International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-12 Issue-2, January 2023

Applications Integration in a Semi-Virtualized Environment

Mbaiossoum Bery Leouro

Abstract*: Enterprise application integration quickly emerged as a challenge for companies, and various solutions were proposed, including point-to-point architecture, ETL (Extract, Transform, and Load), EAI (Enterprise Application Integration), and ESBs (Enterprise Service Bus). With the rise of virtualisation, applications in a physical environment can interact with those in a virtualised environment. The objective of this paper is to study and define a Service Oriented Architecture (SOA) in a semi-virtualised climate. The authors propose an architecture which allows service integration in a semi-virtualized environment. This study employs a survey-based technique to determine which technology can be utilised in the considered environment. A state-of-the-art review of various IT designs and solutions that enable SOA implementation is presented. ESB technology has been retained for this study. A literature review is conducted on ESB to examine how it can be utilised in a semi-virtualised context. The findings of this study propose an architecture that allows for service integration in a semi-virtualised environment. After an in-depth examination of various deployment possibilities and technical solutions proposed for this purpose, a new architecture based on the Enterprise Service Bus (ESB) is proposed for the semi-virtualised context. This architecture is organised around two ESB solutions, each deployed in a separate environment and interconnected by a communication bridge that ensures message routing between the two ESB buses. A practical deployment phase is conducted for illustration purposes using Talend Open Studio, yielding encouraging results. The proposed architecture is a good solution for companies whose information systems operate in a semi-virtualised environment.

***Keywords*: Application Integration, Eia, Esb, Soa, Virtualization, Semi-Virtualized Context.**

I. INTRODUCTION

The objective of this work is to propose an architecture implementing SOA in a semi-virtualized context. Many companies have information systems (IS) made up of several components, such as business applications [1, 2], ERP [3] [4] [5] [6], Customer Relationship Management (CRM), databases, etc. Each component is deployed to meet a specific need within a well-defined scope. Applications and software used in daily activities are therefore not necessarily required to communicate with each other [2]. Data are partitioned. There are two significant questions [7]: how to ensure data consistency and data propagation between several subsystems and how to initiate, in response to an event in a given subsystem, a processing in another subsystem which is foreign to it? The diversification of components and business processes in IS forces companies to resort to integration architectures. Integration platforms link all business applications together to cross and combine data used by each of them [7] [8] [9] [10]. They optimize the use of data for company users. Formerly known as “point to point”, integration can now be automated thanks to numerous technical solutions, in particular Service Oriented Architecture (SOA) [11, 12]. A distinction can be made between data integration, which involves synchronising databases used by various IT components, and application integration, which occurs through the transmission of messages between applications. Four (4) technical solution families are offered in these two integration categories: ETL (Extract, Transform and Load) [13] [14] [15], point-to-point architecture [11], EAI (Enterprise Application Integration) [16] [17]18] and ESB (Enterprise Service Bus) [19] [20] [21].

ETLs are solutions intended to extract data from various data sources, transform them into a format understandable to their target and load them into one or more target databases [14, 15]. They are much better suited for processing large datasets, particularly when loading data into data warehouses. ETLs only respond to the consistency and propagation of data between multiple subsystems. ETLs are a solution that fails to address the challenges of modern integration effectively and is not well-suited for implementation in SOA within a semi-virtualised context.

Point-to-point architecture, also known as strong coupling, emerged in the early 1990s, utilising specific connectors to connect each application to another without a middle layer. It is a mechanism by which the client is linked directly to its server via a protocol, a specific format, and potentially hard-fixed addresses [11]. Any modification of one of the components may require a new delivery of the others to maintain the system's operation. This is often called “spaghetti dish syndrome”. It is an architecture where evolution and maintenance become increasingly complex as the size of the information system increases. It remains a proprietary solution, costly, and above all, does not use recognised standards protocols to facilitate its opening to the outside world.

EAIs (Enterprise Application Integration) come with a new architecture based on the so-called "hub and spokes" topology, allowing inter-application information exchange through a central point (hub) [16]. Communication between applications is ensured at this point according to its standards. EAI intervenes at the application level, thanks to specific connectors (spokes), to exploit data consumed and stored by each application.

