Architecture for Digital Ecosystems, beyond Service Oriented Architecture (IEEE-DEST 2007)

Pierfranco FERRONATO[1], Dr., Chief Architect of Digital Business Ecosystem (DBE) project

[1]Pierfranco Ferronato, Dr., Chief Architect, Soluta.net, Italy, EU, e-mail : pferronato@soluta.net

***Abstract*— A Business Ecosystem is a term introduced by James Moore[1] where in "The Death of Competition: Leadership and Strategy in the Age of Business Ecosystems"[1] wrote on page 26 that a Business Ecosystem is based on dynamic interaction of organizations which evolves over time their capabilities and roles. To this extent this paper will tell that Service Oriented Architecture (SOA) is not adequate to face such challenges which are unique in the context of a Digital Ecosystem (DES). The author will highlight such differences and describe the features of a new architectural style: called Ecosystem Oriented Architecture (EOA).**

The paper will explore the fact that a DES oriented architecture is not a "sort of SOA", it's not just a "bigger SOA". A whole set of new problems are to be addressed namely responsive alignment with the business, decentralization, ownership of the knowledge base and self healing.

EOA is a new mindset in decentralized architectures for DES.

I. Introduction

A Digital Ecosystem (DES) implementation need to support a particular dynamic scenario where business service aggregations and evolutions are key. Neither B2B of market place typical solutions are able to adequately tackle such challenges. DES has to "*..exploits 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 foster and push the accelerator in the dynamic of the business, but these solutions are still limited though 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's often the implementation of a single-organization supply chain. B2B solutions are rarely applied outside the boundary of an enterprise and if it does it, it is a challenging project: it's cumbersome, and especially complex to be maintained.

DES are to be implemented applying a new perspective in the Software Architecture that has to overcome SOA: an **Ecosystem Oriented Architecture** (**EOA**). We intends to pin point the fact that DES specific features and issues can not properly be addressed by SOA; there is the need to define a different architectural style that specifically tackle DES requirements as both the functional and structural viewpoint. Applying SOA when dealing with DES implementations is overlooking at the problem.

[1]Dr. James F. Moore is a Senior Fellow at Harvard Law School's Berkman Center for Internet and Society

II. SOA and Ecosystem Oriented Architecture

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 where SOA was born and where it is actually used most of the time. SOA is an architectural style that evolved from EAI, RPC, CORBA, where the focus was on Application, Procedures, Objects; focus on services was added later but still with an "inside enterprise" mindset (Figure 2 below).

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

DES 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 DES, when enlarging participants to a broader spectrum, supporting a wider 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 experience, it is evident that dynamism and flexibility are key in a Business Ecosystem:

- the value chains are overlapping, it is not a partition but they intersect each other;
- the social and business network topologies are not hierarchical[4];
- a single functional reference model can not be implemented;
- there is not a single point of management at both the business or structural point of view;

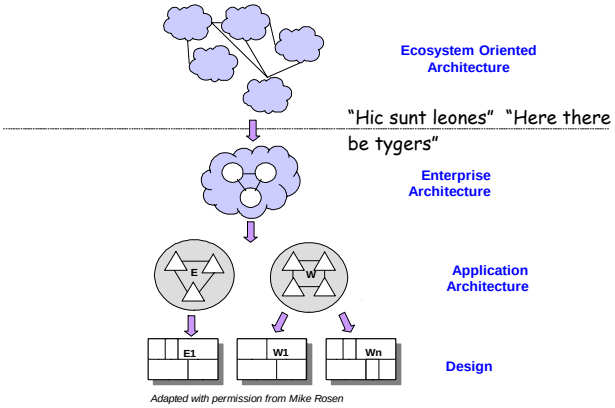

Figure 1: Ecosystem Oriented Architecture positioning

III. FUNCTIONAL REFERENCE MODEL

Digital Ecosystems crosses business domains and different value chain, for this reason they are characterized by not having a single functional reference model. Since it's 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 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 book store when required to automate the order process; for each supplier/distributor a different technical adapter is required. This trouble 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 ecosystem; without fast business alignment, there will be no evolution.

