Ecosystem Oriented Architecture (EOA) vs. SOA

Pierfranco Ferronato
Soluta
http://www.soluta.com

Introduction

As James Moore [1] pointed out, a Business Ecosystem is based on a dynamic interaction of organizations which evolve over time in terms of capabilities and roles. To this extent this section will describe why the Service Oriented Architecture (SOA) is not adequate to face such challenges which are unique in the context of a Digital Ecosystem (DE). The author will highlight such differences and describe the features of a new architectural style, called the Ecosystem Oriented Architecture (EOA).

The chapter will explore the fact that an Ecosystem Oriented Architecture is not a “sort of SOA”, nor is it just a “bigger SOA”. A DE employ a broad set of digital components, such as: software services, business services, knowledge, representations of the economy, etc. Software services being just one of them. To explain the differences between EOA and SOA, this article will focus on software services. All the components interact together, forming a digital ecosystem. A whole set of new problems are to be addressed, namely responsive alignment with the business, decentralization, ownership of a distributed knowledge base, self-organisation and self-healing. EOA is a new mindset in decentralized architectures for Digital Ecosystems.

A Digital Ecosystem implementation needs to support a particular dynamic scenario where dynamic business service aggregations and evolutions are key. Neither are B2B market place solutions able to adequately tackle such challenges. DE has to “... exploit the dynamic interaction (with cooperation and competition) of several players in order to produce systemic results in terms of innovation and economic development”[2].

It is indeed true that the recent achievements in Business to Business (B2B) implementations are enabling enterprises to accelerate the dynamic of business, however these solutions are still limited because Service Oriented Architectures (SOA), the prime supporting architectural style of B2B, has been conceived for supporting a single value chain, in a single business domain and usually between a static set of participants; in fact it is often the implementation of a single-organization supply chain. We shall rather name the current implemented scenarios as “business to Business” (b2B) to enhance the fact that the structure is not “democratic”: in these cases there is a single Master in the supply chain, which is often the company that owns the chain. This company which is usually a large organization, can impose it’s standard to its suppliers and customers, and the ‘small’ players have to accept the specifications; it's not a peer based model, as the name B2B do inspires, the two ‘b’ have to be different: this is key. Such b2B environments are thought as being an example of an across enterprise implementation where interoperability is tackled and successfully implemented but -on the other hand- this is an oversimplification, since in reality supply chains do intersect and overlap.

B2B solutions are rarely applied outside the boundary of an enterprise and it is a challenging project: it is cumbersome, and especially complex to maintain. Digital Ecosystems are to be implemented applying a new perspective in Software Architecture that has to overcome the limitations of SOA: an Ecosystem Oriented Architecture (EOA). We intend to pin-point the fact that DE specific features and issues cannot be properly addressed by SOA; there is a
need to define a different architectural style that specifically tackles DE requirements from both the functional and structural viewpoint. Applying SOA when dealing with DE implementations overlooks the problems.

SOA has been conceived in the context of intra enterprise systems: in essence, the assumption is that any aspect either functional or structural is managed (or manageable) via a central governance entity. The infrastructure is under control and managed via a single department unit: network appliances like routers, firewalls, cables, routing and topology are planned and managed centrally. In addition, also the functional specifications of the SOA are planned in advance either in joint meetings between parties or defined by a single central authority. The WSDL representing the common technical contract for service invocation are defined up front and are to be used by all the partners in order for the value chain implementation to be effective: this is the environment in which SOA was born and where it is actually used most of the time. SOA is an architectural style that evolved from EAI, RPC and CORBA where the focus was on Applications, Procedures, Objects; focus on services was added later but still with an “intra enterprise” mindset (Figure 2 below).

An SOA implementation is often conceived, funded and implemented by an organization with the sole goal of supporting and increasing its business, as a consequence this drives the entire environment which is single-party centred and does not follow the competition/evolution core feature of a DE.

In an enabling ICT-based infrastructure aiming at supporting the economic activity of networks of business clusters (or business ecosystems) fostering systemic synergies with special focus to SMEs. DE scenarios are changing the rules, because the focus is moving from “intra enterprise” to “across enterprises” (inter community) and soon “across communities”. Using SOA for implementing a DE, that requires enlarging the participants in a broader spectrum, supporting a wider set of functional models, running over the Internet, spanning a WAN, is underestimating the problem. As a matter of fact, reading the literature[3], and from the author's experience, it is evident that dynamism and flexibility are key for running a Business.

