DELOS – Deliverable 1.1.1

WP 1: Survey on

Peer-to-Peer Architectures, Grid Infrastructures, and Service-oriented Architectures for Digital Libraries

Authors:
M. Agosti [1], L. Bischofs [2], L. Candela [3], D. Castelli [3], N. Ferro [1],
W. Hasselbring [2], N. Moumoutzis [4], H. Schuldt [5,6], G. Weikum [7],
M. Wurz [5], P. Zezula [8]

Editors:
D. Castelli [3], H. Schuldt [5,6], G. Weikum [7]

[1] Department of Information Engineering,
University of Padua, Italy

[2] OFFIS, Oldenburg, Germany

[3] Istituto di Scienze e Tecnologie dell'Informazione (CNR)
Pisa, Italy

[4] Laboratory of Distributed Multimedia Information Systems & Applications
Department of Electronic & Computer Engineering
Technical University of Crete, Greece

[5] Information & Software Engineering
University for Health Sciences, Medical Informatics and Technology (UMIT)
Austria

[6] Database and Information Systems Group
University of Basel, Switzerland

[7] Max-Planck-Institut für Informatik
Saarbrücken, Germany

[8] Faculty of Informatics
Masaryk University Brno, Czech Republic

Abstract

Second generation DLs have new requirements. Firstly, these requirements include the way to make data and specialized DL applications available to a large set of users by means of well-defined services. Secondly, the interaction between independent data/service providers (peers) within a network needs to be addressed. Finally, the computing and storage resources that are available in large networks have to be made available in an efficient and effective way by means of grid technology.

We provide a survey in order to highlight whether and to what extent i.) service-oriented architectures, peer-to-peer infrastructures, and grid infrastructures can contribute to satisfy the requirements of these new DLs. This survey contains a detailed summary of state-of-the-art in the three architectural paradigms and lists current approaches that aim to exploit these architectures in DL Management systems and DL applications.

Introduction

Current Digital Libraries (DLs) are usually content-centric, special-purpose systems that are targeted for storing static digital content and are in most cases used for library and/or cultural heritage applications. A result of this content-centric approach is that systems tend to be tailored to concrete application domains rather than being of general applicability.

Second generation DLs aim to overcome these limitations. The overall goal of these new DL systems is to enable any citizen to access all human knowledge anytime and anywhere, in a friendly, multi-model, efficient and effective way by making use of multiple Internet-connected devices. Moreover, they shall help in overcoming the barriers of distance, language, and culture. This goes along with a large number of vital requirements for these next-generation DLs. Different DL applications have different requirements on the underlying DL systems as can be seen in the following three examples.

Example 1: Management and Coordination of Information Spaces.

In large organizations, data and documents are usually stored in various distributed databases, repositories, etc. As an example, consider the information space of a university, consisting of several databases (e.g., of the university library, research reports databases, etc.). Moreover, data is also stored in file systems and made accessible by different web servers (of institutes, research groups, etc.). Search for data and documents in the information space has to be supported by dedicated indexes. Consider a user that exploits a content-based image similarity search service (no. 1 in Figure 1). The query results stemming from the distributed data sources are displayed (2) while the actual distribution is made transparent. When new information is inserted into the information space (3), it should be made available as soon as possible. This means that the index needs to be updated immediately and not in a style that is conventionally used by search engines – the latter need quite some time i.) until a search robot happens to detect this new artefacts and ii.) until the index is updated. The active propagation usually encompasses several activities to be performed on the newly inserted artefacts (e.g., replication, extraction of characteristic features like colour, texture, shapes, and finally the update of the index). This sequence of activities (workflow process) has to be automated by the underlying Digital Library management system to ensure timely updates of derived data and consistency of the overall information space.

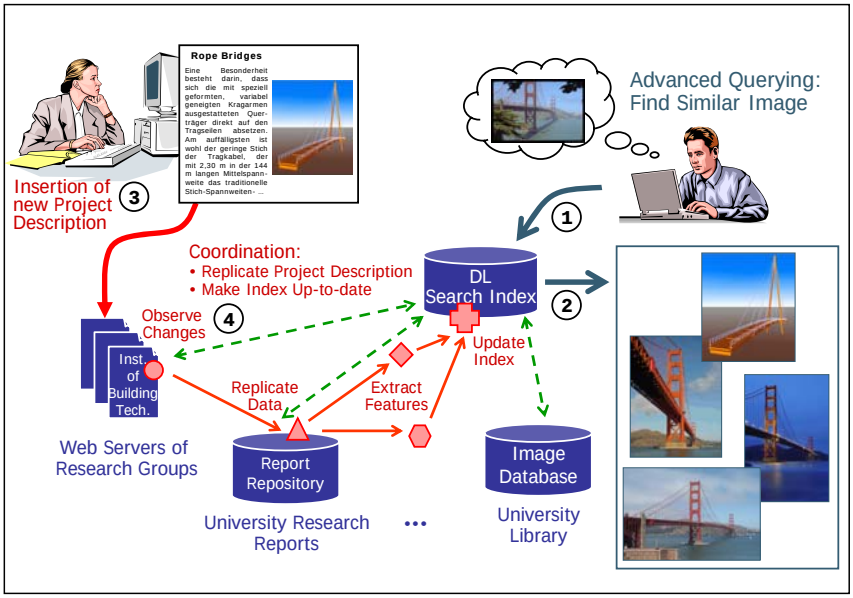

Fig. 1: Management and Coordination of Information Spaces

Example 2: Telemonitoring in eHealth.

Telemonitoring supports the analysis and processing of data reflecting vital information on patients. Based on sensors that are integrated into the patient's clothes or their home environment, the health state can be seamlessly monitored by processing and analyzing continuous streams of data. The underlying DL has to provide a high degree of reliability and availability (a system their users can rely on). Moreover, considering the mobility of out-patients, also mobile devices for hosting (some) of the operators required for processing sensor data have to be supported (see Figure 2). Finally, aggregated stream data and the results of data stream processing have to be integrated into other eHealth DL applications like the (distributed) electronic health record of a patient.

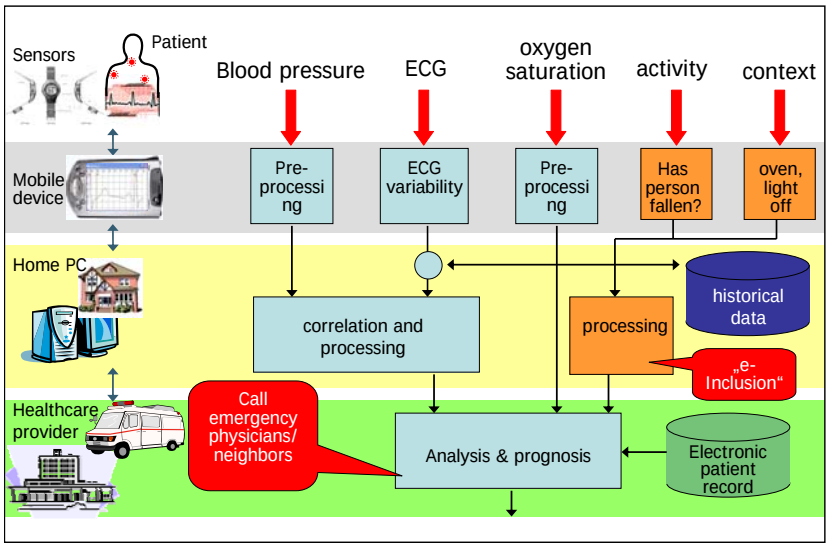

Fig. 2: Telemonitoring in eHealth

Example 3: Distributed Electronic Patient Records and Similarity Search.