The rather ingenuous approach to fix 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 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 create again interoperability problems. Nevertheless, maintaining the specification would be very time consuming and at the end it'd not be possible to keep it aligned with the business requirements: new features driven by the end users or marketing will incur the risk of being left behind, waiting for the new specification to emerge or -even worst- to be implemented in contrast to the standard. As a consequence, the expected well ordered mechanism will soon break.

This scenario is a gross over-simplification of the models what might be found in an DES, 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 are being investigated only in recent times. Even maintaining the functional models of a complex ERP project, with well defined boundaries and dependencies, can be be very hardly and unable to be managed by a single person; changes and update are often a tough asks to accomplish.

In an business ecosystem this effort can not be addressed at all, and a new mindset and approach in this sense is required, and 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 contrast the dynamic nature of a DES, it is better to support it.

The way to go then assumes the inability of controlling the reference models; we might assert there is not reference model at all, and take all the required architectural decision to support it and letting the ecosystem to converge, in dependence with the time, in a model. What is fundamental to assume when defining the architecture of a DES is to recall that it's 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 need a mechanism to allow participants to:

- publish whatever model;
- to investigate which is the most adequate to their needs;
- to adopt is (and to change it) in a total free uncontrolled space (regulatory and restrictive features, shall be added as a mean just to avoid hacking the environment).

A structured and highly connected repository has to manage models, their dependencies and the 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 book sector that is adopted by the 75% of the book stores and another one less adopted (hence less connected) but more close to their technical need, that would be more straightforward to be implemented due to the better alignment with their back-end systems. The distributor has the chance to decide if to adopt the most connected model, hence easing the migration and adoption by the book stores, or to stick to the easy way with an obvious drawback regarding the level of adoption. In this scenario it is evident that book stores (the service consumers) on the other hand will try to reduce the number of different models in order to ease their integration efforts and be biased by the quality of the service offered. The bal-

[2]FETISH Project IST-2001-35113, FP5 Programme, 2002-03-01 to 2003-08-31

[3]http://research.lumeta.com/ches/map/, http://www.opte.org/, http://www.opte.org/

ance between those symmetric aspects is the base of competition and evolution.

IV. 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, 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 descents from classical RPC approaches. UDDI is essentially a catalogue of programmatic resources.

As an example: two different books distributors might use the same technical specification of the service (eg. WSDL) but have different kind of discount policies, different return policies, different quantity discount 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 a SOA the need for modelling the business specification is not a prime need because there is no economical transaction involved. SOA is often implemented, in the author experience, in a context where the associated business transaction costs is 0 (zero). Nevertheless, the writer is aware o some SOA implementations (rather tough though) were an invocation implies an effective business transaction, i.e. "money exchange". But also in these cases the participants, and the services involved have been defined upfront -statically- and the business models are known in advance: there is not dynamic discovery or negotiation in there, for this reasons -under these assumptions- SOA works fine: in DES on the other hand it'd not scale. From reference documentation about UDDI it can be read "*Companies can establish a structured and standardized way to describe and discover services*"[5], but DES are not a structured or standardized environment.

In a DES 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 unable. Services in DES 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 assume 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 the most correct 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 actual service instance catalogue: "The service registry".

V. Service Registry

The service registry contains the references to actual services published in a DES associated with the technical and business models. Each entry contains 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 is essentially the business specification (it might be a reference to an entry in the model repository, this an implementation aspect which is not relevant in this context).

The second is the technical specification of the service.

The third are information specific to the service instance, for example the name of the published or the location of the service; in general these information are associated with the business model.

The fourth are programmatic information needed to actually invoke the service, for example – it is an over simplification- the IP address and the protocol used.