Manuscript received on 25 December 2022 | Revised Manuscript received on 03 January 2023 | Manuscript Accepted on 15 January 2023 | Manuscript published on 30 January 2023.
*Correspondence Author(s)
Bery Leouro*, Department of Computer Science, University of N’Djamena, N’Djamena, Chad. Email: bery.mbaiossoum@gmail.com, ORCID ID: https://orcid.org/0000-0001-8620-0668

This solution is a very effective way to break the strong coupling employed by point-to-point architecture, but it still has a proprietary character. Its internal topology, the protocols used, and the exchange formats are all proprietary, as are its connectors, which allow it to access applications and remain largely vendor-specific despite attempts at standardisation. Depending on its technical specifications, its complexity and above all, its proprietary nature, this solution is not profitable for companies in terms of cost and time savings [17, 18].

ESBs (Enterprise Service Buses) are a new generation of application integration solutions, seen in some ways as the heir to EAI [7, 19]. ESBs are built on open standards such as XML for message description and Web Services for data exchange. Unlike EAIs, ESBs do not intervene directly in system applications but via Web Services modules within each of them [20, 22]. All of these service modules are then directed and shared to a central flow where exchange takes place on the application bus. The deployment of new tools and components in the information system is thus facilitated by this method. Because of their architectures based on message buses and especially their implementation based on known communication standards and their ability to exchange messages with other ESB solutions from different manufacturers, ESBs have become the favoured technology of integration for implementations of Service Oriented Architecture [19]. Given their advantages, it is the best-suited solution for an architecture based on SOA in a semi-virtualized context [23] could be defined. Thus, a SOA implementation architecture utilising ESB-type solutions for service integration in a semi-virtualised context is proposed.

In the remainder of this paper, the materials and methods are presented in Section 2, and the results are presented in Section 3. Then, discussions are presented in point 4. And, the paper ends with a conclusion.

II. Materials and Methods

A literature review is conducted on ESB to examine how it can be utilised in a semi-virtualised context. The Enterprise Service Bus (ESB) is the primary technology used in this work. Roy Schulte defines Enterprise Service Bus (ESB) [24] as "a new architecture that leverages Web services, message-oriented systems, intelligent routing and transformation." ESB acts as a lightweight and ubiquitous backbone of integration through which software services and application components flow. Other definitions followed that of Roy Schulte [24], their synthesis retains ESB as a middleware solution based on a service-oriented model and bus principles, allowing interoperability between heterogeneous applications and facilitating the implementation of the SOA by promoting the use of Web Services [25]. ESB is often considered a new generation of EAIs built on standards such as XML, JMS (Java Message Service), or Web Services. Significant difference with EAI is that ESB offers fully distributed integration through the use of service containers, which contain the integration logic and can be dropped anywhere on the network [20, 22]. An ESB service container abstracts a service, isolating it from its communication protocol, invocation methods, message exchange patterns, QoS requirements, and other infrastructure needs. A real breakthrough of ESBs is their ability to virtualise services.

A. Principles of an ESB

From a strictly technical point of view, the role of an ESB boils down to connecting and mediating between heterogeneous services and applications within an IS, as well as decoupling consumers and service providers. As a mediator between clients and service providers, ESB is based on the following principles [19]:

- Dynamic discovery: services as well as associated semantics are recorded in a shared directory;
- Choreography of business processes and orchestration of associated services: A tool automatically orchestrates necessary services for the implementation of collaborative processes.
- Strong distribution: services are distributed over the company's network or the Internet.
- Message communication: services exchange messages represented by textual documents.

B. Main characteristics of an ESB

To fully play its role, an ESB must have some characteristics and functionalities, of which the main ones are [7, 20, 22]:

- Connectivity and communication: An ESB must support communication with multiple protocols and formats based on standards recognised by all (e.g., SOAP, XML, JSON, JMS), allowing synchronous or asynchronous communication. To achieve this, it must, in particular, be able to interact with Message-Oriented Middleware (MOM), whether embedded within or not part of the ESB.
- Mediation, routing, and orchestration: An ESB must enable the decoupling of data suppliers and consumers. It must therefore provide a mechanism that allows for the transparent distribution of messages from a supplier to one or more consumers, benefiting both parties. Mediation is the principal added value of an ESB. In this field, it must therefore offer a rich semantics, capable of making exchanges more reliable and simplifying implementation.
- Transformation: Since a supplier and a consumer do not necessarily communicate in the same language, an ESB must provide means for transforming messages.
- Monitoring (monitoring tool): As a central point, ESB must allow monitoring and reliability of exchanges which pass through it, thanks in particular to:

  - guarantee of message delivery while keeping unused messages (application unavailable, failure of a transaction, etc.);
  - functionalities making it possible to monitor processing operations carried out and the messages received;
  - service security management (authentication, authorization, confidentiality and audit);
  - Control of SLAs (Service Level Agreements) and the ability to modify bus behaviour (priorities, etc.) to ensure these SLAs.

