ENTERPRISE SERVICES ARCHITECTURE IN THE WORLD OF INFORMATION TECHNOLOGY

Ph.D., Stefan **IOVAN,** Informatica Feroviara SA, Bucuresti, stefan.iovan@infofer.ro
Ph.D. Candidate, Gheorghe Iulian **DAIAN,** Informatica Feroviara SA, Cluj-Napoca, ghita.daian@infofer.ro

Abstract: *Enterprise Services Architecture (ESA) is blueprint for now enterprise software should be constructed to provide maximum business value. The challenge facing most companies is not whether to adopt Service-Oriented Architecture (SOA), but when and how to do so. There is always a lag between technological vision and business feasibility. It also takes time to fully realize the potential of existing technologies, a process that does not stop the moment the new thing arrives. But when the value of a new approach such as ESA starts to make a difference and produces a competitive advantage, the motivation to change skyrockets. The time to change becomes now and the hunger for learning grows. The goal of this paper is to satisfy the hunger for information for those who suspect that ESA may be a gateway to transforming Information Technology (IT) into a strategic weapon. This paper will explain – in more detail that ever before – what ESA is bringing the concept to life in all of its products as a platform supported by an ecosystem.*

Keywords: Enterprise Services Architecture, Service Oriented Architecture, users interface,

1. INTRODUCTION

The current state of the art is a long way from ESA. Most enterprise software programs now use Internet-inspired technologies, such as portals, web-based User Interfaces (UIs), application servers, and XML-based messaging services, but they still cling to client/server and even mainframe this will change dramatically over next six years. IT will become connected by networks, awash in data, faster, more adaptive, and more in sync with business. Companies that understand how to unlock the business value of this new architecture before their competitors do will have a huge advantage.

The skeptics among us cannot help but ask, "*Has something really changed?*" Buzzwords – web services, service-oriented architecture, and enterprise service bus are the current rage – come and go, but the network, the Internet, is here to stay. ESA represents a refactoring of the core architecture of enterprise applications to make sense of a flock of new possibilities and to bring them in formation to the level of business, application, and technology architectures.

IT will change not simply because new things are possible, but because most markets are presenting companies with a whole new set of requirements that traditional IT is having a hard time meeting. Most companies live in a world in which business models change every year, on even more frequently. An implementation cycle of a year or more an IT project can no longer be tolerated. New processes must be designed and built in three months, six months, or nine months. The systems of record that provide the context for most business activity have been built out. Now the challenge is to quickly a new layer of flexible processes based on those systems of record in a way that preserves flexibility so that future adjustments are affordable.

2. WHAT FORCES CREATED ESA?

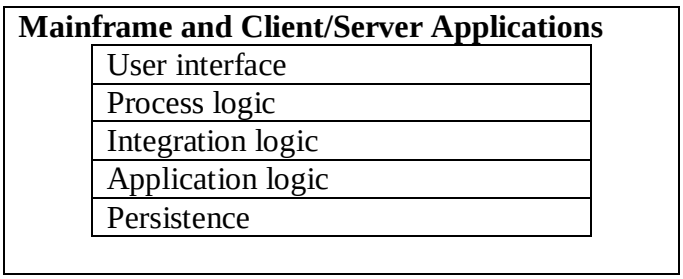
Modern businesses need functionality that is both distributed and centralized. Existing systems of record, such as Enterprise Resources Planning (ERP), Customer Relationship Management (CRM), Supply Chain Management (SCM), and so on, serve the needs of key segments of the organization. But at the same time, a need for many new processes has arisen that requires a flow that moves from one system of record to another, with the context for the process kept outside of any of the existing systems. The traditional way of building enterprise software is not well-suited to these new requirements and does not take full advantage of the new world of pervasive networks, reusable services, and distributed data. Treating an application as a self –contained world no longer meets the needs of business.