As another example for next generation DLs, consider the digital artefacts which are generated about patients throughout their lifetime. Usually, these artefacts cannot be physically integrated but are, for administrative and also legal reasons, stored under the control of the healthcare providers that have generated them. When a physician needs access to the full medical history of a patient, the different artefacts have to be identified, located, and virtually integrated. The search for and integration of the artefacts has to be supported by the underlying eHealth DL (see Figure 3, top), by appropriate, privacy-preserving index services. Patient records comprise many multimedia objects of various media types. Another requirement for an eHealth DL is the support for multi-feature, multi-object queries within these eHealth patient records (c.f. Figure 3, bottom). This, in turn, requires the possibility to compute document similarity based on similarity of component objects, even when the numbers and types of objects in documents differ.

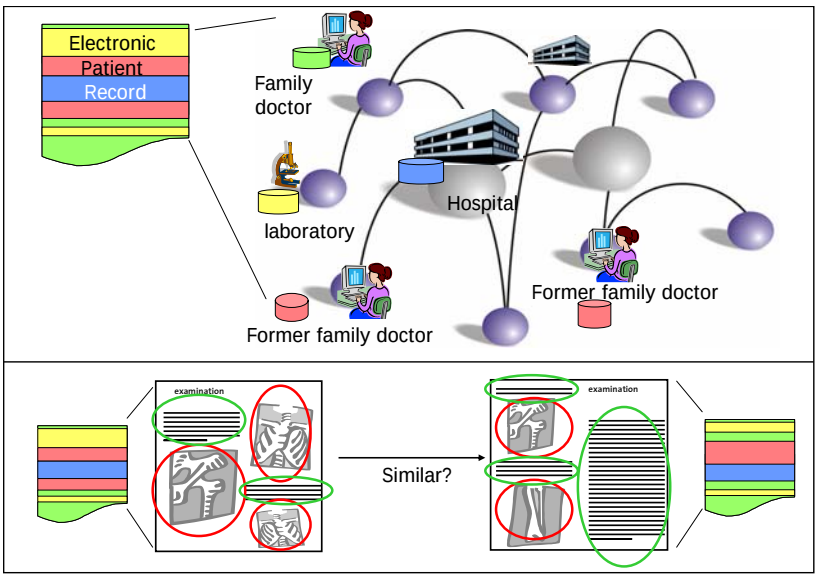

Fig. 3: Distributed Patient Record (top) and Similarity Search (bottom)

The above mentioned examples have shown that second generation DLs require a shift from content-centric to person-centric solutions. In terms of the digital artefacts to be stored, not only static content has to be considered but also content that is frequently updated, continuously created/revised, etc. Systems have to support the active communication and collaboration of their users.

Although the three of examples of second generation DLs and DL applications mentioned above are taken from a very long list, it can be seen that all have individual, application-specific requirements a DL has to support. The most important of these requirements (stemming from the above mentioned examples but also from others that are not mentioned in detail) are:

- Availability of specialized services, *local to a content provider*, such as
  - Search (different media types, content-based similarity search, multi-object, multi-feature queries, relevance feedback, etc.)
  - Indexing
  - Annotation
  - Metadata management
  - Content management
  - Resource Management
- Availability of specialized services, *across different providers*. In addition to the local services like content management, search, etc. which are also needed outside of content providers, this includes metadata management and indexing in a distributed way, without central control (no censorship)
- Generation of virtual DLs across several content providers
- Management of services which are distributed, heterogeneous, and/or autonomous. Especially for computationally intensive services, this includes the possibility to scale-out (installation and deployment on demand) as well as load balancing
- Composition of services; this includes the definition of complex services (processes) on the basis of existing services, the automation of these processes, and the flexible, automatic adaptation to changing environments
- Notification of changes and the guaranteed consistency of derived data is needed in the case of dependencies between digital artifacts
- Personalization, visualization, access from everywhere, especially from mobile devices
- Context- and location-awareness
- Authentication and authorization, preservation of privacy
- High degree of availability: access needs to be guaranteed 24/7 (e.g., by means of replication)
- High degree of scalability
- High degree of dependability/reliability: DLs must be systems their users can count on
- Processing of continuously generated data streams (e.g., from sensor networks, hardware or software sensors)

These requirements can be summarized by three main issues: i.) the way to make data and specialized DL functionality available to a large set of users by means of well-defined interfaces, ii.) by the interaction between independent data/service providers within a

network, and iii.) by making available the computing and storage resources that are available in large networks in an efficient and effective way.

The goal of this survey is to analyze whether and to what extent service-oriented architectures (SoA), peer-to-peer (P2P) infrastructures, and grid infrastructures can contribute to satisfy the requirements of second generation DLs.

This survey is organized as follows. The first part contains a detailed summary of state-of-the-art in service-oriented architectures (Section 1.1), peer-to-peer architectures (Section 1.2), and Grid infrastructures (Section 1.3). The second part starts with an overview on relevant DL systems/repositories (Section 2.1) and provides an overview on current approaches on service-oriented architectures, peer-to-peer architectures, and grid infrastructures for Digital Libraries (Sections 2.2 – 2.4) mainly undertaken by the DELOS WP1 partners. Finally, Section 3 discusses the applicability of these technologies for DL management systems and DL applications.

Part 1: State-of-the-Art

1.1 Service-oriented Architectures

A *service-oriented architecture* is a component model that inter-relates the different functional units of an application, so called services, through well-defined interfaces and contracts between these interfaces. It is also known as an architectural style whose goal is to achieve loose coupling among interacting software entities. The communication among the functional components, the services, can involve either simple data passing or it could involve two or more services coordinating some activity. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software entities on behalf of their owners. More formally, a service is a function that is well-defined, self-contained, and does not depend on the context or state of other services [116], [125].

Service-oriented architectures differ from other architectural concepts by their emphasis on ‘separation of concern’ and on loose coupling. The loose coupling among interacting software entities and services is achieved by

- Using a simple and ubiquitous interface to all participants. Only generic semantics is encoded at the interface. Interfaces have to be universally available for all providers and consumers, meaning that they should be independent of hardware platform, operating system, and programming language they are implemented with.
- Descriptive messages constrained by an extensible schema delivered through the interfaces. No or only minimal system behaviour is prescribed by means of messages. An extensible schema allows new versions of services to be introduced without breaking existing services.

Having simple, generic interfaces is vital to interaction among components. The emphasis on loose coupling enables the architecture to better survive evolutionary changes in the structure and implementation of the internals of each service. When working with traditional distributed architectures, interfacing is an expensive and very error prone task. The need for loosely coupled systems rose from the need of business applications to act more agile based upon the need of businesses to adapt quicker to changing environments as there are changing policies, business focus, partnership, industry standing, and so on. Attracting these companies

to shift their computer systems to this architectural pattern, the marketing slogan "On Demand Computing" was coined by IBM [69] and is now used throughout the industry [66], [76].

Since only few generic interfaces are available, application specific semantics have to be expressed in the messages exchanged. The following rules have to be followed when calling a system following a service-oriented architecture:

- The message has to be descriptive rather than instructive (the service provider is responsible for how to solve the problem, so only a problem description is passed)
- The message has to follow a certain format, structure and vocabulary to be understood. Limiting the vocabulary and structure is necessary for efficient communication, but reduces extensibility.
- Extensibility is fundamental (although it has to be reduced in favour for efficient communication)
- There has to be a mechanism for a service consumer to find service providers under the context of a service sought by the consumer. This can, but need not be, a central registry.