In a digital ecosystem the value chains are overlapping, they are not partitioned but intersect each other;

- the social and business network topologies are not hierarchical [4];
- a single functional reference model cannot be implemented;
- there is no single point of management from both the business and structural viewpoints.

Taking the previous premises into account, the final goal should be the integration of the services offered by each of these SMEs, without involving extra investments in items not related to their businesses (such as information systems). Therefore, the system should be operated automatically without human intervention. In other words, the system that supports integration of the aforementioned services should have self-organising capabilities.

On the other hand, it should be decided what would be the minimum infrastructure required to allow the presence on the Internet (that is: great portals, operators, ISPs, etc) of these businesses without the need for great investments, or great resources. To reach this goal, two clear premises were identified: minimum hardware, and zero maintenance. The need for maintenance and administration should be eliminated, wiping off the greatest source of cost. This now opens the challenge of zero-administration, which requires the development of software technology that provided self-organization mechanisms.
Along with these elements, the system should bring us the possibility to publish the presence of a given business (identified as a service) from the moment it's connected to the DBE network, and the ability to detect its disconnection to eliminate the service from all the contents in which it was referenced.

1. Functional Reference Model

Digital Ecosystems cross business domains and different value chains, for this reason they are characterized by not having a single functional reference model. Since it is not feasible to define up front all the required functional models, which are intricate, complex and continuously changing, the ecosystems participants need to be free to define, publish and use any models that they consider adequate for their business.

As an example, a book distributor or reseller might create a model that represents their application interface to allow consumers to search, browser, order and buy books. This model could be published and implemented by their service component. Other competitors in the ecosystem will probably do the same in autonomy and this will end up with a set of different APIs that would burden the effort of a bookstore when required to automate the order process; for each supplier/distributor a different technical adapter is required. This constraint would slow down the rate of adoption and lock stores on a single supplier because of the effort required to align the software again. This would represent the dead end of the ecosystem; without fast business alignment, there will be no evolution.

One rather ingenious approach to overcome this issue is to have all the book distributors sit around a table in an association defining “The” reference model for the book store sector. From direct experience of the author(2), this is a method that does not scale for a long time and, assuming that the participants are able to converge to a suitable model, there will soon be other “competing standards” (notice the oxymoron) that would again create interoperability problems.

Also, maintaining the specification would be very time consuming and in the end it would not be possible to keep it aligned with the business requirements: new features driven by the end users or marketing would incur the risk of being left behind, waiting for the new specification to emerge or -even worse- of being implemented diverging from the standard. As a consequence, the expected well ordered mechanism would soon break.

This scenario is a gross over-simplification of the models what might be found in a DE, especially considering cross value and supply chains. The overall map of models would be so complex and articulated that managing them would be impossible. As a comparison, we can recall the Internet map(3)[5] and its topology; no-one can have full control of it. It emerges rather autonomously from complex usage mechanisms that have been investigated only in recent times. Even maintaining the functional models of a complex ERP project, with well-defined boundaries and dependencies, can be very difficult and impossible for a single party; changes and updates are often tough tasks to accomplish. In a business ecosystem this effort cannot be addressed at all, and a new mindset and approach in this sense is required, and the SOA approach is hence inadequate. In addition, assuming that an ecosystem can be managed is a contradiction in terms. The keyword is “self-regulation”, “self-adaptation”[6] and the EOA has to implement the required instruments for this to happen, it is useless to fight and oppose the dynamic nature of a DE, it is better to support it.

The way to go about then assumes the inability to control the reference models; we might assert that there is no reference model at all, and take all the required architectural decisions to support it and let the ecosystem converge, dependant on time, in a model. What is
fundamental to assume when defining the architecture of a DE is to recall that it is a highly dynamic environment where the IT related frictions and inertias needs to be reduced to the minimum. This is the prime condition that will allow an ecosystem to self-converge and adapt.

The architecture needs a mechanism to allow participants to:

• publish any model;
• investigate which is the most adequate to their needs;
• adopt it (and change it) in a totally free and uncontrolled space (regulatory and restrictive features shall only be added as a means to avoid hacking or spamming the environment).

A structured and highly connected repository has to manage the models, their dependencies and their association with implementing services.

