Oriental Journal of Computer Science & Technology Vol. **3**(1), 171-184 (2010)

SOA-RTDBS: A service oriented architecture (SOA) supporting real time database systems

OLUSEGUN FOLORUNSO[1]*, LATEEF O. YUSUF[1] and JULIUS O. OKESOLA[2]

[1]Department of Computer Science, University of Agriculture, Abeokuta, (Nigeria).
[2]Departments of Computer and Information Science, Tai Solarin University of Education, Ijebu-Ode, (Nigeria).

(Received: April 21, 2010; Accepted: June 18, 2010)

ABSTRACT

With the increase of complexity in Real-time Database Systems (RTDBS), the amount of data that needs to be managed has also increased. Adoption of a RTDBS as a tightly integrated part of the SOA development process can give significant benefits with respect to data management. However, the variability of data management requirements in different systems, and its heterogeneity may require a distinct database configuration. We addressed the challenges that face RTDB managers who intend to adopt RTDBS in SOA market; we also introduce a service oriented approach to RTDBS analytics and describe how this is used to measure and to monitor the security system. A SOA approach for generating RTDBS configurations suitable for resource-constrained real-time systems using Service Oriented Architecture tools to assist developers with design and analysis of services of developed or new systems was also explored.

Keywords: Service Oriented Architecture, Real time database systems.

INTRODUCTION

As data complexity is growing the need for a uniform, efficient, and persistent way to store data is becoming increasingly important. Using a Real-time Database System (RTDBS) as a tightly integrated part of SOA system has the potential to solve many of the problems that application designers have to consider with respect to data management, SOA system can reduce development costs, result in higher quality of the design of the systems, and consequently yield higher reliability (Mike et al., 2008). In this work, we propose SOA as a new approach to building RTDBS that allows businesses to leverage existing assets and easily enable the inevitable changes required to support the business. One of the most important aspects of SOA is that it is a *business*, a *technological* as well as methodological approach (Judith et al., 2007). SOA enables businesses to make business decisions *supported* by technology instead of making business decisions *determined* by or *constrained* by technology. And with SOA, the folks in RTDBMS finally get to say "yes" more often than they say "no." One of the biggest deals in the SOA world is the idea that things are not thrown away, the best of software assets used every day is packaged in a way that allow for use, reuse and keep on reusing it securely in the knowledge that future changes will be simple, straightforward, safe, and fast. This makes system less complicated and less expensive to maintain. Mike et al. (2008) described SOA as the careful balance and blending of the big picture and the immediate requirements to the practical application of theory to meet a set of goals in the present and in the future.

According to Krithi et al. (2004), a real-time system consists of a controlling system and a controlled system. In an automated factory for example, the controlled system is the factory floor with its robots, assembling stations, and the assembled parts, while the controlling system is the computer and human interfaces that manage and

coordinate the activities on the factory floor. Timely monitoring of the environment as well as timely processing of the sensed information is necessary. In addition to the timing constraints that arise from the need to continuously track the environment, timing correctness requirements in a real-time database system also arise because of the need to make data available to the controlling system for its decision making activities. Besides robotics, applications such as medical patient monitoring, programmed stock trading, commerce, electronic banking, telecommunications and military command and control systems like submarine contact tracking require timely actions as well as the ability to access and store complex data that reflects the state of the application's environment. That is, data in these applications must be valid, or fresh, when it is accessed in order that the application performs correctly. A RTDB is composed of real-time objects which are updated by periodic sensor transactions. An object in the database models in a real world entity, for example, the position of an aircraft. A real-time object is one whose state may become invalid with the passage of time. Associated with the state is a temporal validity interval. To monitor the states of objects faithfully, a real-time object must be refreshed by a sensor transaction before it becomes invalid, that is, before its temporal validity interval expires. The actual length of the temporal validity interval of a real-time object is application dependent.

It is worth mentioning the challenges faced by RTDBS managers in SOA markets. The most important challenges are the lack of knowledge about terms and aspects related to it (LaPlante, 2005). Most of the software engineering courses offered in our faculties is only based on traditional methodologies such as Object-Oriented Programming (OOP) with some related topics such as Unified modelling Language (UML). Object-Oriented Analysis and Design (OOAD), and Object Oriented Languages. To solve this problem, curriculums offered by our faculties should be extended to contain different aspects of RTDBS in SOA market including definition and type of services, approaches and strategies, enabler tools and technologies, the role that RTDBS in SOA market can play in integrating different systems and missing points in current SOA-RTDBS models. This will in no doubt play a great role in yielding a new generation of software engineering specialists that can meet new market needs.

As our society is becoming more integrated with SOA concepts, information processes through the client service consumer paradigm is becoming inevitable, responds to request by the service provider in real-time manner with predictable and timely behaviour is highly desirable, thus databases with the amalgamation of real-time systems and the concepts of SOA give birth to our new concept of SOA-RTDBS Interoperability. Interoperability of data and services means that data and services can be defined and used independently of the application, programming language, operating system, or computing platform which implements them. Interoperability has a long history in computing. Major phases include: Electronic Data Interchange, object models, virtual machines, and web services. We are motivated by the work of Joseph et al. (2008); they investigated and improved on the activities of visanet by enforcing interoperability through the SOA approach to improve transactions between member banks that own visa for transactions. They enforced error free, extremely high system reliability in a real time manner between member banks. They also ensured that all data in visanet and all processes that operated on data were always the same. This guaranteed uniformity in the system environment. A critical mass of widely-adopted technologies is available to implement and use a web services-based SOA, and more technologies, as well as tools are on the way. A service-oriented architecture for RTDBS is an information technology approach or strategy for RTDBS in which RTDB applications make use of (perhaps more accurately and in a real time manner) rely on Data-based services available in a network such as the World Wide Web. Implementing a service-oriented architecture can involve developing applications like RTDBS that use services, making RTDBS applications available as services so that other applications can use those services or both.

