Infrastructures for Digital Business Ecosystems : the wrong question ?

Maurizio De Cecco

http://maurizio.dececco.name/

http://www.linkedin.com/in/mauriziodececco

As an independent expert working for the Digital Ecosystems unit I had the the occasion to have an external view, with a different degree of details, of many of the projects member of the Digital Business Ecosystem cluster. I had also the opportunity to be reviewer of the DBE Integrated Project, that set to the context for the cluster.

This unusual external knowledgeable position allows to formulate an analysis that is not tied to the single reality of each project, but that try to start from a global point of view, starting from the basic idea of a digital ecosystem, that is to be an self sustained open system without single point of failures, from the technical and from the organizational point of view.

The result of this analysis is a fundamental critic, not to each of the project, each of them having a perfectly coherent point of view and strategy, but on the collective set of results as a way to actually implement a real Digital Business Ecosystem.

In the following I'll try to detail my analysis, and try to provide a strategy for a solution of the problems identified.

The Big ESB syndrome

Most of the projects and proposal i have seen in this area, includin g DBE and also in general many of the projects in the Networked Enterprise area, suffer from a basic drawback, if analyzed as a enabling technology for Digital Business Ecosystems.

This drawback have two faces, technical and organizational, that i call the Big Enterprise Service Bus Syndrome, and the Big B2B syndrome.

In synthesis, both aspects can be resumed as: there is a barrier around the project, and either you are inside the barrier, or you are outside the barrier.

Technically, each project defined its own basic technical infrastructure; this infrastructure constitute the network connecting the system user together and require (some class of) users to deploy this infrastructure. The infrastructure implement the connection, mediate the communication and offer a number of services like security, distributed storage etc.

In this sense, these infrastructures are like Enterprise Service Buses (Esb), overgrown and applied to a distributed environment; this explain the name Big Esb Syndrome.

The existence of such technical infrastructure have an implication on the organization behind: the infrastructure must be maintained, promoted, deployed; this naturally bring toward a business model where the system is managed by a kind of global supervision organization, that may be a foundation, an association, and so on.

The existence of this structure bring naturally to consider the users of the system as “customer” of the system; this bring the vision of an over sized and distributed Business to Business system, and the relative business model. That is what I call the ig B2B syndrome.

In the following i'll discuss why the Big Esb Syndrome is in contradiction with the digital business ecosystems objectives and why it represent a big danger for the success of the project cluster; finally I give my opinion about which alternatives should be found.

I will not discuss the Big B2B Syndrome, that I also think is in contradiction with the Digital Business Ecosystem idea, but that is fundamentally outside the scope of my core competencies.

The Big EBS Syndrome

Or why depending on a specific infrastructure is a bad idea

The infrastructure approach is a single point of failure

We are not concerned here with the technical point of view, but with the from an historical and organizational point of view.

Each project defined its architecture in term of a specific implementations and code base, this being Fada, SSRN, Soapsod; or other; the perennity of a deployment of such system is then strictly tied to the perennity of the code base.

The perennity of the code base (and not of the ideas and concepts behind the project) is a very diffult objective to achieve in an IST project, and even the open source dissemination strategy put in place by many projects offer no guarantee of survival of the code base. For example, the complex DBE infrastructure did not survived as code base.

This is even more true for those projects that are application oriented and essentially infrastructure users and not a technology oriented project; these projects do not have the resources even to try to make the infrastructure viable on the long term.

The infrastructure approach do not produce an open system

Even if the infrastructure reach a reasonable perennity, it is still a non open system, even if the code base is free software.

This means that the internal behaviour of the system is not defined by a well defined set of rules and protocols that are handled and managed as such; even if the internals are documented and published (as is for example in Seamless, but not in Fada), there is no commitment to these internals, there is no guarantee that this internals are stable, because no project see these internals as an important aspect of the system.

The results is that is essentially not possible to have an independent implementation of a system node by a third party; for example, implementing a Fada node or a Seamless node from a completely different code base, and having an independent evolution of them is not possible.

Essentially, the infrastructure introduce a form vendor locking, even if the “vendor” is an open source community; this form of vendor locking essentially prevent the development of the

technology from a set of independent actors, and so again the perennity of the system and of any deployment is limited to the perennity of the original code base.