The early representatives of service-oriented architectures, even if not named that way in these days, are DCOM [88] or Object Request Brokers (ORBs) based on the CORBA [96] specification. Another technology for loosely coupling components that is widely spread in enterprise wide system architectures is *message-oriented middleware* [125]. Representatives of this kind of systems are, for example, MQseries [68] by IBM, DECmessageQ [39] by DEC or Message Queue Server [89] by Microsoft. An extensive list of vendors of message oriented middleware can be found in [85]. These technologies already allow the communication and collaboration of software across network boundaries, although having increasing difficulties passing corporate firewalls. This is one of the major drawbacks of these architectures that is addressed by *Web Services* based service-oriented architectures.

Web Services – a Modern Approach to Service-Oriented Architectures

Although service-oriented architectures are not bound to a specific implementation or technique, the state of the art in service-oriented architectures is *Web Services* [7], [28], [40], and [92]. It is mainly driven by IBM, Microsoft, and other industry partners. Web Services have to support protocols like SOAP (Simple Object Access Protocol) [118] for the invocation of web services, with parameters and invocation details shipped in XML format [172]. In order to allow services to be dynamically discovered, they have to be described, e.g. using languages like WSDL (Web Service Description Language) [163].

It is generally accepted that a web service implements a service-oriented architectures, placing the following additional constraints on the architecture:

- Interfaces on the transport layer are based on internet protocols such as HTTP, FTP, and SMTP.
- Messages must be in XML format, except for binary data attachments

Web services mainly refer to SOAP-based service invocations. SOAP web services encapsulate their messages within a SOAP envelope, and are described using WSDL. The advantage of this approach is that it allows for rich message exchange patterns ranging from traditional request/response to broadcasting and sophisticated message correlation.

SOAP allows for the unidirectional information exchange in a distributed, service-oriented environment. Within the SOAP web services world, two flavours of SOAP calls exist: SOAP

RPC for remote procedure calls using the technologies mentioned above, and document-centric SOAP web services. The former type supports the 'tunnelling' of application-specific remote procedure calls through a generic interface. SOAP 1.2 [117], as well as the web services interoperability (WS-I) basic profile [148], made the support for RPC optional.

WSDL provides an implementation-independent description of methods (operations) of web services and their interfaces, respectively. In addition, a WSDL web service description also includes the possible bindings within transport protocols (how the web service can be invoked) which is in most cases SOAP.

Finally, web service and their descriptions need to be found in a distributed system. To this end, several repositories and registries exist. The most common directory service used for service discovery is UDDI (Universal Description, Discovery, and Integration) [129]. The web services dynamic discovery (WS-Discovery) specification [144] defines a multicast discovery protocol to locate services. The web service inspection language specification (WS-IL) [149] provides an XML format for assisting in the inspection of a site for available services and a set of rules for how inspection related information should be made available for consumption. A WS-Inspection document provides a means for aggregating references to pre-existing service description documents which have been authored in any number of formats.

Within the approach of using web services to realize service-oriented architectures, there is a vast amount of specifications that is currently transferred to working implementations. These specifications address messaging and routing, notification mechanisms, transactional semantics, and security aspects. Finally, the WS-Manageability specification [151] introduces the general concepts of a manageability model in terms of manageability topics (identification, configuration, state, metrics, and relationships) and the aspects (properties, operations and events) used to define them.

In terms of messaging, WS-Addressing [134] provides transport-neutral mechanisms to address Web services and messages. Essentially, XML elements are used to identify web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner. The WS-MessageDelivery specification [152] defines a mechanism to reference Web services, a SOAP binding fo*r* abstract message delivery properties (AMDP), and the relationship of those properties to WSDL definitions and message exchange patterns. These properties enable SOAP messages to be transport-independent, thereby extending messaging capability to use separate transport protocol sessions or even using different transport protocols within the context of a message exchange pattern. WS-Routing [157] is a simple, stateless, SOAP-based protocol for routing SOAP messages in an asynchronous manner over a variety of transports like TCP, UDP, and HTTP. Reliable messaging which is critical to some applications of Web Services is addressed by the web services reliability specification (WS-Reliability) [155].

In terms of notifications, WS-Eventing [145] describes a protocol that allows web services to subscribe to or accept subscriptions for event notification messages. WS-Notification [137] is a family of documents that supports publish/subscribe notification patterns. The WS-BaseNotification, the basis of all WS-Notification specifications, defines the web services interfaces for producers and consumers of notifications. It includes standard message exchanges to be implemented by service providers that wish to act in these roles, along with operational requirements expected of them. WS-BrokeredNotification defines the web services interface for the notification broker, which is an intermediary that allows publication

of messages from entities that are not themselves service providers. Finally, the WS-Topics specification defines a mechanism to organize and categorize items of interest for subscription.

The Web Services Transactions specifications (which comprise WS-BusinessActivity, WS-AtomicTransactions, WS-Coordination) define mechanisms for transactional interoperability between Web services domains and provide a means enrich Web services applications by transactional semantics. The WS-AtomicTransaction [136] specification defines three specific agreement coordination protocols for the atomic transaction coordination type: completion, volatile two-phase commit, and durable two-phase commit. Developers can use any of these protocols when building applications that require consistent agreement on the outcome of short-lived distributed activities that have the all-or-nothing property. The WS-Coordination [142] specification describes an extensible framework for providing protocols that coordinate the actions of distributed applications. Such coordination protocols are used to support a number of applications, including those that need to reach consistent agreement on the outcome of distributed activities. The WS-BusinessActivity specification [138] provides the definition of a business activity coordination type used to coordinate activities that apply business logic to handle business exceptions. Actions are applied immediately and are permanent. Compensating actions may be invoked in the event of an error. The BusinessActivity specification defines protocols that enable existing business process and workflow systems to wrap their proprietary mechanisms and interoperate across trust boundaries and different vendor implementations. The Web Service Transaction Management specification (WS-TXM) [161] defines three distinct transaction protocols that can be plugged into the coordination framework for interoperability across existing transaction managers, long running compensations, and asynchronous business process flows.

For securing web service calls, WS-Security [158] describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies. WS-SecurityPolicy [160], which extends WS-Security, indicates the policy assertions for which apply to WS-Security specifications. The Web Services Trust Language [162] (WS-Trust) uses the secure messaging mechanisms of WS-Security to define additional primitives and extensions for security token exchange to enable the issuance and dissemination of credentials within different trust domains. The Web Services Secure Conversation Language (WS-SecureConversation) [158] is built on top of the WS-Security and WS-Trust models to provide secure communication between services. The WS-Federation specification [146] defines mechanisms that are used to enable identity, account, attribute, authentication, and authorization federation across different trust realms.

Service Composition and Orchestration

In addition to the description and invocation of single web services, it is of vital importance to be able to combine several web service invocations to value-added composite web services or (workflow) processes [7]. Essentially, such processes allow for the integration of arbitrary local and remote web service calls. Several approaches to define specifications and languages for service composition exist. The most common one is BPEL4WS (Business Process Execution Language for Web Services) [23], which combines ideas from WSFL (Web Services Flow Language) [166] and XLANG (XML Business Process Language) [171]. Similarly, the WS-Choreography specification [141] provides an information model that describes the data and the relationships between them that is needed to define a choreography

that describes the sequence and conditions in which the data exchanged between two or more participants in order to meet some useful purpose. The Web Services Conversation Language (WSCL) [163] allows the definition of business level conversations supported by a Web service. WSCL specifies the XML documents being exchanged, and the allowed sequencing of these document exchanges. WSCL conversation definitions are themselves XML documents and can therefore be interpreted by Web services infrastructures and development tools.