Background and Related Work

Database systems in which time validity intervals are associated with the data are discussed in Kuo and Mok (1993, 2000) and Song and Liu

(1995). Such systems introduce the need to maintain data temporal consistency in addition to logical consistency. The performance of several concurrency control algorithms for maintaining temporal consistency are studied in Song and Liu (1995). In the model introduced in Song and Liu (1995), a real-time system consists of periodic tasks which are either read-only, write-only or update (read-write) transactions. Data objects are temporally inconsistent when their ages or dispersions are greater than the absolute or relative thresholds allowed by the application. Studies of two-phase locking and optimistic concurrency control algorithms, as well as rate-monotonic and earliest deadline first scheduling algorithms show that the performances of the rate-monotonic and earliest deadline first algorithms are close when the load is low. At higher loads, earliest deadline first outperforms rate-monotonic when maintaining temporal consistency. They also observed that optimistic concurrency control is generally worse at maintaining temporal consistency of data than lock based concurrency control, even though the former allows more transactions to meet their deadlines. It is pointed out in Song and Liu (1995) that it is difficult to maintain the data and transaction time constraints due to the following reasons:

- A transient overload may cause transactions to miss their deadlines
- Data values may become out of date due to delayed updates
- Priority based scheduling can cause pre-emption which may cause the data read by the transactions to become temporally inconsistent by the time they are used.
- Traditional concurrency control ensures logical data consistency, but may cause temporal data inconsistency

Motivated by these problems, and taking into account the fact that transactions process data with validity constraints and such data will be refreshed with sensor transactions, the notion of data-deadline is introduced in Xiong et al. (2002). It is coupled with data-deadline and used to improve the performance of transaction scheduling. The notion of similarity is used to adjust transaction workload by Ho et al. (1997), and incorporated into embedded applications (e,g., process control) in Chen and Mok (1999).

Many real-time database applications are inherently distributed in nature. These include the intelligent network services database, telecom databases, mobile telecommunication systems and the 1-800 telephone service in the United State. More recent applications include the directory, data feed and electronic commerce services that have become available on the World Wide Web. The performance reliability and availability of such applications can be significantly enhanced through the replication of data on multiple sites of the distributed network.

An algorithm for maintaining consistency and improving the performance of replicated DRTDBS is proposed in Son (1987). In this algorithm, a multiversion technique is used to increase the degree of concurrency. Replication control algorithms that integrate real-time scheduling and replication control are proposed in Son and Kouloumbis (1993). In contrast to the relaxed correctness assumed by the latter, conventional one-copy serialzability support by the algorithm called MIRROR reported in Xiong et al. (2002).

Temporal consistency guarantees are also studied in distributed real-time systems. In Zhou and Jahanian (1998), Distance constrained scheduling is used to provide temporal consistency guarantees for real-time primary-backup replication service. In Harista et al. (2000), its authors propose and evaluate a commit protocol that is specifically designed for the real-time domain without these changes. PROMPT allows transactions to optimistically borrow in a controlled manner, the updated data of transactions currently in their commit phase. This controlled borrowing reduces the data inaccessibility and the priority inversion that is inherent in distributed real-time commit processing. A simulation-based evaluation shows PROMPT to be highly successful as compared to the classical commit in minimizing the number of missed transaction deadlines. In fact its performance is close to the best on-line performance that could be achieved using the SOA approach.

Nizar et al., 2008 described real-time databases as information with time-constrained data

and time-constrained transactions. They noted that real-time database design requires the introduction of new concepts to support both data structures and dynamic behaviour of the database. They also gave an overview about different aspects of real-time databases and clarified requirements of their modelling. A frame work for real-time databases was described and their fundamental operation was elaborated. Their study demonstrates the validity of the structural model and illustrates SQL queries and java code generated from the classes of the model.

According to Ed (2005), Electronic Data Interchange (EDI) introduced a standard syntax and a standard set of messages for various common business transactions. EDI began by introducing standard formats for purchase orders, invoices, and bills of lading so that these could be processed without human intervention. In other words, EDI approached interoperability by requiring all applications using purchase orders to use a standard structured format. Each different business document had its own structured format. With the introduction of Object Models, interfaces for both data and methods became formalized. A wide variety of object models have been introduced, including Microsoft's COM and DCOM, Sun Microsystems Java Beans and Enterprise Java Beans, and the Object Management Group's (OMG) Object and Component Models. OMG's Common Object Request Broker Architecture or CORBA is an architecture providing interoperability for objects across different languages, operating systems, and platforms (Aniruddha et al, 2003). Virtual Machines was popularized by Java, the idea of supporting interoperability by mapping a language to an intermediate format (byte code) which could be executed on any machine which had an interpreter for byte code (virtual machine). In this way, the same code could be executed on different operating systems and platforms. The limitation is that the same language (Java) must be used. More recently, web services and service oriented architectures have popularized the use of XML to define data and message formats. In particular the Web Services Description Language or WSDL provides a way to define services that can be bound to different programming languages (e.g. Java, Perl, C/C++) and protocols (http, smtp, etc.). Our approach to data interoperability is based upon a service oriented architecture that is specifically designed to support statistical and other analytical models, as well as various services employing them.