In the past, enterprise applications contained the end-to-end processes that were being automated. One program running on one computer automated a workflow process that began and ended inside application. A single database was the central mechanism of integration. All elements of the stack were contained within one program, as shown in Figure 1.

Figure 1 actually shows a prettier picture than what exists in many mainframe applications. Even after workflow mechanisms were in use and points of integration [1] were designed, process and integration logic ended up strewn all over the stack and was mixed in with application and UI logic. This structure, however, captures the spirit of mainframe applications, which at their best were organized into the following layers:

- The UI layer
- The process logic layer (which controls the automation of the steps)
- The integration logic layer (which controls the way the program interact)
- The application logic layer (which controls what the program is actually doing)
- The persistence layer that serves as the database (where all the information is stored).

Mainframe and client/server applications had complete control
of the stack from top to bottom.



Developers were able to control a vertical slice when coding from UI
to persistence. A single, consistent database was the point of integration.
Figure 1. *Mainframe and client/sever architecture*

From a development perspective, the mainframe and client/server tools gave developers control over a vertical slice of this stack, from UI to the persistence layer. If functionality in other slices was to be reused, the developer would have a conversation with the developers of the other slices to figure out how to use their functionality. Everything came together and had to be carefully reconciled in the database, which was the central point of integration. One of the major points of ESA is to transform such conversations about reuse from an ad hoc event into a formal design based on the needs of business processes.

It is possible to access the functionality in a mainframe/client/server stack through application programming interfaces (APIs) [4]. But there was no template for this; each time an API was developed, it came with its own assumptions about how it worked and how it should be used.

The mainframe/client/server applications did anticipate the need for customization through metadata, different variables controlling an application's behavior, and templates for UIs. But because the stack contained within one application, with the UI, process, application, integration, and information layers tightly coupled at design time, it was impossible to break open a mainframe application and restructure it to solve new problems.

3. APPLICATION PROLIFERATION

In the late 1980s and 1990s, ERP systems showed the power of the mainframe/ client/ server stack. Despite growing pains, the widespread success of ERP – with SAP leading the charge – led to the creation of other applications, as seen in Figure 2.