Retrieval Number: 100.1/ijitee.B94030112223
DOI: 10.35940/ijitee.B9403.0112223
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

C. Types of deployment of ESB solutions

ESBs are increasingly used in SOA projects under different approaches at the infrastructure deployment level [7, 22]. Each approach has advantages and disadvantages. Two approaches which can be helpful in the definition of a new architecture are presented:

- separate ESB servers in the company, each solving a specific integration problem for a specific department;
- A distributed ESB in the company connects all parts of the information system.

The first approach is to let each department manage its SOA by implementing its own ESB solution. The department oversees the integration of the application and the development of its business processes. By using separate and independent ESBs, each department is free to adopt the solution it wants. But when communication with other partners is required, it must explicitly access or provide certain services using standard "bridge" technology such as web services. In this case, the communication relies on protocols such as HTTP, and quality of service, such as reliability or security, must be implemented manually [7].

The second way to implement a SOA infrastructure is to have a unified ESB across the company. The ESB is deployed only once to ensure communication between the various department servers. Invocation of a service by another service will be done in a simple way, since the consumer contacts the ESB in the same way as he does for a local service [7]. From this perspective, ESB can be considered the real backbone of this infrastructure, much like an Ethernet network. In more practical terms, ESB is a set of natively interconnected nodes. A node is a connection point for consumers or service providers. A unified ESB deployment approach has multiple technical advantages. Communication between subsystems is simplified by facilitating service invocation methods that are similar to calling a simple local service. Its deployment and maintenance are easier compared to the separate approach. From an organizational point of view, this approach can be a source of conflict of competence between the administrators who manage the different business services. In a large company, particularly one with multiple business services, they are often organised by department. ESBs in their default configurations allow generalized access to all services they connect and provide an administrator with the privilege to modify them. To resolve this generalised access problem, which could likely be a source of insecurity, a method of segmenting an ESB into a sub-part called a domain, defining the management limit is implemented.

D. Domains in a unified ESB

In ESB architecture, domains and subdomains are techniques for defining boundaries between entities. Thus, from a technical point of view, each "node" of the ESB can only be managed by the administrator of the relevant subdomain. He is therefore the only person authorized to start, stop, install connectors and deploy services on this node. The administrator will be able to manage the ports used by the ESB nodes in their domain and has the option of deploying proxies or firewalls to protect them. A business service deployed on a domain node can be public or private. A private service is visible only in the registry and to consumers in its domain. Additionally, it is only accessible to service consumers within its domain. Tools for monitoring processes and services have access to information that concerns their domains and those of public domains. It is not possible to visualize processes of a private domain or statistical information concerning its services.

III. RESULTS

The proposed architecture is based on the deployment of two separate ESBs (one in each environment), then linked together by a bridge technique:

- An ESB deployed in the physical environment: allows the integration of applications or services that simply operate in this environment.
- An ESB deployed in the virtual environment enables the same scenario of the virtual environment to be carried out;
- A communication bridge between the two environments to ensure exchange of messages between the two ESB buses, which will therefore allow an interaction between the subsystems of these two distinct environments.

A. Physical Environment Architecture

Architecture planned as part of the implementation of SOA in the physical environment is essentially an ordinary deployment of the technical solutions provided for this purpose. In this specific case, a unified ESB solution will mediate between different services running in this environment. Fig. 1 gives an overview of this architecture.