Services are at the core of SOA. A service in SOA is. Just like the definition of SOA, defining the term service is not an easy task (Perrey and Lycett, 2003). The simplest idea is that a service performs a (reusable) function. This function can be anything from simple retrieval of data to performing a whole business process (Papazoglou, 2003) and Papazoglou and van, 2007). The services in SOA always have a business aspect to them instead of a more technical aspect. An example of a service is get customer profile, opposed to upload file, which is not a business service (Krafzig et al., 2004). A service is defined by (OASIS, 2006) as "the performance of work (a function) by one for another". This definition is related to the following ideas:

- capability to perform work for another
- specification of the work offered for another
- offer to perform work for another

SOA-RTDBS Architecture and Methodology

Creating service oriented architecture for RTDB takes thought, patience, planning, and time. It is a journey, and depending on the size and scope of an organization, it may be a journey of years or even a decade. But we can start seeing returns on our RTDBS in SOA market investment very quickly, without having to rewrite all our software. Figure 1 is a simple software architecture related to RTDB

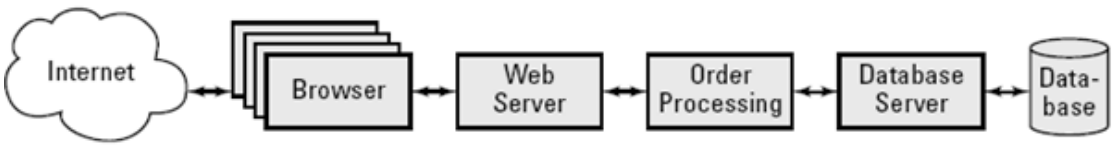

Fig. 1: A simple software architecture (Adapted from Service Oriented Architecture for Dummies by Judith et al., 2007).

This is how it works:

The Browser is a program located on a user's device (PC, laptop, PDA, or cell phone) that accesses the business application through a Web site. Many users can access the DB application or any other applications at the same time; so many browsers will typically link to the Web server. The primary job of the browser is to display information and accept input from the user.

The Web Server manages when and how the many Web pages are sent to the browsers of the users who access the business application. (Web servers may do other things as well, but we are concentrating on its primary service.)

The Order-Processing Application carries out the business process that is being executed, which in this case means carrying out the necessary steps to accept the order and fulfill the customer's request, if possible. This component embodies the company's business practices for interacting with customers.

The Database Server is computer software that reads data from a database in a real-time manner and sends the data where it is needed.

The Database is where the definitions of the business data and the data itself are stored.

Information passes from the browser to the Web server to the order-processing application, which decides what to do next. The order-processing application might pass data to the database server to write to disk, or it may request data from the database, or it may simply send information back to the browser through the Web server. What the order-processing application does depends upon the information and commands passed to it by the user via the browser.

The diagram in Figure 1 can also be referred to as a *business service* which means in simple terms, the wrappings up of everything we have to do to make a particular business function happen.

In Figure 2, a credit-checking component is added to our business diagram. Its service is called on when new customers place an order to determine whether they are credit worthy; this must be done in a real-time manner as not to frustrate the customer. In the figure, we don't show or even care about how the credit checking is done. For the sake of simplicity, it is assumed that the credit-checking software component is a database run by an external company and simply provides a service in a real time manner. The company using this credit-checking software is confident that the service conducts a credit check in the right way.

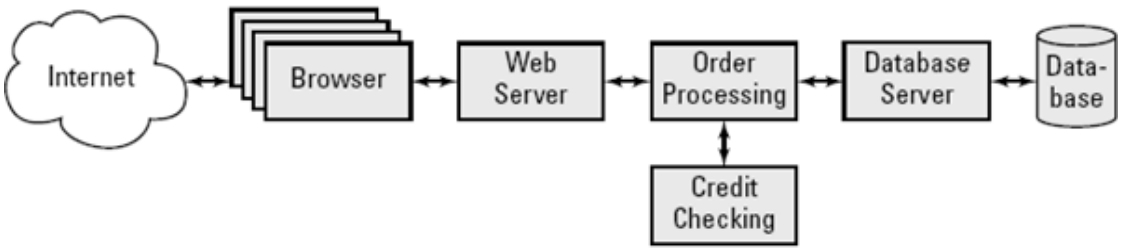

Fig. 2: Adding a service oriented component (Adapted from Service Oriented Architecture for Dummies by Judith et al., 2007).

The order-processing application simply requests the credit-checking service and passes along the necessary information (a person's name and Social Security number). The credit-checking component consults its information sources, does some calculations, and passes back a credit rating. The credit checking component may connect with many computers, consult many different data sources, and use a very sophisticated algorithm to calculate the credit rating, but this is of no concern to the order-processing application so far the information is received in a timely manner. As far

as the order-processing application is concerned, credit checking is just a *black box.* Also, we need to emphasize that the credit-checking component *does only credit checking*. It doesn't offer a wide range of services. It is precisely because the components have a narrowly defined scope — that is, they do "just one thing" — that they can be used and reused as building blocks. SOA's use and reuse of components makes it easier to build new applications as well as change existing applications. Using well-proven, tested components makes testing new applications more efficient.

Folorunso et al., (2008) described Real Time Database System (RTDBS) as the mainstream of computer operations which aptly described as a system that produces result in a timely and consistent fashion. They observed that the RTDBS designers are interested in seeing how their algorithms behaved, they proposed an extendable framework using Model-View-Controller paradigm which was adopted for our SOA architecture. In a real-time database system (RTDBS) transaction travel through various components until their termination. We present in Figure 3 how the database server uses the services of RTDB system model to respond to the client in a real time manner.