As an example: if the book distributor could inspect the ecosystem (specifically using a model repository), it could detect that there is a functional model for the book sector that is adopted by 75% of bookstores and another one less adopted (hence less connected) but closer to its technical needs and more straightforward to implement due to the better alignment with their back-end systems. The distributor has the chance to decide whether to adopt the most connected model, hence facilitating the migration and adoption by bookstores, or to stick to the easy way with an obvious drawback regarding the level of adoption. In this scenario it is evident that bookstores (the service consumers) on the other hand will try to reduce the number of different models in order to lower their integration efforts and favour the quality of the service offered. The balance between the symmetric aspects is the basis for competition and evolution.

2. Model repository

In SOA, UDDI is the catalogue of services and service models. They are mixed with binding information, there is no separation between the technical specification and the functional one, and in addition the service end-point is also written in the service specification. Such structure is a consequence of the fact that UDDI has been conceived as a static catalogue of intranet services(4); it is clearly a consequence of the fact that it descends from classical RPC approaches. UDDI is essentially a catalogue of programmatic resources.

For example: two different book distributors might use the same technical specification of the service (e.g. WSDL) but have different kinds of discount policies, different return policies, different quantity discounts or serve different regions. The WSDL is a technical specification that exposes the service protocol that in turns implements the business service. What has to be modelled and delivered is the business service rather then the mediator to the service. In an SOA the need to model the business specification is not a prime need because there is no economical transaction involved. SOA is often implemented, in the author's experience, in a context where the associated business transaction costs are null (zero). Nevertheless, the writer is aware of some SOA implementations (rather tough though) in which an invocation implies an effective business transaction, i.e. some “money exchange”. But also in these cases the participants and the services involved have been defined up-front -statically- and the business models are known in advance: there is no dynamic discovery or negotiation and for this reason -under these assumptions- SOA works fine; in DE on the other hand it would not scale. Reference documentation about UDDI mentions “Companies can establish a structured and standardized way to describe and discover services”(5), but a DE is not a structured or standardized environment.
In a DE, the model repository needs to manage business models instead of programmatic specifications. OMG's XMI is the prime choice for encoding models because it is a platform-independent specification; it supports meta-modelling, model dependency, merging, inclusion, inheritance and versioning. XMI is able to represent semantically rich model specifications, where WSDL is not. Services in DE need to make use of more complex specifications, the definition of software interfaces is not sufficient: there is the need to express the underlying business model. The plain interface specification is not relevant in the context of an ecosystem where services need to be explored automatically via recommendation agents: having computable business models is essential.

In addition, the functionalities provided by the repository need to support an enormous amount of unstructured and related information. The users, either a software component or a human being, must be able to navigate the intricacy of models and their dependencies in order to identify those that are most useful and adequate. In this sense the repository needs to provide intelligent and semantically aware research and recommendation tools[7].

It is also essential is to decouple the service model catalogue from the actual service instance catalogue: “The service registry”.

3. Service Registry

The service registry contains the references to actual services published in a DE associated with the technical and business models. Each entry includes self-contained information about the service (called Service Manifest[8]), made of:

- service business models;
- technical specification (i.e. Service APIs);
- business data;
- service end-point.

- The first type is essentially the business specification (it might be a reference to an entry in the model repository, this is an implementation aspect which is not relevant in this context).
- The second is the technical specification of the service.
- The third is information specific to the service instance, for example the name of the published service or the location of the service; in general this information is associated with the business model.
- The fourth is programmatic information needed to actually invoke the service, for example – it is an over simplification – the IP address and the protocol used.

Whatever way this registry is implemented, the essential aspect is that is has to be extremely dynamic and bind to the actual published service. In SOA it is a great frustration to try to invoke services from information found in the UDDI just to discover that they are not available. The real issue in these cases is that the requesting service is not able to provide the reason for the failure: is it due to the fact that it has been discontinued or because there are some temporary technical issues? In an intranet SOA implementation, the architect has the ability to put all the efforts in order to have a high availability of service: in the Internet this cannot be assured. As a solution, the service entry in the registry needs to be bound with the actual remote published service so that it provides up-to-date status information; since it is too administratively intense to manually keep it aligned, a lease base mechanism is a good technical approach, like SUN’s Jini(6) framework dynamic lease management or the FADA framework (7).
As for the model repository, the service registry needs to be MOF8 compliant in order to ease the issues related to model interoperability.

The model repository and service registry represent a single point of failure (SPoF) for the DE architecture and this can jeopardize the entire ecosystem. This issue is addressed via a decentralized architecture, described in section 5 “Single Point of Failure” Chapter.