The Web Services Composite Application Framework (WS-CAF) [139] is a collection of three specifications: Web Service Context (WS-CTX), Web Service Coordination Framework (WS-CF), and Web Service Transaction Management (WS-TXM, see above; it includes a solution to bridge different transaction models which is needed when multiple Web services are used in combination to support information sharing and transaction processing). The Web Service Context specification (WS-CTX) [143] provides an open, common, interoperable runtime mechanism to manage, share, and access context information among related Web services. The Web Service Coordination Framework (specification WS-CF) [140] defines a software entity to handle context management. Web services in a composite application register with a coordinator to ensure that messages and results are correctly communicated and allow, e.g. the success or failure of an individual service to be tied to the success or failure of the larger unit of work comprising multiple Web services.

For the execution of business processes (composite web services), several commercial systems like the IBM WebSphere Choreographer [70] or BizTalk of Microsoft [86] exist. These commercial systems follow a centralized approach, where every call to a service provider returns to the process engine. Albeit navigation tasks can be distributed in a cluster, storage of process instances is usually done by using a single, centralized database instance (products like Oracle 10g [98] can also support clustered databases). In the distributed Mentor-lite approach [58], the setting of the process engine in a cluster can be changed actively using a configuration tool. In addition, various workflow and process management systems have been developed in academia (e.g., SWORDIES [128], Panta Rhei [43], MENTOR-lite [133], WISE [8], or OSIRIS [114], [115].

Crucial to processes is that they are executed and coordinated with dedicated transactional guarantees. In the context of transactional workflows, the intersection of transaction and process management, various contributions can be found in the literature (e.g., [57], [74]). ConTracts [131] bring together aspects of programming languages for control flow specifications and transactions. Each ConTract is considered as a long-running transaction. Chen and Dayal propose to apply nested transactions for process execution (Open Process Management [30]). Spheres of joint compensation [79] address the fault-tolerant execution of single processes. Flexible transactions [46], [175] introduce advanced failure handling strategies that can be applied to processes. This allows for the specification of alternative executions that can be chosen in case of failures. Transactional processes [113] which are supported by the OSIRIS system (Open Service Infrastructure for Reliable and Integrated process Support [114]), combine these sophisticated failure handling strategies with the guarantee to enforce consistent interactions of concurrent process executions. In the context of the IBM WebSphere Choreographer [70], a distinction between microflows (short-running business processes) and macroflows (long-running business processes) is made. Microflows are non-interruptible and fully-automated processes which usually encompass only transactional activities, i.e., activities whose resources support the XA protocol. In this case, the execution of a microflow is atomic and the complete microflow is executed within a single transaction. Macroflows, in contrast, are interruptible and can involve asynchronous activities

or activities with human interaction. In a macroflow, each activity is executed within an individual transaction. During the execution of a macroflow, the execution state is made persistent in the underlying database, which allows for forward recovery in case of failures.

In terms of service discovery, existing systems usually implement either an approach based on tModel types of WSDL (e.g., ServiceGlobe [75]), or include service discovery into the process navigation (e.g., eFlow [27], ISEE [83], or CrossFlow [63]).

1.2 Peer-to-Peer Architectures

Peer-to-peer (P2P) architectures have become very popular for information systems in general. Especially file-sharing applications like Napster, Gnutella, etc. have demonstrated the potential strengths of P2P approaches, but have also pointed to more sophisticated and partly controversial issues along technological, economic, and legal dimensions. Recent books that aim to discuss all these aspects of P2P architectures are, for example, [99] and [111].

In addition to file sharing, other application areas of P2P technologies include instant messaging, collaborative authoring and other groupware, publish-subscribe applications, etc. For all these cases, P2P networks are the basic infrastructure for virtual communities that share resources, computer resources like processors, memory, and disks as well as intellectual resources like user annotations and recommendations.

In contrast to client-server systems, P2P architectures emphasize that all peers are equal and autonomous and there is no central coordination of how peers share resources and interact with each other. P2P should rather be self-organizing, even in the presence of many failures and the high dynamics of large-scale systems. The following principles are widely seen as key characteristics of a P2P system, distinguishing this system paradigm from other classes of distributed computer systems:

- *Decentralization:* Each node of a P2P network can store and process data and can exchange it with other nodes at its discretion. There is no central coordinator, which could become a single-point-of-failure or a load bottleneck. All nodes have equal capabilities and rights: they are peers.
- *Sharing of distributed resources:* Peers share physical resources like storage space, computing power, and network bandwidth as well as logical resources like data, metadata, statistics, etc.
- *Autonomy:* Each node has full control over its resource usage on behalf of other nodes and its interactions with other nodes. The extent to which resources are shared with other nodes may vary over time. In particular, nodes may temporarily leave the network at arbitrary points or may become unavailable for other reasons, and they may permanently leave the network without notice.
- *Self-organization and autonomic behaviour:* The data and load sharing among nodes and their interactions are completely self-organized and should be adapted to changing conditions dynamically and automatically. Every node should be autonomic in the sense that none of its decisions for self-monitoring, self-management, self-healing, and self-optimization requires input by human administration staff.

Historically, P2P architectures can be traced back to the early days of the Internet with simple but completely decentralized services like the Usenet discussion forums. The breakthrough of

P2P started with file-sharing systems like Napster and Gnutella for exchange of MP3 files and other entertainment data. Such systems constituted what is today viewed as the first generation of P2P systems. Both Napster and Gnutella are simple publish-subscribe systems. Napster uses a central index for metadata (i.e., the locations of files); Gnutella uses a simple message-flooding algorithm for locating files; this is very effective but potentially wasteful in terms of its network costs.

The second generation of P2P file-sharing systems improved the Napster/Gnutella technology by either distributing the index over a larger number of super-peers or reducing the messages in the Gnutella-style flooding algorithm by appropriate routing protocols. This generation includes systems such as Freenet, eDonkey, KaZaA (FastTrack), Morpheus, AudioGalaxy, or JXTA. In addition to these commercial or semi-commercial systems, a number of more advanced research prototypes have been developed in the last five years: Chord [124], CAN [102], OceanStore [105], Pastry [106], Farsite [3], Pier [67], YouServ [14], Peers [101], PlanetP [35], ODISSEA [123], to name just some of the most prominent ones.

The technical challenges that the P2P research community is addressing include "standard issues" like efficient localization of data objects and request routing, and strategies for load balancing, failure resilience, and replication. In addition, new challenges that were not discussed in earlier forms of distributed systems are how to deal with denial-of-service attacks, how to define and manage trust, privacy, and anonymity, and how to establish incentive mechanisms for peers to contribute resources and active participation in the P2P network. The importance of incentive mechanisms and fair sharing has become obvious with the analysis of the so-called "free-riding" phenomenon in Gnutella, the fact that most nodes merely download files without contributing any resources to others.

In addition to file-sharing and publish-subscribe, new application domains are emerging for P2P systems. These include Web crawling [22] and search engines (see below), collaborative work or games, collaborative data mining, electronic marketplaces, etc. Also, there is a strong trend to combine P2P architectures with other modern technologies, most notably, Web Services, Enterprise Application Integration (EAI), and Workflow Management [7].