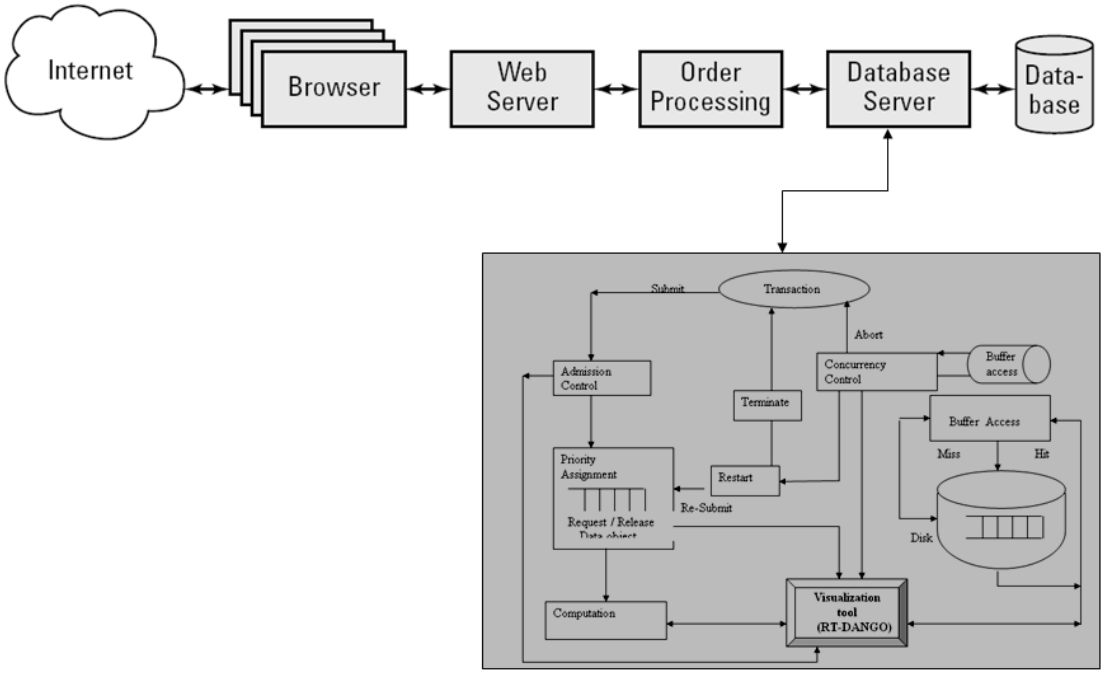

Fig. 3: Adding RTDBS with an embedded visualization output tool (RT-DANGO).

In Figure 4, we introduce the idea of a business layer and a plumbing layer, and in doing so, we introduce the idea of specific services. (For simplicity's sake, we've left out the Web server and the browser.) It works like this:

The Business Service Layer consists of our RTDBS tools and other software components which provide and carry out specific business functions. Another way to say this is that they deliver specific business services.

The Plumbing Layer consists of components that support the abovementioned business services by marshalling and managing actual computer resources. Here are two such components:

• *Presentation Service:* The Web server called by a different name, for example Internet Explorer, Mozilla Firefox, Netscape, safari etc.
• *Data Service:* The database server called by a different name.

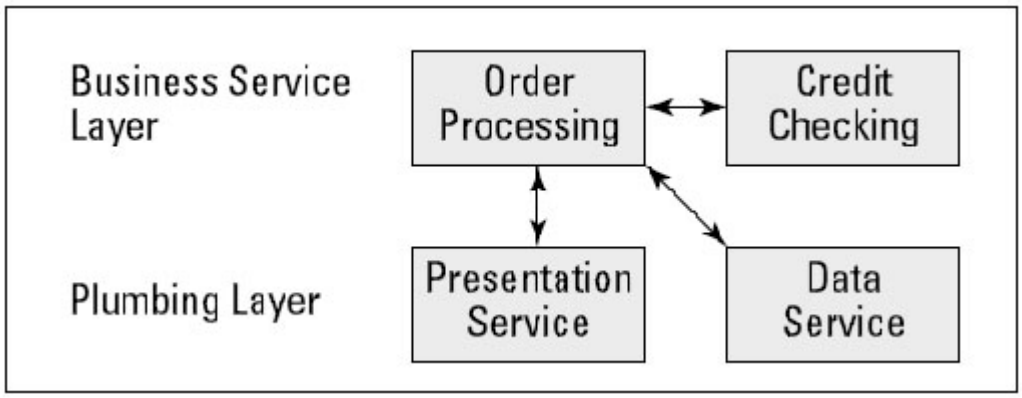

Fig. 4: A service oriented view
(Adapted from Service Oriented Architecture for Dummies by Judith et al., 2007).

By splitting the architecture diagram into two layers, we divide the software that is of direct relevance to the business — because it carries out business

In Figure 5 we include in our own little way the graphical depiction of the SOA supervisor (The SOA supervisor acts like a traffic cop and helps prevent SOA accidents) to our overall SOA model. We have also made the computer network and the Internet visible, for two reasons:

To depicts how software components including RTDBS services actually connect with each other across a computer network. In most cases, applications run on separate server machines that connect via the network or possibly over the Internet. The SOA supervisor needs to connect to every other component within the SOA in order to do its job. If we drew each of the connections in, the diagram would get very busy very quickly.

The SOA supervisor manages the end-to-end computer process created by connecting all the other software components together. In our illustration, applications are divided between external components (components outside the corporate network) and *internal* components (components inside the corporate network). The

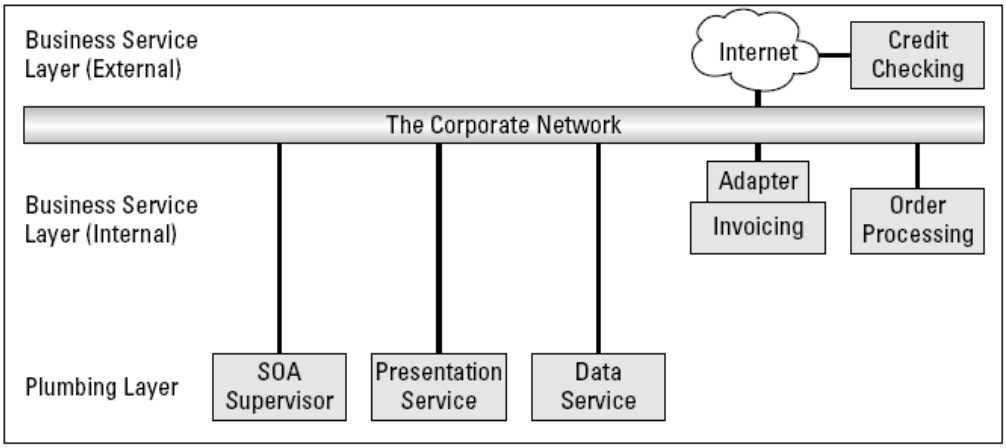

Fig. 5: The SOA supervisor
(Adapted from Service Oriented Architecture for Dummies by Judith et al., 2007).

credit checking component, for example, is an external component that is connected through the Internet.