In 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 of a great frustration to try to invoke services from information found in the UDDI just to discover that it is not available. The real issue in these cases is that the requesting service is not able to tell the reason of the failure, is it due to the fact that it has been discontinued or because there are some temporary technical issue? In an intranet SOA implementation, the architect has the ability put all the efforts in order to have an high availability of the service: in the Internet this can not be assured. As a solution, the service entry in the registry needs to be bind with the actual remote published service so that it provides up-to-date status information; since it's too administrative 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 need to be MOF[8] compliant in order to ease the issues related to

[4]"By enabling policy-based distribution and management of enterprise Web services, a UDDI registry can deliver significant business value", Frank Kenney, analyst at Gartner
http://www.oasis-open.org/news/oasis_news_02_03_05.php

[5]"What's New in Enterprise UDDI Services" Microsoft Windows Server 2003: http://www.microsoft.com/windowsserver2003/evaluation/overview/dotnet/uddi.mspx

[6]http://www.jini.org]

[7]http://fada.sourceforge.net/

[8]http://www.omg.org/mof/

model interoperability.

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

VI. Basic Services

An architecture for DES need to consider a set of basic business services to support the ecosystems and easing the quick and fast interaction between business services. A DES without a proper set of basic service, will hardly self sustain: 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 a SOA implementation as it was mentioned above essentially because a service invocation in a DES a business service consumption. For same reasons services such are reputation and trust are as fundamental to be provided.

The following service are needed to facilitate the bootstrap phase in a DES:

- Payment
- Business Contract & Negotiations[9]
- Information Carriers
- Billing
- Trust
- Reputation
- Legal compatibility

It's important to put in evidence that is not specifically required for these services to be implemented up front. It's import to support for them for example by defining their models in the repository and to provide adequate infrastructure for it's implementation: it might be up to participants and organizations to implement them. Some, like the accounting service, requires to be supported by the core infrastructure of a DES because it has to properly intercept the inter services messages.

One of the most significant service required in a DES is the support for negotiations. In SOA, in those rare case in which it's implemented in across enterprise B2B environments, negotiation happens outside of the IT systems, often in real meetings; in SOA implementations only the service execution is supported together with a poor search mechanism. In DES, following the definition given at the beginning, the ecosystem is such only if the integration mechanisms are fast and automatic. As a matter of fact DES had to replicate in an e-environment what happens in the real concrete 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

In Figure 2 below is an example of the service stack in the Digital Business Ecosystem project (DBE)[9].

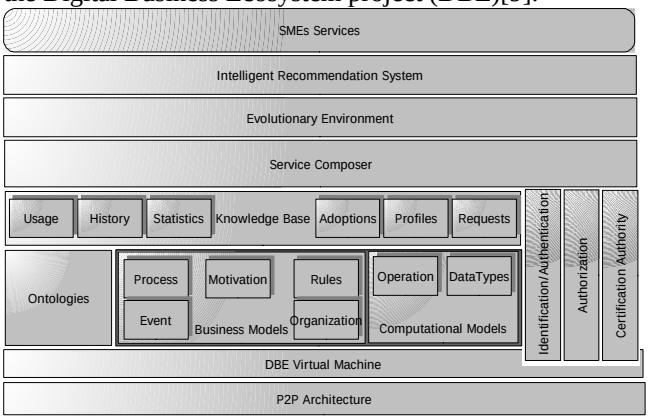

Figure 2: Service Stack in the Digital Business Ecosystem

VII. Single Point of Failure

The Service registry is a key elements 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, all the services won't be reachable as a consequence.

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, but 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 before hand. But for a DES, given the complexity and intricacy of infrastructure, the very fast changes and the absence of any "root" node, it is not the solution to go.

In DES the registry is even more critical because service's IP addresses changes very often while in a classical SOA all the services are published in static IPs and change pretty 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 even not guarantee service continuity in case of natural calamity. The alternate solution is to exploit decentralized approaches, i.e. a topology and replication schema that does not make the DES to depend on a single node but rather on a collaborative set of peer nodes (more on the following Chapter "Scale free networks"). Instead of a controlled cluster of nodes, there is the need to advocate the use of a peer-to-peer networks as the routing infrastructure that improves routing resilience to node failure and attacks for service registries[16]. Such a network of nodes need to be self healing and self adaptable to the ever changing nature of the requests and traffic: there has not to 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.