P2P Approaches for Digital Libraries and Search Engines

In a P2P federation of digital libraries, every digital library acts as a peer, and additionally every user and her PC-based software tools (e.g., for personalization) may be viewed as a peer, too. Here, the term digital library is interpreted in a broad sense, including, for example, software repositories, scientific databases (e.g., with gene expression data), thematically specialized Internet portals, and also customized search engines for specific Web fragments. The following considerations apply equally to queries over digital library federations and to metasearch over multiple Web search engines. Key issues in this context are:

- *Peer selection*, traditionally known as database selection or query routing: When a user has an information demand, to which peers does she send her query?
- *Query execution*: How is a query that involves multiple peers executed in a distributed manner? How are execution plans dynamically adapted to an unpredictably changing environment (e.g., because of failures, overload, or peers leaving the federation)?
- *Result reconciliation*: How are search results that are obtained from multiple peers merged into a single ranked result list?

- *Maintenance of metadata and statistics*: How are metadata and statistics about the peers in the system maintained in a distributed manner? What kinds of caching and replication strategies are appropriate? How aggressively should metadata and statistical summaries be proactively disseminated among peers, using gossiping-style protocols? To what extent are consistency and freshness of metadata and summaries needed?

Recent research on P2P systems, such as Chord [124], CAN [102], Pastry [106], or P-Grid [1], is based on various forms of distributed hash tables (DHTs) and supports mappings from keys, e.g., titles or authors, to locations in a decentralized manner such that routing scales well with the number of peers in the system. In such systems, an exact-match key lookup can typically be routed to the proper peer(s) in at most O(log *n*) hops, and no peer needs to maintain more than O(log *n*) routing information. These architectures can also cope well with failures and the high dynamics of a P2P system as peers join or leave the system at a high rate and in an unpredictable manner.

However, the above approaches are limited to exact-match, single keyword queries on keys. This is insufficient when queries should return a ranked result list of the most relevant approximate matches [29]. In the following we briefly discuss some existing approaches towards P2P search with ranked results, obtained from different Web sites, databases, or digital libraries.

Galanx [132] is a peer-to-peer search engine implemented using the Apache HTTP server and BerkeleyDB. It directs user queries to relevant nodes by consulting local peer indexes similar to our approach.

PlanetP [35] is a publish/subscribe service for P2P communities and the first system supporting content ranking search. PlanetP distinguishes local indexes and a global index to describe all peers and their shared information. The global index is replicated using a gossiping algorithm. The system, however, is limited to a few thousand peers.

Odissea [123] assumes a two-layered search engine architecture with a global index structure distributed over the nodes in the system. A single node holds the entire index for a particular text term (i.e., keyword or word stem). Query execution uses a distributed version of Fagin's threshold algorithm [47]. The system appears to cause high network traffic when posting document metadata into the network, and the query execution method presented currently seems limited to queries with one or two keywords only.

The system outlined in [104] uses a fully distributed inverted text index, in which every participant is responsible for a specific subset of terms and manages the respective index structures. Particular emphasis is put on three techniques to minimize the bandwidth used during multi-keyword searches.

[81] considers content-based retrieval in hybrid P2P networks where a peer can either be a simple node or a directory node. Directory nodes serve as super-peers, which may possibly limit the scalability and self-organization of the overall system. The peer selection for forwarding queries is based on the Kullback-Leibler divergence between peer-specific statistical models of term distributions.

Minerva [17] is a distributed search engine prototype based on P2P techniques. Every peer has a full-fledged search engine with a (thematically focused) crawler and a local index whose contents may be tailored to the user's specific interest profile. Peers are autonomous and post meta-information about their bookmarks and index lists to a global directory, which is efficiently implemented in a decentralized manner using Chord-style distributed hash tables.

A query posed by one peer is first evaluated locally; if the result is unsatisfactory the query is forwarded to selected peers. These peers are chosen based on a benefit/cost measure where benefit reflects the thematic similarity of peers' interest profiles, derived from bookmarks, and cost captures estimated peer load and response time. The meta-information that is needed for making these query routing decisions is efficiently looked up in the global directory; it can also be cached and proactively disseminated for higher availability and reduced network load.

Strategies for P2P request routing beyond simple key lookups but without considerations on ranked retrieval have been discussed in [174], [33], [31], but are not directly applicable to our setting. The construction of semantic overlay networks is addressed in [80], [34] using clustering and classification techniques; these techniques would be orthogonal to our approach. [126] distributes a global index onto peers using LSI dimensions and the CAN distributed hash table. In this approach peers give up their autonomy and must collaborate for queries whose dimensions are spread across different peers. [2] addresses the problem of building scalable semantic overlay networks and identifies strategies for their traversal.

In addition to this recent work on P2P Web search, prior research on distributed IR and metasearch engines is potentially relevant, too. [26] gives an overview of algorithms for distributed IR like result merging and database content discovery. [56] presents a formal decision model for database selection in networked IR. [93] investigates different quality measures for database selection. [65], [82] study scalability issues for a distributed term index.

A good overview of metasearch techniques is given by [84]. [168] discusses specific strategies to determine potentially useful local search engines for a given user query. Notwithstanding the relevance of this prior work, collaborative P2P search is substantially more challenging than metasearch or distributed IR over a small federation of sources, as these approaches mediate only a small and rather static set of underlying engines, as opposed to the high dynamics of a P2P system.

1.3 Grid Infrastructure

A Grid infrastructure is a hardware and software infrastructure that concern with "*coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organization*" [54]. The key concept is *sharing* and resources subject to sharing are computing and storage devices, software, data, services and in general each kind of networked resource usable in a remote way. This sharing is necessarily highly controlled as providers and consumers want to clearly define what to share, who is allowed to share, and the conditions under which sharing occurs. A Grid infrastructure aims to supply the ability to negotiate resource-sharing rules among a set of providers and consumers and then to use the resulting pool of resources for some purpose.

In order to better understand the notion of Grid infrastructure it is important to take a look at its history. As reported in [18], three generation of the Grid systems can be identified. The first generation, at that time termed *metacomputing* [121], dated from the early to mid 1990s, puts the effort on linking supercomputing sites in order to provide computational resources to high performance applications.

The second generation views the Grid as a viable distributed infrastructure on a global scale that can support diverse application requiring large-scale computation and data. This view introduces three main issues: (a) *heterogeneity*, resources are heterogeneous in nature and span numerous administrative domains, (b) *scalability*, the number of available resources

increases and they are also geographically distributed, this mean that the applications must be latency tolerant in order to do not degrade their performance, and (c) *adaptability*, the infrastructure is highly dynamic, resource failure is quite common, this mean that applications must be able to extract the maximum performance from the available resources. In this second generation Grids attention is also dedicated to the *middleware* required to support this vision. Middleware has to be intended as a means for hiding the heterogeneous nature of the resources, it aims to provide an environment enabling access and use, via standardized interfaces, to a variety of them. During this phase many noteworthy projects, dealing with various aspects of the Grid, have been undertaken. The most well known among these projects is the *Globus* [61]. The toolkit developed by this project, which has been used by several other projects, provides a collection of solutions to problems that frequently come up when trying to build collaborative distributed applications. It has evolved from its first version [51] to the current forth version. It offers building blocks and tools for application developers and system integrators that are related with:

- *Security*: mainly related with *authentication*, *authorization* and *delegation* issues. It is based on three components: *WS Authentication and Authorization*, *Community Authorization Service (CAS)* and *Delegation Service*;
- *Data Management*: mainly related with *data transfer* (*GridFTP* and *Reliable File Transfer (RFT)*) and *management of mapping information from logical names for data items to target names* (*Replica Location Service (RLS)*). It will also include the *OGSA-DAI* [95] component, i.e. a component allowing data access to relational databases and XML repositories;
- *Execution Management*: it is based on the *Grid Resource Allocation and Management (GRAM)*, a service that provides a single interface for requesting and using remote system resources for the execution of jobs;
- *Information Services*: the main component is the *Monitoring and Discovery System (MDS)*. It provides information about the available resources on the Grid and their status;
- *Common Runtime Components*: provides tools for building stateful Web services in three programming languages: Java, C, and Python.

