1. INTRODUCTION

Commonly used in distributed systems, the initial idea of SOA was able to create complex systems from simple separate parts (Kuba & Krajicek, 2007). But, SOA has some shortcomings that make it quite complex architecture. One problem of SOA is that, it generally lacks precision. Then, for some researchers, it is simply a collection of services that communicate with each other to meet certain tasks (Berry, 2012). For others, it is a way by which a provider is consolidating its expertise to meet the needs of a client (MacKenzie et al., 2006), etc. In fact, there are no unique definitions, solutions or approaches in the implementation of SOA. SOA is more an architectural paradigm that style (Nickull et al., 2007). We can talk about some element of SOA in term of a technology (web: SOAP, UDDI), however, when considering SOA like an architecture (represented also by other distributed technologies: CORBA, DCOM, etc.), it becomes more difficult to define the architectural elements (connectors, constraints, components, etc.). In fact, apart from web technologies (e.g., SOA) (Anderson,
which are well known, it seems, it does not exist precise specification (architectural elements) for the SOA style in general. SOA is rather perceived by companies as a way to support the business processes of the enterprise (Raymond, 2011). In short, it is very difficult to talk about SOA as an architectural style (architectural style is a set of principles; a coarse grained pattern that provides an abstract framework for a family of systems) because of the lacks of information or specification about elements in SOA architecture. However, there are some efforts trying to define SOA specification. OASIS (Organization for the Advancement of Structured Information Standards) for example has worked for six years on a SOA reference model. Their approach can be summarized in three objectives: establishing definitions, unifying concepts existing SOA, creating a clear semantics for modeling SOA solution. Despite this, the lack of precision of SOA is still a big challenge, because SOA has a varied nature.

At this first problem, we can talk about another problem concerning SOA. This is the problem of performance. This can be explained in the following way: the notion of abstraction and ease of creation of services within SOA lead to many small interactions between the different systems (applications) created, which dramatically affects the performance of the whole (Heubès, 2008). Another way to express this problem is the access time and availability (Crochet-Damais, 2007). Improve the access time and availability in a system is also a task or a challenge for architect. It’s the reason of that, apart from the need to answer the question of a definition of SOA (specification), our second objective is to provide tactical and architectural approaches to improve this performance problem.

We will focus on two important points, firstly, propose a specification (definition) to a better understanding of SOA and secondly to apply a technique to improve performance. For this second purpose, we will use the ADD method to support performance. Note that ADD is a method of creating architecture based on the quality attributes (Wojcik, 2006).

In this paper, we will talk about our two contributions. The first focuses on a specification or definition of SOA. The second offers in addition to the specification, a way to meet the performance (an extension to improve SOA model) using the ADD method. A series of case study is made to validate the performance of applications with ADD.

2. RELATED WORK

2.1. OASIS

Based on OASIS’s model, there are several layers in an SOA. The layers which are commonly used are: the service layer, the business layer and data layer.

- **Service Layer**: Services are exposed or resident at that level. This is the outsourcing company.
- **Business Layer**: This layer defines the business processes of the company.
- **Data Layer**: It is the ability to provide access to enterprise data that feed the layer or business processes.

Figure 1 is a representation of an SOA where you can see the different layers of this architecture.

2.2. IBM and Web Services

IBM, in its architecture, uses a stack of layers (layered architecture), composed of a layer of communication, message and process management. The communication layer handles the transport of messages via such means as the RPC (Remote Procedure Call), message layer (connectors) deals with the semantics of messages SOAP (Simple Object Access Protocol). In IBM’s model, the main format for describing a service is (Web Service Definition Language (WSDL). It also uses Universal Description, Discovery and Integration of Web Services (UDDI) for publishing and discovering services. IBM’s approach is specially designed to web services (Anderson, 2002).
Related Content

An Efficient QoS Routing Protocol for Real Time Applications in MANET
[www.igi-global.com/article/an-efficient-qos-routing-protocol-for-real-time-applications-in-manet/158966?camid=4v1a](www.igi-global.com/article/an-efficient-qos-routing-protocol-for-real-time-applications-in-manet/158966?camid=4v1a)

Internet of Things for a Smart Transportation System
[www.igi-global.com/article/internet-of-things-for-a-smart-transportation-system/216414?camid=4v1a](www.igi-global.com/article/internet-of-things-for-a-smart-transportation-system/216414?camid=4v1a)
Design Issues for Multi-Mode Multi-Standard Transceivers
[www.igi-global.com/chapter/design-issues-multi-mode-multi/62935?camid=4v1a](www.igi-global.com/chapter/design-issues-multi-mode-multi/62935?camid=4v1a)

OpenVX Integration Into the Visual Development Environment
[www.igi-global.com/article/openvx-integration-into-the-visual-development-environment/193620?camid=4v1a](www.igi-global.com/article/openvx-integration-into-the-visual-development-environment/193620?camid=4v1a)