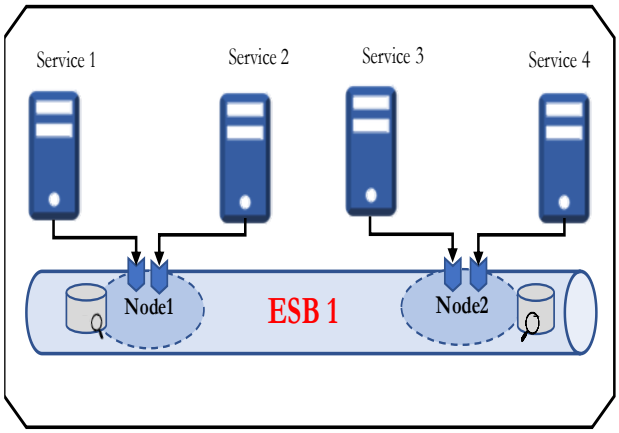

Figure 1: Deployment Architecture in Physical Environment

B. Virtual Environment Architecture

As expected, interoperability between services in each environment must be ensured by an independent ESB, and the exchange between them will be through the bridging technique. Fig. 2 shows schematically this deployment.

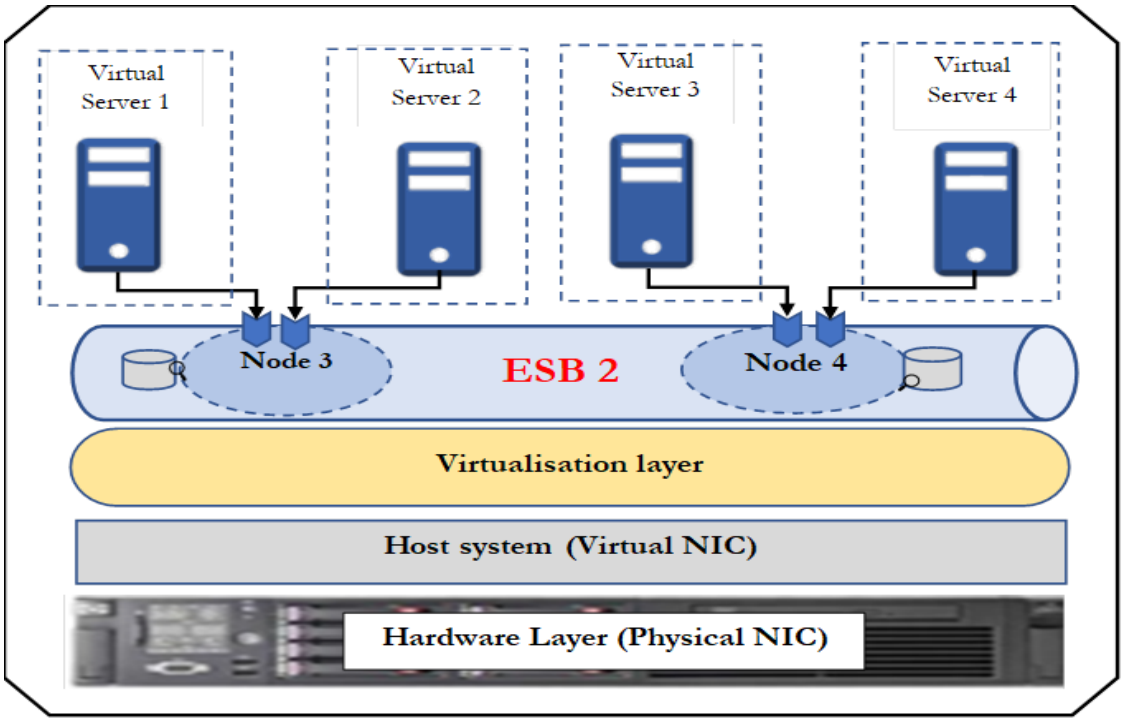

Figure 2: Deployment architecture in a virtual environment

C. Overall architecture

The overall architecture is a combination of the two architectures presented above. Their communication is made possible by a communication bridge, which provides specific routing between physical and virtual network interfaces. This technique enables message packets sent from one bus to be transmitted to another. Fig. 3 schematically presents this overall architecture.