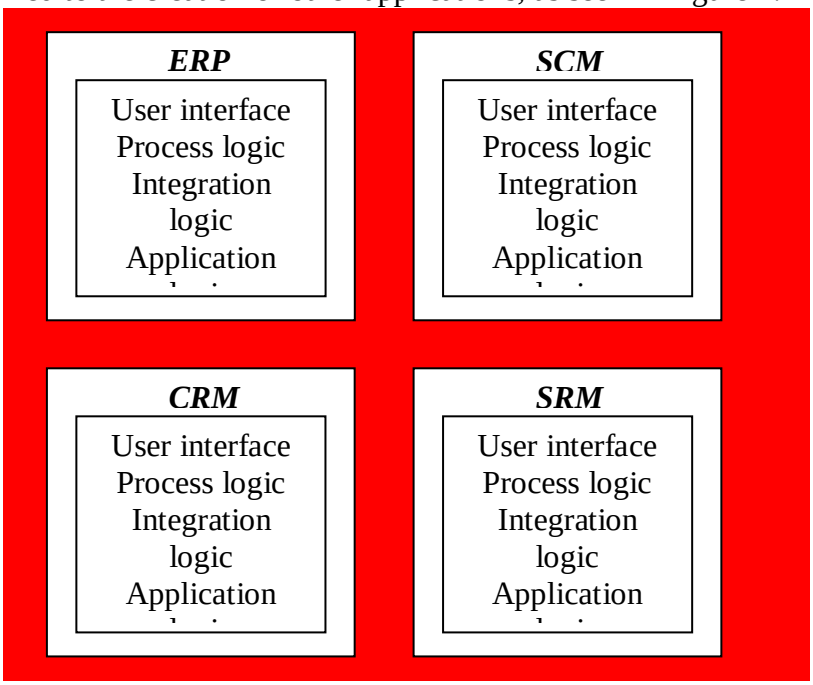


Figure 2. *Many applications, many vendors*

While ERP was focused primarily on only the financial and management aspects of a company – before it expanded throughout the 1990s to sales, distribution, on other key functions – new applications such as CRM, SCM, and supplier relationship management (SRM), among other expanded the range of automation. This led to a proliferation of applications for most companies under the label "*best of breed*". The idea was to get the best application for each purpose. This allowed the VP of manufacturing the best SCM application, and so on. The main benefit of this proliferation was the creation of a comprehensive collection of systems of record that automated common business processes from end to end.

But solutions from different vendors created a problem because they took away the central point of integration in the mainframe/ client/ server world: the single database. Data was scattered all over the system landscape, or even worse, was duplicated in multiple systems.

Communication and integration [2, 3] among applications became even more important when companies realized that essential processes may flow through several enterprise applications. The process that starts with taking and order and ends with the receipt of money, the so-called "*order to cash*" process, involved many enterprise applications. A financial transaction in the ERP system would move to the SCM system for a factory order, which then went to the CRM system for service questions, and then back to ERP for the final confirmation of the order. Getting it to work at all actually required expensive, hard-wired integration projects.

4. BRIDGING THE GAP AMONG SYSTEMS OF RECORD

The next challenge facing companies using enterprise applications was integration [5] by SOA. Hence enterprise SOA [7] is a new standard which allows integrating the functionality of existing SAP applications. How could all of the best-of-breed applications be made to work together to serve the needs of the cross-application process that were becoming the key to increased efficiency and innovation? As shown in Figure 3, the key question concerned how to bridge the gap among systems of record.

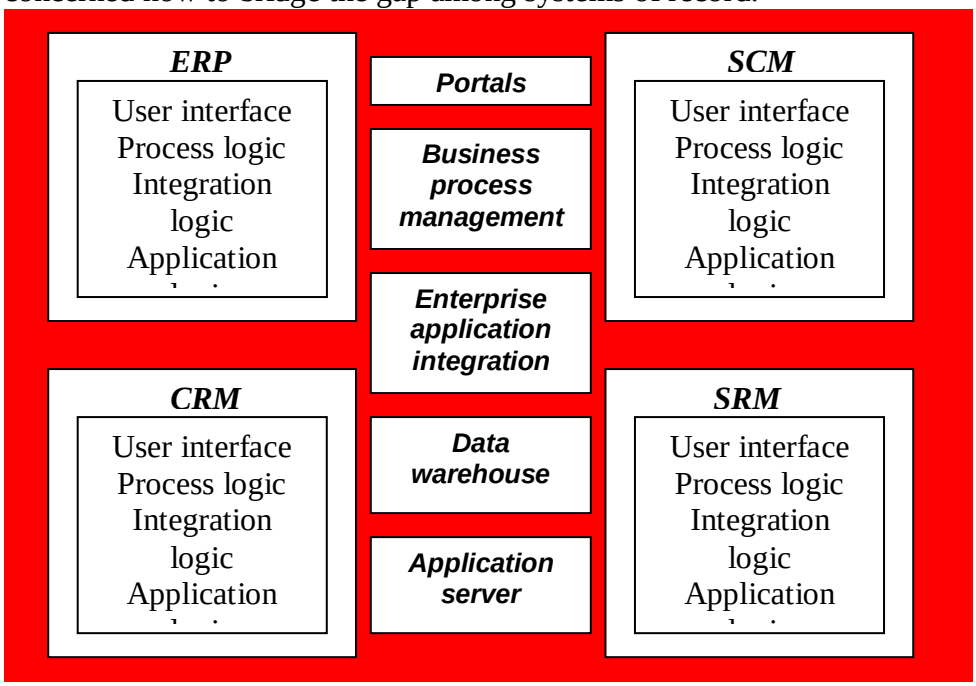


Figure 3. *New solution emerged to connect and unify the distributed enterprise applications.*