During this phase, several projects dealt with resource brokering and scheduling. Among them we like to cite *Condor* [32], i.e. a software system for executing batch jobs on a variety of UNIX platforms with strong fault tolerant mechanisms (checkpoint and migration of jobs), and *Storage Resource Broker* [122], i.e. a middleware able to provide uniform access to distributed storage resources across a range of storage devices, via a well-defined API. Other project tried to integrate these components into coherent systems. For example, the European DataGrid project [38], aimed to set up a computational and data-intensive Grid of resources for the analysis of data coming from scientific exploration. Many of the products (technologies, infrastructure etc.) of this project are going to be included in a new EU grid project - “Enabling Grids for E-science in Europe” (*EGEE)* [44]. EGEE aims to build on recent advances in grid technology and to develop a service grid infrastructure in Europe which will be available to scientists 24 hours-a-day.

The third generation Grid systems, currently under development, move the emphasis on distributed global collaboration following a service oriented approach and paying attention to information layer issues. While the first two generation systems can be described in terms of large scale data and computation, here the focus is really on *virtual organizations* and *distributed collaboration*, i.e. distributed and loosely coupled users and resources will be

enabled to group together in order to solve the new kinds of problems that the society have to deal with.

Even if the past generation systems have proposed different Grid architectures e.g. the layered architecture described in [54], it is now wide accepted that the service-oriented approach is the most appropriate paradigm for the actual generation of Grid systems. This approach is confirmed in [53], the paper introducing the *Open Grid Service Architecture (OGSA)*. The key concept in this architecture is the *Grid Service*, i.e. a network-enabled entity having a well defined semantics in terms of mechanisms for *dynamic service creation*, *lifetime management*, *notification*, *manageability*, *naming* and *discovering* of services instances. Computational resources, storage resources, networks, programs, databases, and the like are all represented as services.

In July 2003 the *Open Grid Services Infrastructure (OGSI)* specification was released by the OGSI Working Group with the objective of defining a set of conventions and extensions on the use of Web Service Definition Language and XML Schema to enable stateful Web services. OGSI is set of WSDL specifications defining standard interfaces, behaviours, and schema for Grid computing consistent with the OGSA vision. It introduces the idea of stateful Web services and defines approaches for creating, naming, and managing the lifetime of instances of services; for declaring and inspecting service state data; for asynchronous notification of service state change; for representing and managing collections of service instances; and for common handling of service invocation faults. These interfaces and behaviors define the *Grid Service*.

Since the start of the OGSA development in early 2002 the world of Web services evolved significantly by the emergence of a set of new Web service standards. In addition, since the publication of OGSI 1.0, fierce discussion on the applied techniques between the Web service and Grid service communities took place. In January 2004, experts from the Web Services community proposed the *WS-Resource framework (WSRF)* [36] as a re-factoring and evolution of OGSI aimed at exploiting new Web services standards, specifically *WS-Addressing*. WSRF retains essentially all of the functional capabilities present in OGSI, while changing some of the syntax and also adopting a different terminology in its presentation. In addition, WSRF partitions OGSI functionality into six distinct, composable specifications that are fully compatible with the existing established Web service specifications and concepts. With WSRF the concept of a Grid service has not disappeared. The WS-Resource construct defines creation, addressing, inspection and lifetime management of stateful resources, so-called *WS-Resources*. It defines the relationship between Web services and stateful resources in terms of the implied resource pattern that is built on Web service conventions. WSRF models stateful resources with Web services as a stateless service that acts upon stateful resources, it provides access to or manipulates a set of logical stateful resources (documents) based on messages that it sends and receives. Both OGSA [53] and WSRF [36] will be supported by the last release of the Globus Toolkit (GT4) announced for the beginning of 2005.

The experience done with OGSI and WSRF convinced the Grid community on using as much as possible *standards*. Setting and using standards is a key to tackling heterogeneity and encourage tooling and code re-use. In WSRF view Grid Services are naturally and critically tied to Web Services and so must be built on top of Web service standards. However, there are about 60 active WS-* specifications that represent critical features of Grid systems in various areas[1]:

[1] For each are we will just cite the involved specifications.

- *Core Infrastructure Specifications* (XSD [173], WSDL [164], SOAP [118]);
- *Service Discovery* (UDDI [129], WS-Discovery [144], WS-IL [149]);
- *Security* (SAML [107], XACML [170], WS-Security [158], WS-SecurityPolicy [160], WS-Trust [162], WS-SecureConversation [159], WS-Federation [146]);
- *Messaging* (WS-Addressing [134], WS-MessageDelivery [152], WS-Routing [157], WS-RM [156], WS-Reliability [155], SOAP MTOM [119]);
- *Notification* (WS-Eventing [145], WS-Notification [137], JMS [72]);
- *Workflow and Coordination* (WS-CAF [139], WS-CTX [143], WS-CF [140], WS-TXM [161], WS-Coordination [142], WS-AtomicTransaction [136], WS-BusinessActivity [138], BTP [94], BPEL [24], WS-Choreography [141], WSCL [163]);
- *Characteristics* (WS-Policy, WS-Agreement);
- *Metadata and State* (RDF [103], DAML+OIL [37], OWL [100], WS-DistributedManagement [165], WSDM-MUWS [91], WSDM-MOWS [90], WS-MetadataExchange [153], WS-RF [36], ASAP [10], WS-GAF [147]).

It is of vital importance to keep order and harmonization among all these specifications. The Web Services Interoperability Organization [150] is an open industry effort chartered to promote Web Services interoperability across platforms, applications, and programming languages. This organization brings together a diverse community of Web services leaders to respond to customer needs by providing guidance, recommended practices, and supporting resources for developing interoperable Web services. One of the results of this organization is the *profile*, i.e. sets of Web services specifications that work together to support specific types of solutions. The *WS-I Basic Profile 1.1* incorporates just XSD, SOAP1.1, WSDL1.1 and UDDI. It is probable that the 60 specifications will be checked out, evolved in the cauldron the real word and best practice will identify new specification to be added to WS-I profile.

Related to standards, there is another interesting initiative that has been undertaken by the Open Middleware Infrastructure Institute (OMII) of the University of Southampton. In its strategy paper [9] the promoters of the initiative defines a web service specification profile *WS-I+* that builds upon the recognized *WS-I Basic Profile* adding some specifications: WS-Addressing, WS-ReliableMessaging and the BPEL.

Finally, it is important to notice that Grid services must be able to communicate with other services but also with human users. So, component models for resources automatically lead to component models for the user interfaces. It is a good practice, also supported by existing tools (e.g. GridSphere [64], Jetspeed [71]), to build the user interface of grid applications with portals and portlets. Behind this models there are two noteworthy standards: the WSRP [167] and the JSR168 [73].