One of the SOA-RTDBS supervisor's responsibilities is to monitor the various components within the SOA-RTDBS. The SOA-RTDBS supervisor directly monitors only things in its purview. However it can also monitor results and responses from services provided from the outside.

We may not be able to do much if an external service suddenly fails or goes very slowly. However, with internal components, the SOA supervisor not only monitors the whole service that a component provides but may also initiate corrective activity if things start to go wrong.

SOA-RTDBS security

RTDB professionals have always viewed security as a network perimeter issue (Josuttis, 2007). Protecting the RTDB system consisted of deploying firewalls and access control mechanisms. SOA systems are integrated with RTDBS. The edge of the RTDBS is not where the security ends. Another major problem is how to implement access control for a large number of different identities, who are outside the RTDB system security realm (Yuan and Tong, 2005). In the traditional security realm users are known beforehand, and this is not always possible with RTDBS in SOA market. In addition, since applications and application logic can be addressed through services, more data is exposed. This increase in data exposure can increase potential damage since very single service can be seen as a potential attacking point (Pajevski, 2004). We provide below an overview of threats of RTDBS in a SOA security market that can be found in the literature.

Threats to RTDBS services

RTDBS services provide the means to interact with (legacy) applications and/or systems. Each RTDBS service can in fact be seen as a possible point of attack. Security requirements of these systems can be different from the service client, it can also provide more functionalities. A RTDBS service can for instance give access to desktop applications and custom applications that would normally not be accessible to a client. Providing authorization for standalone applications can be hard to manage (Peterson, 2006). If authorization is not managed correctly, access could be granted (or denied) to service clients that should not have access, thereby possibly accessing sensitive information. Differently from an application inside an organization, RTDBS in SOA market (services) can also extend beyond the perimeters of the organization. Traditionally authorization is performed on the front-end of a system / application. In a basic SOA, a loosely coupled service does not have a coordinating service that provides security features. Moreover, the loose coupling predicts that none of the services is aware of its context. A resource that a service provides could require authentication/authorization. The RTDBS service client must then provide the required information to authorize for that specific service. Because of the loose coupling of services, securing the confidentiality and integrity of the message could pose a problem. Traditionally transport level protocols (such as SSL/TLS) were used between two endpoints to maintain confidentiality and integrity. Since RTDBS services are also location transparent, it is not possible to predict where the endpoints are and if they can be trusted. Therefore, instead of using transport-level security, message-level security should be employed (Erl, 2005).

Service registration / deregistration

The service repository (which can be either within the Database Service (DBS) security domain or outside) can be susceptible to replay attacks. An adversary could capture the registration or deregistration of a RTDBS service and perform a replay attack. This replay attack could result in some sort of denial of RTDBS service attack or registration of an (insecure) older RTDBS service. An adversary could also perform an enumeration attack which allows the adversary to create an inventory of available RTDBS services (Beznosov et al. (2005)) and (Schushel and Weske, 2004). To prevent these attacks authentication, authorization, integrity and confidentiality must be maintained during registration and deregistration (Beznosov et al. (2005).

Use of RTDBS standards

Since RTDBS in SOA market relies on standards, it is imperative that standards have an

emphasize security. Standards used in RTDBS in SOA market (mostly web services standards such as XML, SOAP, UDDI) do not emphasize security (Josuttis, 2007). Some protocols enable security within these standards, such as SSL for HTTP and encryption for XML. An example where usage of standards can lead to problems is with firewalls. Most companies use firewalls that control all traffic from the external organization to the internal organization. When RTDBS in SOA market is implemented using web services, HTTP and SSL are normally used. These protocols use TCP ports 80 and 443 that usually can pass-through the firewall (Pulier and taylor, 2006) and Pajevski, 2004). Therefore, additional security features have to be implemented to prevent possible threats. RTDBS Services require a description language to describe what the service offers and what it requires. (One of these description languages commonly used in SOA is the Web services description language WSDL). Description languages should use open standards to have full compatibility with potential service consumers. The open standards also allow a possible adversary to scan for vulnerabilities in the service. Using standards alone does not provide a secure SOA (IBM and Uwe, 2006) and (Paterson, 2006).

In order to have a secure RTDBS in SOA market assuming that the basic security principles: confidentiality, integrity, availability (CIA) holds. In addition, security principles such as authentication, authorization, auditing and non-repudiation are available. We tried to create a set of elements from the literature that will satisfy these principles.

Secure interaction

Interaction between a service client and service provider in a RTDBS should be secure to prevent threats. Secure in this case means confidentiality, integrity, non-repudiation and authentication of messages between a service provider and service consumer (Nezhad et al., 2006). One way to have secure interaction is to set up a private connection between service client and service provider. Commonly this process is known as transport level security (TLS). However, using TLS in SOA will not be a viable option when there are intermediaries (such as ESB, or service providers) involved. TLS encrypts everything, not just the data that is important, that way it can increase the load on systems that need to encrypt messages. Also important is that it cannot be routed easily (Rahaman et al., 2006) and (Nezhad et al., 2006). If there is some sort of intermediary, the security context only from service consumer to intermediary and from intermediary to service provider, not from service consumer to service provider. A message is therefore encrypted at the service consumer, send to the intermediary who decrypts and encrypts the message again, before it finally arrives at the service provider (or vice versa). This means that the intermediary can read all the data, which is not always allowed or preferred. Message-level security can be a solution. Message level security ensures a message is protected throughout communication of the message (Pajevski, 2004) and this was also buttress by SOA software (http://www.soasoftpressroom.com/SOASoft_Security_Issues_in_SOA_Whitepaper.pdf). This means that confidentiality and integrity of the message is protected all the way from message sender to message receiver. Another interaction that should be secure is that of registration and deregistration of a RTDBS service provider in a service registry. Only authorized services should be able to register with the service repository. During registration, the integrity of the RTDBS service and confidentiality of the service repository should be maintained.