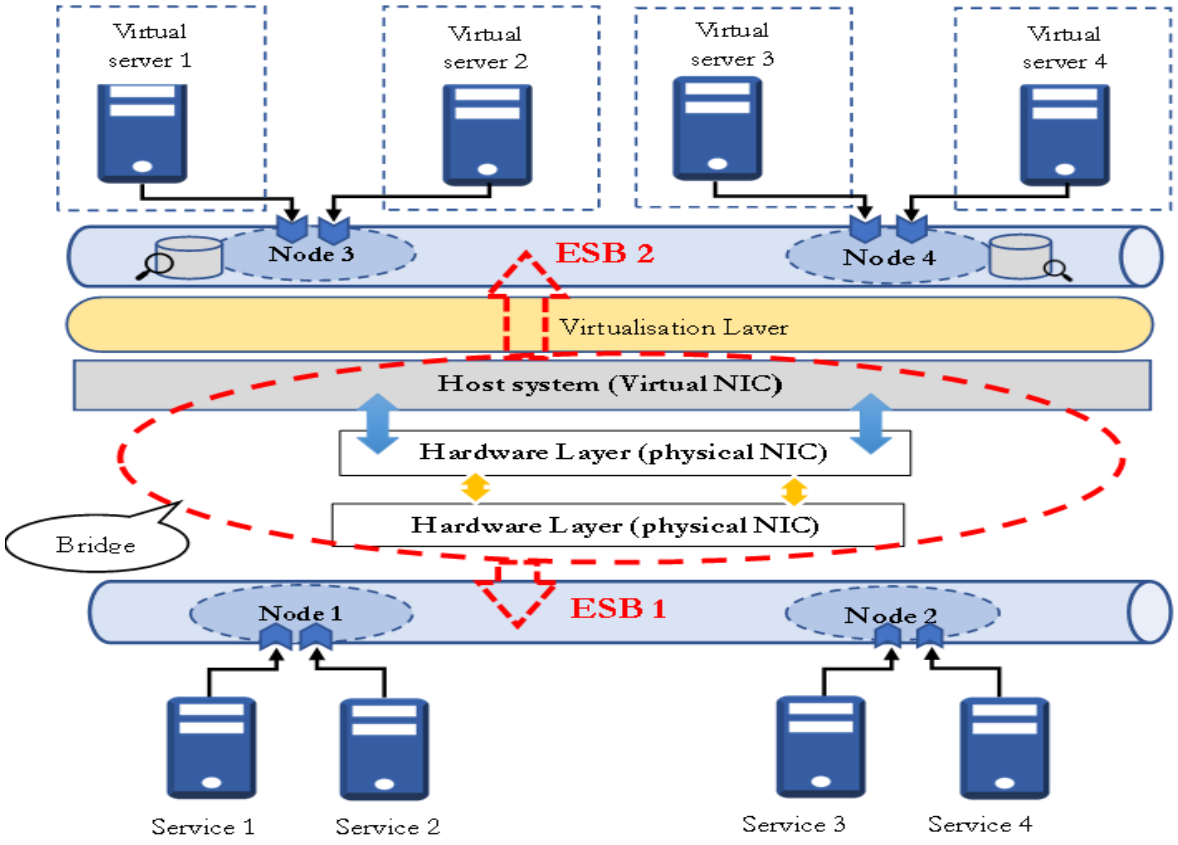

Figure 3: Overall architecture

IV. DISCUSSIONS

The limitations associated with point-to-point integration can hinder a business's growth and are usually mitigated by adopting an ESB. With ESB technology, integrating systems and applications has become easier and faster, supporting enterprise-wide change and growth. The flexibility of the ESB bus is an essential advantage. Version updates or application location changes, as well as fluctuations that occur anywhere on the network, are no longer a problem. The ESB also facilitates orchestration, which refers to the technical ability to coordinate different applications. It is also called "routing" or "mediation".

These advantages and the limitations of the previous technologies mentioned above lead to the choice of ESBs for defining the architecture. To demonstrate the feasibility of the proposed architecture, a deployment scenario in a semi-virtualised environment is proposed: Imagine a commercial company that conducts its activities in traditional physical stores, utilising a Customer Relationship Management (CRM) and an inventory management system (ERP), each deployed on a separate physical server. The company decides to establish an online commerce service (e-commerce) to enhance its business and, simultaneously, create a platform that enables the generation of a real-time dashboard based on the commercial flow, facilitating better business planning and informed decision-making. The proposed architecture allows a rational use of its resources. This involves deploying two new services on a single physical server using virtualization technology. That seems very important in terms of investment, management, maintenance, and other related aspects. This situation reflects the reality of many businesses today. For the implementation, Talend Studio [26] is used. The installation and configuration of Talend Open Studio for ESB (TOS-ESB, an open-source version) were completed. Then, Web services were created and exposed in a service directory, allowing them to be accessed and invoked by other consumer services. They are:

- A Web service (supplier) providing information on the stock status of a product requested by the consumer (e-commerce platform). These items are collected from the inventory management database (ERP).