[9]Addresses by the Open Negotiation Project (ONE) FP6-2004-IST-5, FP6, 6th Call, 2006

In such kind of networks, data replication within nodes happens on a smart basis: entries migrate automatically in relation to requests, moving data toward nodes that started the request. In this way, like in typical caching mechanism, information is copied closer to its source 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 not to geographical distance, e.g. often in Internet 100km far away hub nodes are faster to ping then local server. 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 to have single point of failures for an EOA is key. Beside the technical non marginal aspect of having a more reliable system, the DES will not suffer from the "big brother syndrome". With a decentralized P2P based architecture the knowledge, which is represented by the model repository and the service registry, are not managed by a single institution which could tamper the community by imposing unwanted control. A DES 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 adoption and sustainability. Considering for example what would happen in case the organization hosting the service registry decides to shut it down. Such possibility would impede the adoption of the DES.

DES founds its entire sustainability and existence on knowledge about models and services. Participant in the DES are providing and using models while actively participating and being part of a business community, they are scared about loosing models. Owners of DES knowledge need to be the community itself, to this extend a peer to peer network (Chapter "Scale Free Network below) is a good approach because it is democratic; it gives participants the possibility to offer resources to host part of this knowledge.

The significant drawback is the implementation: such a peer to peer infrastructure need to be self healing and self adaptable. But there are already some framework and tools that support leverage the properties of the Scale Free networks.

VIII. Scale Free networks

Most of the solution in SOA, like the cluster of UDDI registries, are based on an hierarchical structures because this is the way humans do 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 becomes 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 simplified to a hierarchy that represents a good approximation., but with the increase in complexity it also becomes impossible to stick to an hierarchy because reality is not as simply structured: it's based on different models and topologies: Scale Free network[15].

The scale free network 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 base of business and social networks[15], a proper EOA has to support it and define the proper mechanisms in order to let it emerges in a self organized way without human intervention.

In order to have s Scale Free Network emerge it is required 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 property of a "small world" allows efficient searches[17][18].

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

Technologies are already available and they make use of concept like the Tuple Space or the Distributed Hash Table, for example Sun's Jini(tm) Network Technology[10], FADA[11], Bamboo[12], Cord[13], and others; there are also commercial implementations like GigaSpaces(c)[14]. P2P architecture can help, even if they are used to infringe copyright: there is no need to have a prejudice, a technology is not bad per-se, it depends of the way it is used.

The Digital Business Ecosystem (DBE)[15] has make significant step forward in this direction.

IX. Conclusion

Service Oriented Architectures (SOA) does not scale nor it addresses the new challenges in 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 addresses[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 filled with data;
- service instances: service name and endpoint to actually invoke and consume a service.

In DES it is essential to have a repository of model separated from the registry of services[20]. The model repository needs a whole set of discovery features and supports

[10]http://www.jini.org
[11]http://fada.sourceforge.net/
[12]http://bamboo-dht.org/
[13]http://pdos.csail.mit.edu/chord/
[14]http://www.gigaspaces.com/
[15]http://www.digital-ecosystem.org

XMI in order to implement model driven capabilities like dependency, versioning, merge and inheritance. Services need to be described also from the business point of view, the computational specification is not sufficient in DES because services are not known in advantage and the discovery process need to be more smart and based on business specifications.

The service registry need to overcome the static limitation of UDD like services and be dynamically bind to actual published service. In the next future a lot of mobile services are expected and these devices are going to make use of dynamic IPs, a SOA based approach is not enough. The service instances are to be resolved at run-time via a sort of DNS service.

For the nature of a DES, the architecture needs to avoid single point of failure, the best approach envisioned is the 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 like negotiation, information carriers, payments, accounting, billing and others. While SOA essentially supports only the service execution phase, a DES has to support the entire business service life-cycle like 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 DES: 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 DES is no different.