Responses of a service provider to a service client in a RTDBS should always be confidential, to prevent an adversary from building a list of services and their responses. The interaction between a service and a service client should have mutual authentication. Another important part of secure interaction is to maintain the integrity of the service provider and service registry. If the service client interacts with either of these, it has to be sure that the service has not been compromised, and is the service that the client was expecting.

Distributed identities

The basis of both authentication and authorization are (distributed) identities. Access to resources can differ greatly from identity to identity. For instance, discovery of RTDBS services published in the service registry should be limited to specific service clients (this can be both an issue of Quality of service and security). For instance

some RTDBS services should not be accessible to the marketing department, and should not be discovered (Papazoglou et al., 2006) and (Rahaman et al., 2006). Services can be built upon (legacy) applications. These applications can have their own authentication and authorization processes. Managing identities across all the applications can be a difficult task. Therefore, the SOA security model should have distributed identities that can be used across all of the services and applications. Federated Identity Management (FIM) is a solution for the management for identities (Pajevski, 2004). Achieving single sign-on is the goal that distributed identities hopes to achieve. This means that a service client only has to establish its identity once. With this identity, it will be authenticated and authorized to several sources without re-establishing its identity again. Distributed identities are not only assigned to the service client, but also to the service registry and RTDBS service provider. This identity can be used to maintain the integrity of the service registry and RTDBS service provider. There can be a difference between identity of a service and service client. The service itself is not required to have an identity because it can impersonate (propagate) the identity of a client. The service client must always have an identity (presuming anonymous access is never allowed).

Distributed policies in RTDBS

These policies define the rules that authorize a RTDBS service client to access a service provider. These policies should be able to take into account a specific context and the identity of a RTDBS service client (Peterson, 2006). The policy defines what an authenticated identity is authorized to do. In addition, policies can be created that define when a service provider can no longer handle any more RTDBS service clients. This aids in maintaining the availability of the service provider. Other policies that can be created are trust polices. Trust policies help manage the security in. A service client can be from any specific domain. A domain can for instance be an organization or a department of the organization. Incorporating "all" domains is unmanageable. Therefore, a service provider could use trust relationships with other service providers to trust other service clients. For instance service provider A has a trust relationship with service provider B. Service B trusts service client C. Service A can therefore choose to also trust C. Trust policies can however be hard to implement since trust relationships between interdepartmental and inter-organizational can differ. Distributed policies can only authorize a service client if it provides an (distributed) identity.

Model for the secure SOA-RTDBMS

We add the security elements to the Basic SOA in Figure 1. This produces the model in Figure 6 (adapted from (CSI/FBI Computer Crime, 2006) and (seduhkin, 2003) for a secure RTDBS in SOA. This is not a model for an implementation of a RTDBS-SOA. When implementing RTDBS in SOA, one should implement a security system that takes care of all the policies, manages the identities and provides secure interaction.

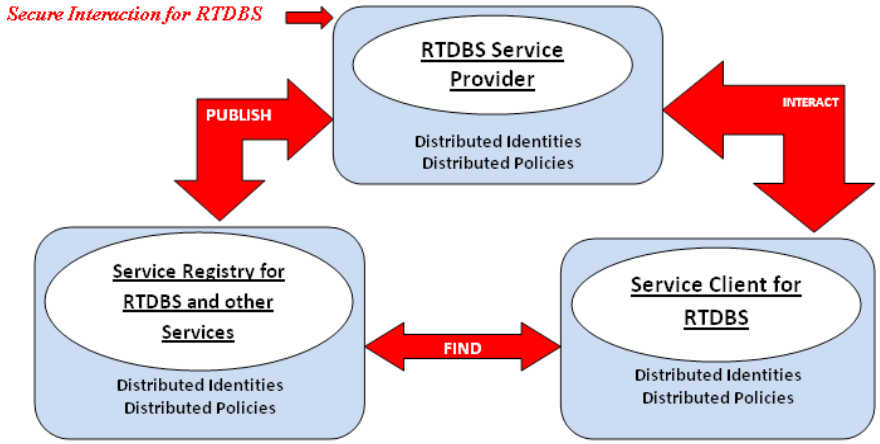

Figure 6 Model for secure SOA-RTDBMS

The security threats of RTDBS in SOA market include threats to services in general. RTDBS services can provide functionalities to users that were not available before the service was in place. In addition, RTDBS services can exist beyond the organization's security perimeter. Since RTDBS services use standards, a possible adversary can use flaws in these standards to attack the RTDBS service. These threats prevented by introducing security principles into the RTDBS-SOA model. These principles include secure interaction, distributed identities and distributed policies. Secure interaction provides confidentiality and integrity of messages between service providers, service registry and the service client. Distributed identities are used as the basis to provide authentication, authorization, integrity and non-repudiation. Distributed policies are used for authorization and availability. A service client can be authorized to access a service provider, or can be authorized access the service registry. For authentication and authorization, we usually need an identity claim (which can be user ID and password or certificated etc.). Encryption and digital signatures on the other hand are used for confidentiality and integrity. Table 1 shows which security principles can be addressed with a specific security standard.

Table 1 Standards in Secure SOA model (Adapted from SOA Security by Jamie (2007)).