The infrastructure approach make generalization difficult

There are a number of examples in the open source community where a system that was not open and that implied a kind of vendor locking became anyway a largely deployed de-facto standard. The most interesting case is the Perl language, for which no formal or informal specification exists other than the implementation itself.

It is very unlikely this could happens for a technology that is an research oriented niche technology that is the phase of seeding the development of new market; it may happens later, to whichever technology is largely adopted during a potential mass adoption phase of DBE.

The infrastructure approach, and the lack of stable, standardized and documented internals, make very difficult that a technical ecosystems develop around the technology, like it happended for example for technologies like usenet and internet mails, and most recently for jabber/xmpp.

It is then very unlikely that the a single project infrastructure can get a wide adoption and a generalization to assure its perennity, expecially considering that these infrastructure are not even shared between projects.

The infrastructure approach make reuse difficult

Passing the infrastructure code from a project to another proved difficult: the perennity of the code is not guaranteed, and often its development stop after the funded period, and even if it is continued there is no compelling reason for a project to reuse a given infrastructure code base; on the contrary, an existing non maintained code base is a liability and a risk for a projet.

As a result, the projects reuse the ideas and the concepts (like ONE w.r.t. DBE), but since the ideas and concepts are instantiated in terms of code and not in terms of open specifications, the resulting projects are completely incompatible (even if they can be interoperable, but that is another subject).

The infrastructure approach make impossible the establishment of a common technical DBE platform

The previous discussion should clearly show that an infrastructure approach do not allows for the construction of an integrated platform between the projects, expecially at the research level prototype.

The running integration experiments (like between ONE and SEAMLESS) are at the interoperability level of traditional EAI solutions, with ad hoc implementation and translations between the systems and not in terms of natural sharing, at the infrastructure level, of common services.

SOA and Protocols, a possible solution

In an environment where a set of projects have their own agenda and contractual constraint, a deep technical integration between projects represent again a risk and a liability for the projects. In general, the choice is to implement some kind of semantic interoperabiltiy by building bridges between the different components.

Of course, this is a sound strategy for each project, but the result have nothing to do with an actual Digital Business Ecosystem; it is very similar to the traditional EAI (Enterprise Application Integration) situation where a number of legacy applications are connected together using a number of bridge technologies.

Often, the Service Oriented Architecture is promoted as a solution of this problem; but by itself, an interconnection based on Web Services instead of other more traditional technologies (Corba or RMI) doesn't really change the situation, as is clearly shown by many ESB systems that are just good old EAI traditional systems with the transport layer based on Web Service technologies.

The complexity of such systems grow with time; the access barrier become quickly unacceptable to new users, and the exploitation of such a set of infrastructures become chaotic in the best case.

The point of SOA is not actually the trasport technology used, but the design philosophy, that is not limited to solving an interoperability problem between different infrastructure, but in designing the rules the different services are defined and composed to allow the construction of a global environment (the ecosystem) from the composition of a set of simple unitary services. The infrastructure do not exists any more as such, but only the rules and the services defined following this rule.

I find that the terminology around SOA is a little bit overcharged, and suffer from a kind of buzz effect; this is why in the following a talk more about protocols, that is formal definition of how things works together.

Protocols

It should be noted that a protocol is not just a documented and open interface of a service.
It is more about decoupling the life cycle of the interface from the life cycle implementation, having a commitment to maintain and evolve this interface through a public, open and documented process. The management of this process and protocol should be performed by an open entity that is not the owner of the protocol implementations.

Protocol are evaluated for coherence with respect to existing protocols, and are a much more valuable asset of a code base, because have bigger probability of being reused, and to survive the project end, not being a liability, but a prepackaged set of solutions to known problems.

Being oriented to protocol reuse do not prevent code from being reused. Typically a protocol is by a reusable reference implementations, if the conditions for globally maintaining this reference implementation exists.

The user-to-infrastructure interface must be a protocol

For start, the external services provided by the infrastructure to client applications should be defined as protocols, in the above sense; I.e. providing the list of web services implemented by the infrastructure is not enough, if the definition and life cycle of these web services is not clearly separated from the implementation of the infrastructure.

This would reduce the risk for an application developers coming from the dependency on the single vendor infrastructure; the basic or applicative service can be potentially provided by other infrastructure and projects, raising the reusability of an application.