A Web service (supplier) enables the provision of detailed information about a client requested by the consumer (e-commerce platform). This information is obtained by querying the customer relationship management (CRM) application.

- A Web service (supplier) making it possible to expose to consumers (dashboard platform) the trade flows necessary for displaying summary information on the company's current affairs;
- Three jobs for the consumer side, allowing the collection of the input flows and sending the request to each supplier;
- A bridge configuration that enables the exchange of messages between the two ESB solutions deployed in different IT environments (physical and virtual).

Once the Web services are built and tested in Talend Studio, they must be deployed in a Talend Runtime container OSGi to put them into operation. In the lack of a compatible technical environment, services were exported and executed in a Talend ESB container for demonstration purposes. This demonstration allows you to see the step-by-step implementation of a Web Service in Talend Studio, using a job that listens to all requests and another to consume the Services. It also allows for viewing the exchanges between services in the Talend ESB container in both physical and virtual contexts.

V. CONCLUSION

The information systems (IS) of many companies are generally based on heterogeneous software and data sources, resulting from the successive use of various technologies or the acquisition of other companies. This heterogeneity leads IT departments to consider integration issues as a significant concern when companies seek to enable different departments to communicate with each other to optimise their business processes. Companies whose IS operates in a hybrid environment (part of the IS function in a physical environment and another in a virtual environment) also need to ensure that the different departments communicate effectively. Services integration in this specific context is considered. Service-Oriented Architecture (SOA) and technical solutions related to the integration problem are addressed. An architecture is proposed that enables the integration of services in a semi-virtualised environment. This architecture is organised around two ESB solutions, each deployed in a separate environment and interconnected by a communication bridge, which ensures message routing between the two ESB buses. An implementation in Talend Studio showed that this architecture is achievable. It is a solution for companies whose information systems operate in a hybrid mode.

DECLARATION

Funding/ Grants/ Financial Support	No, I did not receive.
Conflicts of Interest/ Competing Interests	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval or consent to participate, as it presents evidence that is not subject to interpretation.
Availability of Data and Material/ Data Access Statement	Not relevant.
Authors Contributions	I am the sole author of the article.

REFERENCES

1. BIDAN, Marc. Fédération et intégration des applications du système d'Information de Gestion. Systèmes d'Information et Management, 2004, vol. 9, no 2, p. 5.
2. DE CORBIÈRE, François, ROWE, Frantz, et WOLFF, François-Charles. De l'intégration interne du système d'information à l'intégration du système d'information de la chaîne logistique. Systemes d'information management, 2012, vol. 17, no 1, p. 81-111. [CrossRef]
3. CHTIOUI, Tawhid. ERP: les effets d'une" normalisation" des processus de gestion. In : Normes et Mondialisation. 2004. p. CD-Rom.
4. BESSON, Patrick. Les ERP à l'épreuve de l'organisation. Systèmes d'information management, 2016, vol. 21, no 2, p. 17-47. [CrossRef]
5. COAT, Françoise et FAVIER, Marc. Passage à l'ERP et refonte du système d'information : le cas des ASF. Systèmes d'Information et Management, 2016, vol. 4, no 4, p. 6.
6. LEE, Zoonky et LEE, Jinyoul. An ERP