Many different technologies emerged to bridge the gap, so a cross-application, integrated view of enterprise applications was created, based on the new possibilities of the Internet as a pervasive network and emerging technology standards such as HTTP, HTML, Java and XML:

- Portals emerged as web-based UI technology that enabled one UI to connect to functionality from the different applications [6].
- Data warehouse collected data from all of the different databases within the applications in one place.
- Enterprise Application Integration (EAI) technology created engines that allowed one application to send XML message – a standard for data formatting – to another application, B2B [2, 3]. The receiving application could send a response back, and all sorts of fancy alerting, monitoring, and triggering could happen in central systems for routing and transforming messages.
- Business Process Management applications for process modeling and management were frequently coupled with EAI technologies to create a new way to define and execute processes in the center.
- Many of these integration tools were powered by application server, a new sort of structure for applications based on standards such as Java 2 Enterprise Edition (J2EE) that were created for the world of the Internet.

These new technologies started to bridge the gap among isolated enterprise applications and enabled some cross-application coordination and development. The results were encouraging. Portals could bring together UI elements from different applications, as well as gathering information from different sources and displaying them in one place. Data warehouse created one view of distributed information, albeit with a delay caused by batch-oriented extraction processes. EAI technologies connected applications, but these connections were complex and threatened a new layer of unstandardized spaghetti. Parts of the gap were bridged with these approaches and the requirements from cross-application processes were met to some extent. These capabilities fell far short of a unified approach to UI, process, and information integration, however. They also ushered in a new set of problems – integration of the integration technologies.

Portals might need to talk the data warehouse, which may need to send and receive data through the EAI system, which could be working with a Business Process Management system. The same sort of integration problems hat these technologies were designed to resolve among enterprise applications arose among the integration technologies, which also generally came from a variety of vendors. It was "*best of breed*" all over again, except this time; it concerned integration tools, not applications.

6. EXAMPLE: mySAP Business Suite and SAP NetWeaver

The cost of integrating enterprise applications and integrating integration technologies quickly mounted, leading customers to ask, "*Is this really our problem to solve?*" SAP thought not, and solved this problem in two ways. First, SAP assembled its own solutions for ERP, CRM, SCM, SRM, and so forth into a unified collection called the myAP Business Suite. Second, SAP integrated all of the integration technologies into a unified whole, called SAP NetWeaver. Furthermore, SAP started to develop all of its mySAP Business Suite applications using SAP NetWeaver: in other words, integrated applications built on integrated technologies as a platform.

This created the situation shown in Figure 4 in which an integrated set of tools could help manage processes across a set of enterprise applications designed to work together.

This approach solved a large portion in the problem of connecting enterprise applications and integration technologies to each other.

SAP NetWeaver allows you to write programs not only in ABAP – the language that has been used for more than 20 years to write applications in SAP – but also in Java [7]. This helps solve the problem of integrating the integration tools, but still the problem of getting all of these applications to talk to each other remains.

Bringing all of the applications together in the mySAP Business Suite helped solve the second half of the cross-application integration problem in a variety of ways. SAP was able to add business packages to configure enterprise applications to work together.