Part 2: Digital Library Support

This part surveys current activities, projects, etc. that aim at applying service-oriented architectures, peer-to-peer architectures and/or grid infrastructures to Digital Libraries and Digital Library Management Systems. Section 2.1 starts with an overview on relevant DL repositories. The following three sections (2.2 – 2.4) then report on ongoing activities (mostly of DELOS WP 1 member institutions, but also from others in case detailed information was accessible) to use SoA, P2P, and Grid concepts and systems for Digital Libraries.

2.1 Overview on DL Repositories

This section introduces several *repository* systems. A repository is a central place where data is stored and maintained. For the systems presented in this section, the repository software is released as *open source* product. Moreover, it has been equipped with some DL functionalities, mainly the search feature and a user interface allowing have access to the stored documents. Thanks to these features and characteristics, they are considered powerful enough to meet the DL requirements of a number of communities and thus are usually confused with DLMSs. In particular, they are used to implement *Institutional Repositories*. An Institutional Repository is defined as a system providing a set of services to the members of its community for the management and dissemination of digital materials created by the institution and its community members (e.g., "A university-based institutional repository is a set of services that a university offers to the members of its community for the management and dissemination of digital materials created by the institution and its community members. It is most essentially an organizational commitment to the stewardship of these digital materials, including long-term preservation where appropriate, as well as organization and access or distribution" [78]). We are firmly convinced that DLMSs are more than simple repositories and are capable of improving and enhancing the services offered by an Institutional Repository. However the software systems that are actually used represent a reality w.r.t. DL researchers and systems have to compare with.

DSpace (MIT Libraries and Hewlett-Packard)

DSpace [42], [127] is an open source system designed to operate as a centralized repository able to capture, store, index, preserve, and redistribute the intellectual output of a university's research faculty in digital formats. It manages and distributes digital items, made up of digital files and allows for the creation, indexing, and searching of associated metadata to locate and retrieve the items. It is designed mainly to support the long-term preservation of the digital material stored in the repository.

From an architectural point of view the system is not designed to deal with any of the distributed architectural framework object of this survey. Instead it is a centralized software system organized into three layers: i.) the storage layer, which is responsible for physical storage of metadata and content. It relies on the file system of the server to store the content and on a RDBMS to store all information about the organization of content, metadata about the content, information about users and authorization, and to maintain indices that users can browse; ii.) the business logic layer deals with managing the content of the archive, users of the archive, authorization; and iii.) the application layer contains components that communicate with the world outside of the individual DSpace installation, for example the Web user interface and the Open Archives Initiative Protocol for Metadata Harvesting service.

EPrints (University of Southampton)

EPrints [45] is a software tool, released as open source in 2000, that can be used for creating a web-based archive/repository of files with associated metadata. The most common use for EPrints is thus to enable the creation of a web accessible repository of some, or all, of an institution's research. In our best knowledge, on July 2005, there are 161 repositories running EPrints software spread worldwide for a total of 86'609 records.

From an architectural point of view this system is designed as a centralized service to be hosted on a single server. From a functional point of view, it offers similar features to those presented for DSpace, e.g. submission, search, and browse.

Fedora (University of Virginia Library and Cornell University)

Fedora [48], [77] is an open source repository service for storing and managing complex objects. At its core there is a powerful document model. In accordance to this model a Fedora digital object is composed by i.) a unique identifier, ii.) a set of descriptive properties, iii.) a set of data streams, and iv.) a set of disseminators. Data streams are containers used to maintain both data and metadata belonging to an object. Disseminators are components capable to associate an external service with the object in order to supply a virtual view of the object itself, or of its data stream content. Thanks to the richness and flexibility of this model many institutions are nowadays using the Fedora system.

From the perspective of this report, one of the most important features of Fedora is that it is implemented as a set of web services and its full functionality, including its rich document model, is exposed through well-defined web service APIs. Thanks to this feature, Fedora is particularly appropriate to be used in a broader service oriented framework and act as the storage layer for a variety of applications. This distinguishes Fedora from other repository systems that are vertical applications for storing and manipulating complex objects through a fixed user and management interface like DSpace and EPrints.

The Fedora architecture is composed by four services, the *Fedora Repository service* represents the core one around which other services providing additional functionality exist, i.e., the *Fedora OAI Provider*, the *Fedora Search service*, and the *Fedora Preservation Monitoring service*. In our best knowledge, at the time of writing Fedora is migrating to this new service oriented framework and new versions of the Fedora OAI Provider and Fedora Search services will be release in Fedora 2.1 while the latter one will be implemented as part of the phase II of the Fedora project. Instead the Fedora Repository service is available and exposes API for i.) read/write operations necessary to manage a repository of complex digital objects, ii.) read-only operations for accessing complex digital objects, and (iii) discovery capabilities to locate digital objects via a simple search on the object properties or a browsing of an RDF based index of the entire repository content.

Greenstone (University of Waikato)

Greenstone [62] is a suite of software for building and distributing digital library collections that provides a way of organizing information and publishing them on the Internet or on removable media like CD-ROM and DVD. A liaison with UNESCO and Human Info has been a crucial factor in the development of Greenstone. Human Info began using Greenstone to produce collections in 1998, and provided extensive feedback from user testing. UNESCO wants to empower developing countries to build their own digital library collections and selected Greenstone in 2000, arranges user testing, helps with internationalization, and

mounts courses. Internationalization is another central goal, at the writing time the user interface is available in 35 languages.

From the architectural point of view, a precise distinction must be done w.r.t. the different versions of this software. The last version, Greenstone 3 [11], is a complete redesign and reimplementation of the original Greenstone digital library software needed to overcome the problem of it, e.g. lack of flexibility and expandability. The new version is designed with the goal to meet the following requirements: backward compatibility w.r.t. collections, different levels of customization, software modularity, service based, distributed architecture, future compatibility, dynamic, etc. Worth noting two concepts: it is planned to use decoupled services and a distributed architecture. At the writing time exists a very early pre-release of this software and thus the information to make an in depth evaluation are not available, however the premises seems reasonable.

2.2 Service-oriented Architectures for Digital Libraries

This section presents contributions to the application of service orientation to Digital Libraries at different levels of abstraction. First, the architecture of a particular service for enriching DL content by means of annotations is discussed. The following two approaches address the overall architecture of a service-oriented DL (Knowledge Management DL, BRICKS, and OpenDlib). Finally, the ETHZ/UMIT hyperdatabase approach supports the combination of existing services by means of processes, thereby allowing for the creation of new (DL) functionality.

Annotation Services for Digital Libraries (Uni Padova)

The notion of isolated applications or data is increasingly disappearing in favour of a distributed and networked environment with an information centric view. This allows us to provide integrated services and applications to users, without any distinction between local and remote information resources.

In this context we can envisage a scenario in which a digital library system can become not only a place where information resources can be stored and made available, but also a daily work tool, which can be integrated into the way the user works, so that the user's intellectual work and contents which are provided by the digital library can be merged together, constituting a single working context. Thus the digital library is no longer perceived as something external to the intellectual production process or as a mere consulting tool but as an intrinsic and active part of the intellectual production process [4], [5], [6].