4. Basic Services

An architecture for DE needs to consider a set of basic business services to support the ecosystems and facilitate the rapid and correct interaction between business services. A DE without a proper set of basic services is unlikely to be sustainable: the goal is to improve the level of adoption by easing the participants' effort in publishing and integrating services. It is fundamental for example to execute a negotiation process before actually consuming the service (which is not required in an SOA implementation, as mentioned above) essentially because a service invocation in a DE is a business service consumption. For the same reasons, services such as reputation and trust are as fundamental in a DE.

The following services are needed essential to facilitate the bootstrapping phase in a DE:

- Payment
- Business Contract & Negotiations
- Information Carriers
- Billing
- Trust
- Reputation
- Legal compatibility

It is however important to underline that all these services do not specifically need to be implemented up front. It is important to support them for example by defining their models in the repository and providing an adequate infrastructure for their implementation: it might be up to participants and organizations to implement them. But some, like the accounting service, need to be supported by the core infrastructure of a DE because it has to adequately intercept the inter services messages.

One of the most significant services required in a DE is support for negotiations. In SOA, in those rare case in which it is implemented in across-enterprise B2B environments, negotiation takes place outside of the IT systems, often through real meetings; in SOA implementations, only the service execution is supported together with a poor search mechanism. In DE, following the definition given at the beginning, the ecosystem is such only if the integration mechanisms are fast and automated. As a matter of fact DE had to replicate in an e-environment what happens in the real world environment.
In addition there is the need to reconsider other services, although in a different perspective:

- Service Discovery
- Reliability-guaranteed delivery
- Security
- Long running Transactions
- XML Firewall.

5. **Single Point of Failure**

The Service registry is a key element for SOA; it is used at run time for service discovery and invocation, for this reason it represents a single point of failure for the entire architecture. If the registry is not available, the services will not be reachable.

This is a key issue also in SOA, for this reason UDDI version 3 has introduced replication schema for cluster of registries that provides high availability feature[10]. It supports both clustering and mirroring, however replications are based on the complete mirroring of nodes; in addition the replication policy is to be accurately planned by an administrator and implemented beforehand. But for a DE, given the complexity and intricacy of the infrastructure, the very frequent changes and the absence of any “root” node, this solution is not adequate.

In DE, the registry is even more critical because service IP addresses change very often, while in a classical SOA all the services are published in static IPs and change quite seldom: caching IPs would not work for long[11].

Setting up a single central fail-safe and highly redundant registry server would be very expensive and would not even guarantee service continuity in case of natural
disaster. The alternative solution is to exploit decentralized approaches, i.e. a topology and replication schema that does not make the DE dependant on a single node but rather on a collaborative set of peer nodes (more on this in the next section “Scale free networks”). Instead of a controlled cluster of nodes, there is the need to advocate the use of peer-to-peer networks as the routing infrastructure that improves routing resilience to node failure and attacks on service registries[16]. Such a network of nodes needs to be self healing and self adaptable to the ever changing nature of the requests and traffic: there should not be an administrator. Such kind of solutions would be resilient to node failures and would not loose information under critical circumstances. Nodes within this network interact in rich and complex ways, greatly stressing traditional approaches to name service, routing, information replication and links.

In such types of networks, data replication within nodes takes place intelligently: entries migrate automatically in relation to requests, moving data toward nodes that started the request. In this way, as in typical caching mechanisms, information is copied from the closest nodes so as to increase the probability that sequential requests get fulfilled in less time. It is relevant to notice that “close” in this context is relative to speed and not to geographical distance, since often in Internet hub nodes 100 km apart are faster to ping then local servers. Moreover, such a copying mechanism replicates redundant information among nodes so as to increase tolerance in case of nodes failure. As a matter of fact the new Italian Health Care System is adopting such a decentralized architecture for the Patient Health Record registry[12].

Avoiding having single points of failure for an EOA is essential. Beside the technical non marginal aspect of having a more reliable system, the DE will not suffer from the “big brother syndrome”. With a decentralized P2P based architecture the knowledge which is held in by the model repository and the service registry is not managed by a single institution which could tamper with it at the expense of the community by imposing unwanted control. A DE is self-regulated and self adaptable by definition[13] and a central institution with the potential power to control the environment from a technical and functional point of view could hinder the entire process of adoption and sustainability. Consider for example what would happen in case the organization hosting the service registry decided to shut it down. Such possibility would impede the adoption of the DE.