X. References

[1] Dr. James F. Moore, "The Death of Competition", Collins, May 21 1997, ISBN 0887308503

[2] EU Discussion paper, Bruxelles, September 2002

[3] P.Dini et others, "Towards Business Cases and User-Oriented Services in Digital Business Ecosystems", Conclusions FP7 Workshop on Needs and Requirements of Regions, Bruxelles, 18 April 2005

[4] [1]A. L. Barabási, H. Jeonga, Z. Néda, E. Ravasz, A. Schubert and T. Vicsek, "Evolution of the social network of scientific collaborations", 16 January 2002.

[5] Milena Mihail, Christos Papadimitriou and Amin Saberi, "On certain connectivity properties of the internet topology", Journal of Computer and System Sciences, Volume 72, Issue 2, March 2006, Pages 239-251

[6] F.Nachira, M.Le Louarn, "ICT supporting businesses and industry", "Innovation ecosystems", FP7 – General Orientations, February 2006.

[7] Paolo Dini, ,Tuija Helokunnas, Pierfranco Ferronato, Angelo Corallo, Neil Rathbone, "Towards a semantically rich business modelling language for the automatic composition of web services", e-Business Research Center (eBRC) 2004

[8] P.Ferronato, "Technical Architecture of the Digital Business Ecosystem Project", MDA Technical Forum, Tokyo, 20th, 21st October 2004

[9] T. Heistracher, T. Kurz, C. Masuch, P. Ferronato, M. Vidal, A. Corallo, P. Dini, and G. Briscoe, "Pervasive service architecture for a digital business ecosystem", In ECOOP First International Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT04), Oslo, June 2004.

[10] "UDDI Version 3 Features List", OASIS, 2004, http://uddi.org/pubs/uddi_v3_features.htm

[11] M.Vidal, P. Hernandez, "Building a Digital Ecosystem", SUN, Java Conference 2005

[12] Pierfranco Ferronato, Stefano Lotti, Daniela Berardi, "Titolo: Strategia architetturale per la Sanità Elettronica", Innovazione Italia, Dipartimento per l'Innovazione e le Tecnologie, 31/03/2006 14.17

[13] P. Dini et Others, "The Digital Ecosystems Research Vision: 2010 and Beyond",

[14] Albert-Laszlo Barabasi, Duncan J. Watts et others, "The Structure and Dynamics of Networks", ISBN: 0691113572, Princeton University Press (April 17, 2006)

[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

[16] Albert-Laszlo Barabasi, "Linked: How Everything Is Connected to Everything Else and What It Means", ISBN: 0452284392, Plume; Reissue edition (April 29, 2003)

[17] Nancy A. Lynch, , "Distributed Algorithms", Elsevier Science & Technology Books, 1996.

[18] Fletcher, George, Hardik Sheth, "Peer-to-Peer Networks: Topological Properties and Search Performance", Indiana University, 2003, http://www.cs.indiana.edu/~gefletch/papers/l597report.pdf

[19] P. Ferronato, "WP21 - DBE Architecture Requirements Del 21.1 - Preliminary design, usage scenarios, and architecture requirements", D.B.E. Digital Business Ecosystem Contract n° 507953, October 2004, http://www.digital-ecosystem.org/Members/aenglishx/linkstofiles/deliverables/Del_21.1_DBE_Architecture_Requirements.pdf/download

[20] P.Ferronato, "WP 21: DBE Architecture Requirements, Del 21.2: Architecture Scope Document", D.B.E. Digital Business Ecosystem Contract n° 507953, October 2004,[1]http://www.digital-ecosystem.org/Members/aenglishx/linkstofiles/deliverables/Del_21.2_DBE_Core%20Architecture%20Scoping%20Document_Release%20C.pdf/download

[21] Melvin Conway, an issue of Datamation, April 1968

[16] Melvin Conway a Cobol programmers in the late 60s