. [Online] Available at: http://bit.ly/1Ei4moL [Accessed: 23 Oct 2014].

Sriraman, B. and Radhakrishnan, R. (2005). Event driven architecture augmenting service oriented architectures. *Report of Unisys and Sun Microsystems*.

Taylor, H., Yochem, A., Phillips, L., and Martinez, F. (2009). *Event-driven architecture: how SOA enables the real-time enterprise*. Pearson Education.

van Hoof, J. (2006a). How EDA extends SOA and why it is important. [Online] Available at: http://bit.ly/1L5UfY8 [Accessed: 15 Nov 2014].

van Hoof, J. (2006b). Why to distinguish between soa and eda. [Online] Available at: http://bit.ly/1DiFpfU [Accessed: 27 Oct 2014].

van Hoof, J. (2007a). The magical A of SOA and EDA. [Online] Available at: http://bit.ly/1L5TWfZ [Accessed: 7 Oct 2014].

van Hoof, J. (2007b). SOA and EDA: Using events to bridge decoupled service boundaries. *The SOA Magazine,(4)*.

van Hoof, J. (2008). EDA versus CEP (and SOA). [Online] Available at: http://bit.ly/1EFiOXH [Accessed: 16 Nov 2014].

Woolf, B. (2006). Event-Driven Architecture and Service-Oriented Architecture. [Online] Available at: http://ibm.co/1L5UdiQ [Accessed: 15 Nov 2014].

Yuan, S.-T. and Lu, M.-R. (2009). An value-centric event driven model and architecture: A case study of adaptive complement of SOA for distributed care service delivery. *Expert Systems with Applications*, 36(2):3671–3694.

Zagarese, Q., Furno, A., Canfora, G., and Zimeo, E. (2013). Towards effective event-driven soa in enterprise systems. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 1419–1424. IEEE.

Zicari, R. (2011). Advancing SOA with an event-driven architecture. [Online] Available at: http://bit.ly/17iGxSx [Accessed: 15 Nov 2014].