DE founds its entire sustainability and existence on knowledge about models and services. Participants in the DE are providing and using models while actively participating and being part of a business community, they are hence scared about loosing models. The owners of DE knowledge need to be the community itself, to this extent a peer-to-peer network (see next section “Scale Free Network) is a good approach because it is democratic; it provides participants with the possibility to offer resources to host part of this knowledge.

The significant drawback is the implementation: such a peer-to-peer infrastructure needs to be self-healing and self-adaptable. But there are already some frameworks and tools that support the enhancement of the properties of Scale Free networks.
6. Scale Free Networks

Most of the solutions in SOA, like the cluster of UDDI registries, are based on hierarchical structures because this is the way humans proceed in order to deal with complexity, i.e. in order to create comprehensible models. But as a matter of fact, the social and business networks in the real world are not hierarchical at all: this is essentially the reason why information models become more and more unmanageable with the increase in complexity. The more the IT systems push in the direction of being aligned with the business, the more the IT becomes unmanageable. Below a certain degree of complexity, any model can be reduced to a hierarchy that represents a good approximation, but with the increase in complexity it becomes impossible to stick to a hierarchy because reality is not as simply structured: it is based on different models and topologies: Scale Free networks[15].

The scale free networks are well described in the literature[14], we do not intend to describe it in this paper; what we state is that since scale free networks are the topology at the basis of business and social networks[15], a proper EOA has to support it and define appropriate mechanisms in order to let it emerge in a self organized way without human intervention.

In order for a Scale Free Network to emerge, it is necessary to support connectivity, proximity and preference[16]; it is dangerous and it represents a risk in the architecture to over-impose an unnatural topology. The advantage of a Scale Free Network is well described in the literature, essentially it is tolerant to a random failure of nodes and the properties of a “small world” allow efficient searches[17][18].

The author envisages a service registry and a model repository implementation that take advantage of such kind of networks essentially because this is the way they exist in the real world and supporting this vision will help align the ecosystem with the business -as is required.

Technologies are already available and they make use of concepts like the Tuple Space or the Distributed Hash Table, for example Sun’s Jini™ Network Technology[10], FADA[11], Bamboo[12], Cord[13] and others; there are also commercial implementations like GigaSpaces©[14]. P2P architecture can help, even if they can be used to infringe copyright: there is no need to be prejudiced, a technology is not bad per se, but it depends on the way it is used. The Digital Business Ecosystem (DBE)[15] has made a significant step forward in this direction.

Conclusion

Service Oriented Architectures (SOA) do not scale nor address the new challenges addressed by the architectures for Digital Ecosystems. The author envisions a new architectural style, called the Ecosystem Oriented Architecture (EOA). Three levels of service specifications are to be identified and addressed[20]:

- service models: a catalogue of business and computational models to be reused;
- service implementation: a catalogue of services descriptions (Service Manifest) implementing some models together with their data;
- service instances: service name and endpoint to actually invoke and consume a service.

In DE it is essential to have a repository of models separated from the registry of services[20]. The model repository needs a whole set of discovery features and supports XMI in order to implement model driven capabilities like dependency, versioning, merging and inheritance. Services need to be described also from the business viewpoint: the
computational specification is not sufficient in DE because services are not known in advance and the discovery process needs to be smarter and based on business specifications. The service registry needs to overcome the static limitation of UDD-like services and be dynamically bound to actual published services. In the near future a lot of mobile services are expected and these devices are going to make use of dynamic IPs, enhancing SOA based approaches is not enough. The service instances are to be resolved at run-time via a sort of DNS service.

Given the nature of a DE, the architecture needs to avoid single points of failure, the best approach envisioned is to make use of P2P technology to implement a decentralized data storage system (as opposed to the SOA centralized or distributed approach).

Basic services need to be implemented and defined up front in order to sustain the ecosystem, such as negotiation, information carriers, payments, accounting, billing and others. While SOA essentially supports only the service execution phase, a DE has to support the entire business service life-cycle including service selection (as opposed to service search), negotiation, agreement, contract specification, consumption and delivery.

In any aspect, either functional, structural or topological we have to reflect the real ecosystem in the DE: after over 40 years we realize that we are still applying the Conway[16] law that states “Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations”[21], i.e. any piece of software reflects the organizational structure that produced it, and a DE is no different.

References


[15] Albert-Laszlo Barabasi, Reka Albert, Hawoong Jeong, “Mean-field theory for scale-free random networks”, Department of Physics, University of Notre-Dame, Notre-Dame, IN 46556, USA