SOA security element	Security principle	Security standards
Secure interaction	Confidentiality	HTTP with TLS
	Integrity	WS-Security
	Non-repudiation	XML signature and XML Encryption
		XML Signature
Distributed identities	Integrity	XACML
	Authentication	WS-Security tokens
	Authorization	RBAC / ABAC / LDAP
		HTTP with TLS
		X5.09
Distributed policies	Authorization	SAML

Why SOA is a Better Business and Better IT for RTDB managers

SOA make it easier and faster to build and deploy systems for database and other application in real-time which directly serve the goals of RTDB managers and designer. Contemporary RTDB manager is completely reliant on its IT, and never have business and IT needed to be more aligned. The very survival of a business hinges on its ability to adapt its IT to meet ever-changing business challenges. SOA integrates business and RTDB into a framework that simultaneously leverages existing systems and enables business change. A SOA enables the RTDB managers to keep their focus on business and allows IT to evolve and keep pace in a dynamically changing world. RTDB managers need not understand the intricacies of the plumbing layer and everything it contains. If we cover up the plumbing layer, we are left with a diagram that shows all the business services that software applications provide, both inside our organization and to others that interact (technologically speaking) from outside, like our customers, business partners, and suppliers. Looking at our organization's software resources in this way, we may be able to think about ways to improve or better exploit the software assets we have. Likewise, if we cover up all the business functionality in our SOA diagram, we are left with a set of plumbing services that our IT department is responsible for providing. We know that many of our "legacy" applications also have a good deal of plumbing in them, and the plumbing layer does not replace that. However, SOA enables an IT department to choose how it will evolve toward

providing a "service oriented architecture" and in time may obviate a good deal of lousy plumbing.

RTDBS in a SOA market may not initially guarantee a happier, healthier life, free from business concerns. However, movement toward SOA by the RTDB managers will be a movement toward technical freedom and business flexibility and bodes well for the performance and profitability of an organization and for the sanity of the people managing the business. In SOA, the intention is not to throw away the real system that has been in use over many years, all the existing business applications including our database can be reused. An entire application can be treated as a service, or codes can be amended out of an application and the code make into a service. If the order-processing application is placed on the Internet, customers can use their browsers as the standard user interface to place an order, make a payment, and enter other needed information such as name, billing address, shipping address, and so on. To "put" the order-processing application on the Web requires the use of standard Web services interfaces. After that's done, browsers can talk to the order-processing application, and everyone can live happily ever after. Earlier efforts to promote software reuse required everyone to use the same computer language or operating system or foundation classes, but the entire industry could never be pinned down to use the same programming language. Web services made it possible for the universal acceptance of the underlying standards of the Internet. After every browser could talk to any Web site, it became possible for any computer program to talk to any other program, as long as they both had some path to the Internet. If every function of the order-processing application is a Web service, other programs can use those functions instead of reinventing the wheel.

Recommendation

Despite the strong trend in SOA, some in the IT community don't feel that the web services underpinning for an SOA is mature enough for their enterprise to consider migration to a service-oriented architecture. We therefore believe that software vendors will have to change their applications to come into line with the industry movement or risk their own extinction. They will also have to change their licenses to cover the use of their applications as a set of modular components that can be linked together with Web services.

CONCLUSIONS

There is a great potential in RTDBS in SOA market. As a web service-based, it will speed up the application development process and response system time. It is also a way to build database application systems that is more adaptable, more agile in responding to changing business needs. RTDBMS in SOA market clearly is the wave of the future; soon RTDBS in SOA market will be a prevailing software engineering practice for RTDB managers which will end the 40year domination of monolithic software architecture.

The field of real-time database research has evolved a great deal over the relatively short time of its existence. In the early 1980s, much of the research concentrated on examining how to add real-time support to traditional databases in a monolithic environment. Much of this work involved developing new scheduling and concurrency control techniques that extended existing techniques to allow RTDBs to provide timely transactions and temporally valid data. This has included techniques to relax traditional ACID properties of databases, and managing QoS provided by the database in order to allow for timing constraints to be met in a SOA setting. The future of RTDB research will depend on this continuing evolution. Research in this field should continue to follow cutting edge applications and apply existing techniques to them, as well as developing new ones when needed. It is also important that researchers keep up with new technology in traditional database research. As new approaches to solving database issues are developed, it is important to investigate how such innovations can apply to real-time applications in a SOA setting. On a more practical note, it is important that RTDB research be widely applicable. Many academic papers mention various applications like programmed stock trading, medical patient monitoring, etc., to which their technologies will apply. However, there are very few real-time databases in use in such applications and few applied it to web services while none thought of enhancing their application using SOA paradigm.

In order for this research to continue, it must be shown to be practical and useful in real applications.

We encourage RTDB managers to begin SOA now for their own benefit. Ultimately, SOA will render a business more flexible and RTDBS more reliable, sustainable, extensible, manageable, and accountable. We believe SOA is the most important mandate facing business and RTDB managers today. And because SOA is a joint venture between business managers and RTDB managers, we present the basics necessary for everyone to come to the table with a good grounding from a conceptual level.

REFERENCES

1. Aniruddha Gokhale, Bharat Kumar, Arnaud Sahuguet (2003): Reinventing the Wheel? CORBA vs. Web Services, The Eleventh International World Wide Web Conference, retrieved from http://www2002.org/CDROM/alternate/395/
2. Beznosov, K. and Flinn, D.J. and Kawamoto, S. and Hartman, B. (2005): *Introduction to Web services and their security.* Information Security Technical Report 10, 2-14, Elsevier Ltd.
3. Chen, D., and Mok, A.K. : SRDE-application of data similarity to process control. In the 20th IEEE real-Time Systems Symposium. Phoenix, Arizona, December (1999).
4. CSI/FBI computer crime and security survey (2006): http://www.cse.msu.edu/~cse429/readings/FBI2006.pdf
5. Ed Ort : Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools, Sun Microsystems, Inc (2005).
6. Erl, T. : *Service-Oriented Architecture* Prentice Hall PTR ISBN 0-13-185858-0 (2005).

a. Establishing Information Quality Baselines for Complex, Distributed Systems, 10th