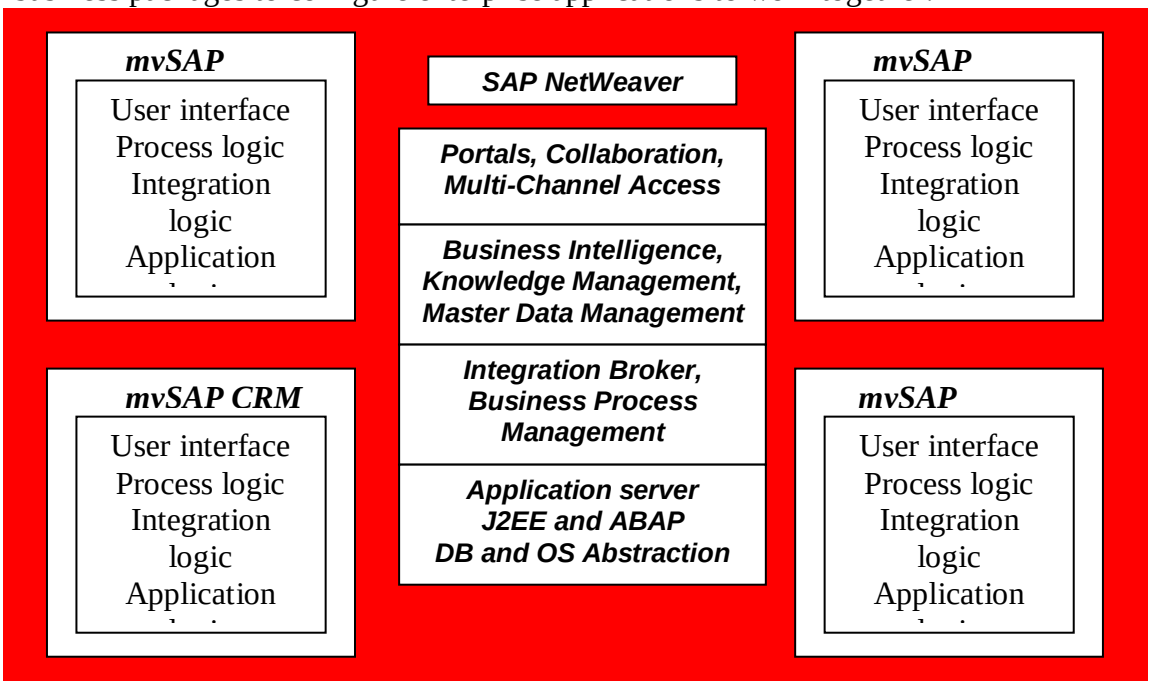


Figure 4. *SAP unified enterprise applications into mySAP Business Suite and combined integration components into SAP NetWeaver, which became a technology platform for the development of enterprise applications.*

The challenge still remained, however, to be able to recombine systems of record to solve new problems. The connections made possible by SAP NetWeaver allowed some processes to flow from one enterprise application to another, and solve a host of other problems as well. Much of the power of enterprise applications was still locked in the monolithic structure. Business needed to change faster than the connections between applications could be constructed.

7. REFERENCES

[1] A. Y. Halevy and et al. "*Enterprise information integration: successes, challenges and controversies*", In SIGMOD Conference, pg. 778-787, 2005.

[2] B. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, A. K. Elmagarmid, "*Business-to-business interactions: issues and enabling technologies*". The VLDB J., **12(1)**, pg. 59-85, 2003.

[3] D. J. Kim, M. Agrawal, B. Jayaraman, H. R. Rao, "*A comparison of B2B e-service solutions*", Commun. ACM, **46(12)**, pg. 317-324, 2003.
[4] Hristina Daskalova, Tatiana Atanassova, "*Integration Platforms – Problems and Possibilities*", Cybernetics And Information Technologies, Volume **8**, No 2, Institute of Information Technologies, 1113 Sofia, 2008.
[5] Haas, Laura. "*Beauty and the Beast: The Theory and Practice of Information Integration*", 11th International Conference on Database Theory (ICDT 2007), Barcelona, Spain, 2007.
[6] Daniel, F., Matera, M., Yu, J. Benatalla, B. Saint-Paul, R., Casati, F., "*Understanding UI Integration. A Survey of Problems, Technologies, and Opportunities*", IEEE Internet Computing, **11**, **3**, pg. 59-66, 2007.
[7] Robert Heidasch, "*Get ready for the next generation of SAP business applications based on the Enterprise Service-Oriented Architecture (Enterprise SOA)*", SAP Profesional Journal.