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

implementation case study from a knowledge transfer perspective. Journal of Information Technology, 2000, vol. 15, no. 4, pp. 281-288. [CrossRef]
7. Adrien Louis works, ESB Topology Alternatives, [Online], URL: https:// https://www.infoq.com/articles/louis-esb-topologies/, visited in November 2022.
8. Bell Marks, « Service-Oriented Architecture: a planning and implementation guide for business and technology », Hoboken, John Wiley & Sons, 2006.
9. WAHYUNI, E. D., NOVITASARI, D., HADI, M. S., et al. Integration at the Application Interface Level between SIAMIK and SIPERPUS (a Case Study in XX). In: Journal of Physics: Conference Series. IOP Publishing, 2020. p. 022047. [CrossRef]
10. PRIANTARI, Rini et KURNIAWAN, Novianto Budi. Integrated IT service platform: systematic literature review and meta-analysis. In: 2018 International Conference on Information Technology Systems and Innovation (ICITSI). IEEE, 2018. p. 562-567. [CrossRef]
11. Erl Thomas, « Service-Oriented Architecture, Concept, Technology, and Design », Indiana, Prentice Hall, 2006.
12. CASEAU, Yves. Urbanisation, SOA et BPM-4e éd.: Le point de vue du DSI. Dunod, 2011.
13. D. Nguyen-Xuan, Intégration de bases de données hétérogènes par articulation apriori d'ontologies : application aux catalogues de composants industriels. Thèse doctorat, LISI/ENSMA et Université de Poitiers, 2006.
14. VASSILIADIS, Panos, SIMITSIS, Alkis, et SKIADOPOULOS, Spiros. Conceptual modelling for ETL processes. In : Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP. 2002. p. 14-21. [CrossRef]
15. YULIANTO, Ardhian Agung. Extract, Transform, Load (ETL) process in a distributed database academic data warehouse. APTIKOM Journal on Computer Science and Information Technologies, 2019, vol. 4, no 2, p. 61-68. [CrossRef]
16. Deng, W., Yang, X., Zhao, H., Lei, D., & Li, H. (2008, August). Study on EAI based on web services and SOA. In 2008 International Symposium on Electronic Commerce and Security (pp. 95-98). IEEE. [CrossRef]
17. GORKHALI, Anjee et XU, Li Da. Enterprise application integration in industrial integration: a literature review. Journal of Industrial Integration and Management, 2016, vol. 1, no 04, p. 1650014. [CrossRef]
18. MANOUVRIER, Bernard et MÉNARD, Laurent. Application Integration: EAI, B2B, BPM, and SOA. John Wiley & Sons, 2010.
19. BISWAL, Satyabrat, CHANDRASHEKHAR, J., SALAUDDEEN, Sk, et al. Enterprise Application Integration Based on Service Oriented Architecture & Enterprise Service Bus: A Review. International Journal of Advanced Networking and Applications, 2019, vol. 10, no SP 5, p. 116-119.
20. MASTERNAK, Tomasz, PSIUK, Marek, RADZISZOWSKI, Dominik, et al. ESB-modern SOA infrastructure. 2010.
21. BHADORIA, Robin Singh, CHAUDHARI, Narendra S., and TOMAR, Geetam Singh. The Performance Metric for Enterprise Service Bus (ESB) in SOA system: Theoretical underpinnings and empirical illustrations for information processing. Information Systems, 2017, vol. 65, p. 158-171. [CrossRef]
22. GOEL, Anurag. Enterprise integration: EAI vs. SOA vs. ESB. Infosys Technologies White Paper, 2006, vol. 87.
23. SUNYAEV, Ali. Cloud computing. In: Internet computing. Springer, Cham, 2020. p. 195-236. [CrossRef]
24. SCHULTE, Roy W. et NATIS, Yefim V. ' Service-oriented architectures, part 1. Gartner, SSA Research Note SPA-401-068, 1996.
25. Garcia, Cristiano, and Abilio, Ramon. Systems integration using web services, REST and SOAP: a practical report. Revista de Sistemas de Informação da FSMA, 2017, vol.1, no 19, p. 34-41.
26. Talend, Qu'est-ce que l'architecture orientée services ? [Online], URL: www.talend.com/fr/resources/architecture-orientee-services/, visited in December 2022

AUTHORS PROFILE

Mbaiossoum Bery Leoouro received his Ph.D. degree from the University of Poitiers in France in Computer Science and Applications in 2014. He is an associate professor at the University of Ndjamena, Chad. His primary research areas are data engineering, ontology, ITC and systems. He works on the physical design of databases, primarily using materialised views. He is interested in Ontology-Based Databases (OBDB). He proposed techniques to create an optimal materialised view on these databases. Bery Leouro studied how to build a smart school by suggesting a good practice for utilising Information and Communication Technology (ICT) in education. He noticed that ITC waste is not well managed in developing countries and explained how it could be managed. He is interested in the Semantic Web as well as in operating systems.