7. Folorunso Olusegun, Longe H. O. D., and Akinwale A. T. Visualizing Concurrency Control Algorithms for Real-Time Database Systems. *Data Science Journal,* **7**, 20 (2008).
8. Haritsa, J., Ramamritham, K., and Gupta, R. : The Prompt real-time commit protocol. IEEE transactions on parallel and distributed systems 1192); 160-181 (2000).
9. Ho, S., Kuo, T., and Mok, A.K. : Similarity-based load adjustment for static real-time transaction systems. In proceedings of the 18th Real-Time system symposium (1997).
10. IBM, Uwe Kissmann : SOA Security, http://www.ibm.com/ru/events/presentations/bf2006/soa_security.pdf a. International Conference on Information Quality (ICIQ) (2006).
11. Jamie Fiere : SOA Security. A Msc thesis for degree of master of Information Science, Faculty of Science Vrije Universiteit Amsterdam. (2007).
12. Joseph Bugajski, Robert Grossman, Eric Sumner, Tao Zhang: A Methodology (2005).
13. Josuttis, N.M. : *SOA in Practice*, 1st edition, O'Reilly Media Inc. (2007).
14. Judith Hurwitz, Robin Bloor, Carol Baroudi and Marcia Kaufman : Service Oriented Architecture for Dummies; Wiley Publishing Inc. (2007).
15. Krafzig, Dirk, Banke, Karl, Slama, Durk : *Enterprise SOA, Service Oriented Architecture Best practices*. Prentice Hall, November 2004. ISBN 0-13-146575-9 (2004).
16. Krithi Ramamritham, Sang H. Son and Liza Cingiser Dipippo (2004): Real-Time databases and Data Services; Kluwer Academic Publisher, vol. 28, pp 179-215.
17. Kuo, T., and Mok, A.K.: SSP: a semantics-based protocol for real-time data access. IEEE 14th Real-Time System Symposium (1993).
18. Kuo, T., and Mok, A.K. : Real-Time data semantics and similarity based concurrency control. IEEE Transactions on computers 49(11); 1241-1254 (2003).
19. LaPlante, A.: Education Key to SOA Success, in SOApipeline.com, Nov. 21 (2005).
20. Mike Rosen, Boris Lublinsky, Kevin T. Smith and Marc J. Balcer : Applied SOA: Service

Oriented Architecture and Design Strategies; Wiley Publishing, Inc. Indianapolis, Indiana (2008).

21. Nezhad, H.R.M and Skogsrud, H. and Benatallah, B. and Casati F. : *Securing Service-Based Interactions: Issues and Directions*. (2006) http://info.computer. org/portal/site/ dsonline/index.jsp?pageID=dso_level1&path=dsonlin e/topics/was/papers&file=motahari.xml&xsl=article.xsl
22. Nizar Idoudi, Nada Louati, Claude Duvallet, Rafik Bouaziz, Bruno Sadeg and Faiez Gargouri : A Framework to Model Real-Time Databases; International Journal of Computing & Information Sciences. Vol. 7, No. 1, January 2009, On-Line (2008).
23. OASIS : *Reference Model for service oriented architecture 1.0*, Committee specification 1, 2 august 2006 (2006). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm
24. Pajevski, M. J.: *A Security Model for Service Oriented Architectures* (2004). http://www.oasis- open.org/committees/download.php/17573/06-04-00008.000.pdf
25. Papazoglou, P, M. : *Service-oriented computing: Concepts, Characteristics and Directions*. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering (2003).
26. Papazoglou, P.M. and Traverso, P. and Dustbar, S. and Leymann, F. : *Service-Oriented Computing Research Roadmap* (2006).
27. Papazoglou, P.M. and van den Heuvel, W. : *Service Oriented Architectures: Approaches, Technologies and research Issues*. The VLDB Journal (2007).
28. Perrey, R. and Lycett, M. : *Service-Oriented Architecture*. Symposium on Applications and the Internet Workshops, 2003. Proceedings. page(s): 116- 119 (2003).
29. Peterson, G., Lipson,H. (2006): *Security Concepts, Challenges, and Design Considerations for Web Services Integration*. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best- practices/assembly/639.html, Carnegie Mellon University.
30. Pulier, E. and Taylor, H. (2006): *Understanding Enterprise SOA*, Manning Publications, ISBN 1-932394-59-1
31. Rahaman, M.A. and Schaad, A. and Rits, M. : *Towards Secure SOAP Message Exchange in a SOA*, Proceedings of the 3rd ACM Workshop on Secure Web Services (2006).
32. Schuschel H. and Weske M. : *Automated Planning in a Service-Oriented Architecture.* Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE 2004 page(s): 75- 80 (2004).
33. Seduhkin, I.,: *End-to-End Security for Web Services and Service Oriented Architectures*, Computer Associates. (2003) http://www.webservices.org/companies/ca/end to end security for web services and services_oriented_architectures
34. SOA Software : Whitepaper Security issues in the SOA. (2005). http://www.soasoftpressroom.com / SOASoft_Security_Issues_in_SOA_Whitepaper.pdf
35. Son. S.: Using replication for high performance database support in distributed Real-Time systems. In proceedings of the 8th IEEE Real-Time Systems symposium. Pp 7986 (1987).
36. Son. S., and Kouloumbis, S.: A real-time synchronization scheme for replicated data in distributed database systems. Information Systems 1896); 79-86 (1993).
37. Song, X., and Liu, J.W.S.: Maintaining temporal consistency. Pessimistic vs. Optimistic concurrency control. *IEEE Transactions on Knowledge and Data Engineering* **7**(5): 786-796 (1995).
38. Xiong, M., Ramamritham, K., Stankovic, J.A., Towsley, D., and Sivasankaran, R. : Scheduling transactions with temporal constraints exploiting data semantics. *IEEE transactions on knowledge and Data Engineering.* **14**(5): 1155-1166 (2002).
39. Yuan, E. and Tong, J. : *Attribute Based Access Control (ABAC) for Web Services*, Proceeding of the IEEE International Conference on Web Services (ICWS'05) (2005).
40. Zhou, H., and Jahanian, F.: Real-time primary-backup (RTPB) replication with temporal consistency guarantees. In proceedings of ICDCS (1998).