The internal infrastructure interfaces must be a protocol

This is the key point: by dividing the infrastructure in a set of distributed components communicating though public protocols, the infrastructure is not an infrastructure anymore, but it move toward an ecosystems of interacting components, each of them can have its own set of actors or management rule behind.

This is very different of providing a set of web services to access the infrastructure internals: the point is that the infrastructure itself should be seen as the result of the collaboration of a number of independent components, each of it having its behavior defined by a set of publicly defined protocols. And it is different from having simply a SOA architecture, because of the requirement on the way the life cycle of the protocols is defined.

The probability that an existing infrastructure can be changed to use open protocols internally is very low; for example, an existing infrastructure can use implementation dependent data type in its communication, or assume the respect of implicit global consistency rules. Most of the time, allowing public protocols for the internals needs an initial design that is based on the concept.

This design must adopt an approach based on loosely interacting components more than a distributed implementation of a specific system; this idea is after all the basic idea of Digital Ecosystems.

Composing Services

Designing functionalities as the results of the collaboration of distributed components thru well defined protocols is more complex than designing it as a centralized module.

For example, many projects needs a kind of knowledge base. The implementation is usually distributed or P2P, to provide some kind of fault tolerant, but it still seen by the user component as a monolithic and centralized service, where each knowledge provider must store its data, and to which all knowledge consumer must subscribe.

Imagine instead a globally distributed architecture where each knowledge provider provide locally its knowledge by standard protocols, for example http and REST, and knowledge consumer access the knowledge thru the services of a set of independently implemented knowledge crawler.

This organization resemble more an open market, where independent entities interact to provide a global result. No global single point of failure exists, even if each single node can still fail to provide its local knowledge.

By the way, this kind of architecture is not at all new or innovative, if you consider that is the standard way searching information on Internet works (local publication, search thru a number of independents search engines).

Build on the past experience

The existing set of protocols have a internal logic that have been created in many years of works and mediation between many actors. This set of protocols, the process to handle them, their genericity and open ness is what made Internet what it is now, and should be considered as the basis to build any Digital Ecosystem, that should be seen as an evolution of the current Internet more than a alternative to it.

For a new set of protocols to be credible, the set of protocols defined should integrate with the existing protocol system and should reuse existing protocols natively without redefining equivalents protocols (thing that is natural in an infrastructure based approach); for example, document management should be done using WebDav, and not by defining a project protocol for document management; the same go on for identity management, security, etc.

But the main point is that the philosophy behind the protocols should be aligned with the current practices, and the new protocols and services defined for the DBE construction should be conceptually seen as an extension of the existing set of protocols; when possible, this approach should include working with normalization organizations like W3C or Oasys to define the DBE protocols as generic protocols.

Not reinventing the wheel for protocols should also maximize the possibility of reusing existing software; for example, if the document management is done with webdav, any existing application using webdav can access or manage the documents; the rebuild-the-world syndrome could be avoided, and the resources used on what give a specific added value to DBEs.

And now what ?

Even if the point of view exposed in this article become widely shared, it is not easy to move from the infrastructure based approach to a real protocol oriented DBE. The adoption of a coordinate, coherent set of protocols and architectural choices shared across the boundaries of different R&Ds and user organizations cannot be achieved without having the proper cooperation structures in places.

For this reason, the first step should probably be the creation of an entity (modeled around existing organizations like W3C, JCP, Oasys etc.) that would manage the protocol definition and the evolution of the common view of the DBE. In order to reduce funding needs, this organization should be lightweight and based on work contributed from the participating organizations and companies.

The creation of this organization would then allows a broad discussion on a real, community wide action plan, and of a set of common strategies for evolving toward a common objective, by encouraging new projects to adopt the existing standards and helping existing project to migrate to the defined standards within their possibilities.

This organization would also be in the ideal position for encouraging the sharing of open source software implementing the defined protocols, and the creation of an open source DBE community by providing standard development support infrastructure in the form of a sourceforge like service.

The architecture problem should not be underestimated; interoperability wrapper is an easy path to take instead of designing a global architecture based on open ended collaboration, but represent a cul-de-sac that if applied on a large scale would make impossible the adoption of real digital business ecosystems.