Annotations are effective means used in enabling this paradigm of interaction between users and digital libraries, in fact annotations introduce a new content layer devoted to elucidate the meaning of an underlying information resource and they can make hidden facets of the annotated information resource more explicit. In particular, annotations allow users to naturally create a hypertext that seamlessly merges personal contents with the contents provided by the digital library. So, to give to the final users the possibility of dynamically developing a hypertext of information that annotates the documents maintained in the digital library of their interest, it becomes necessary to design an annotation service able to cooperate with the digital library system. Architectural choices become a key factor for enabling the design and development of an advanced annotation service capable of both modelling the different facets of the annotation and effectively exploiting annotations for search and retrieval purposes.

In fact it is necessary to have an architecture able to support both the behaviour of the annotation service in a modular way, so that we can easily add new functionalities to the annotation service without the need of redesigning the architecture of the it, secondly, the architecture has to be flexible enough to be implemented according to different architectural paradigms, such as Web Services (WS) or Peer-to-Peer (P2P) architectures. Indeed a flexible architecture allows the annotation service to have a great reach and a widespread usage, so that users can benefit from its functionalities without limitations due to the architecture of a particular digital library system, allowing a strict interaction between users and digital libraries.

A Generic Service-Oriented Architecture for Knowledge Management (TU Crete)

TUC is currently working on the development of a generic Service-Oriented Architecture for Knowledge Management in a distributed environment. The developed system is general-purpose ontology-based P2P meta-data management system and its architecture can be used in many different environments and application domains. The causation for building this system is to enable knowledge management and sharing in an e-commerce environment where thousands of companies offer and demand business services (web services) forming a *digital business ecosystem*. The generic approach followed in this P2P metadata management middleware that TUC is developing, can also be used in digital libraries in order to provide distributed, advanced metadata management and to put into action business models that exploit semantic ontologies. The entire system is based on the OMG MOF Metadata Architecture, and the core services that it offers include:

- *KB Service*: It encapsulates all the functionality Knowledge Base and provides a standard interface to the other service components.
- *Recommender Service*: It is responsible of handling user preferences in terms of partnerships as well as services needed to compose more complex services.
- *Semantic Registry Service*: It provides a standard interface capable to implement Semantic Registry Service functionality and provides standard representation hierarchies and query facilities provided by the standard registries.
- *Ontology Manager*: Provides a GUI to the end user and it is used for the management (creation/update/retrieve) of the ontologies.
- *BML Editor*: It is a Tool that uses the Business Ontologies (created with the Ontology Editor) in order to describe business models, policies, assets, competencies, partners, etc.
- *Service Manifest Creator*: It is a Tool that is used to integrate the semantic (business) description and technical description (interfaces) of services into a single description container named Service Manifest.
- *Service Publisher*: It is a tool that is used to publish Service Manifests to the Semantic Registry.
- *Service Browsing/Discovery Tool*: It is a Tool that is used to contact the Semantic Registry Service in order to browse and retrieve the contents of the Semantic Registry of the DBE.
- *User Profile Editor*: A graphical tool that is used by the users in declaring their preferences. Appropriate guidance is also given to the users with respect to the specific business domain by exploiting domain specific ontologies.

BRICKS: Building Resources for Integrated Cultural Knowledge Services

The aim of the BRICKS project (an FP6 integrated project) is to design and develop an open, user- and service-oriented infrastructure to share knowledge and resources in the Cultural Heritage domain [25]. This project began in January 2004, has a duration of 42 months and involves 24 partners: 7 from academia, and the rest equally distributed between users and industry.

From an architectural point of view, it has been decided that the BRICKS architecture will be decentralised, based on a P2P paradigm, i.e. no central server will be employed. In particular, the BRICKS P2P network will utilize the P-Grid [1] distributed hash table approach (P-Grid DHT). Every institution joining a BRICKS installation is a node (a BNode in the BRICKS jargon) of the BRICKS architecture. Each of the components constituting a BNode (the bricks into BRICKS jargon depicted as vertical boxes in the picture) is a Web Service. These bricks are classified into three categories:

- *Fundamental bricks*, i.e., services hosted on each BNode ensuring the proper functioning of the node as member of BRICKS. The functionalities they provide are: Decentralized XML Storage, Service Registration and Discovery, Index management.
- *Core bricks*, i.e., services needed to provide local user of the BNode to have access to BRICKS. They include User Management, Authentication and Authorization, Search and Browse.
- *Basic bricks*, i.e., optional services that are deployed on a BNode if the functionalities they provide are needed on the single node. They include: Content Management, Metadata Management, Accounting, IPR Protection, Annotation Management, and Service Composition.

OpenDLib: a Digital Library Management System (CNR-ISTI)

OpenDLib [97] is digital library management system, i.e. a system able to support a cost-effective digital library creation and operational model. From an architectural point of view it consists of an *open federation of services* that can be distributed and replicated on the pool of servers belonging to the supporting institutions.

The OpenDLib system is able to grow over time along several dimensions, e.g. services, metadata formats supported, host servers, user communities, searchable metadata, handled manifestations, etc. In particular, it supports three kinds of dynamic service expansions: (i) new services can be added; (ii) new instances of a replicated or distributed service can be mounted on either an existing or a new hosting server; (iii) the configurations of the services can be modified so that they can handle new document types, new metadata formats and support new usages.

The interaction among the OpenDLib services is more complex than a simple client-server communication. A service can act both as a provider and as a consumer, and sharing relationships exists a priori among a subset of the services. Moreover, the topology of the communication among the different service instances allocated on different servers is dynamic since it takes into account load balancing and bandwidth monitoring techniques.

These features of the architecture provide a great flexibility in the management of a digital library. For example, an institution can decide to maintain an instance of a repository service in order to locally control its own documents and to share all the other services with other institutions, a new index service can be added to support another language; a new query mediator service can mounted to better support an enhanced workload.

Hyperdatabases for Service Composition & Process Management (ETH Zürich/UMIT)

The *hyperdatabase vision* [108], [109], [110] was established at ETH Zürich several years ago as an answer to the substantial changes in IT technology and its impact on information systems as well as an answer to what extent traditional database technology could lead to a new and a more radical departure from traditional existing information infrastructure and middleware. While a database system handles data records, a hyperdatabase system deals with services and service invocations. Services in turn may be using a database system. Therefore, the name hyperdatabase, i.e., a software layer for services on top of databases, has been given to this vision. In short, a hyperdatabase (HDB) takes care of optimal routing similar to query optimization in a conventional database and it provides process support with transactional guarantees over distributed components using existing services as a generalization of traditional database transactions [112], [113]. By using processes, existing services can be combined (by defining control and data flow dependencies between them), thereby implementing new, advanced services. The HDB provides sophisticated routing strategies to dynamically choose among the available providers of services at run-time using approximate knowledge about availability and load.

Most importantly and in contrast to traditional database technology, a hyperdatabase does not follow a monolithic system architecture but is fully distributed over all nodes representing peers in a network of a community. Every node is equipped with an additional thin software layer, a so-called hyperdatabase layer (HDB layer). The HDB layer extends existing layers like the TCP/IP stack with process related functionalities, e.g., routing of requests.

OSIRIS (Open Service Infrastructure for Reliable and Integrated process Support) is a hyperdatabase implementation which has been started at ETH Zürich and which is now jointly continued at ETH Zürich and at UMIT. Process management is vital to Digital Libraries in order to combine and integrate services to a coherent whole, i.e., to access information from different content sources, to provide advanced content-based search functionality within a DL, or to transform the retrieved data into a user desired format [87], [169].

2.3 Peer-to-Peer Architectures for Digital Libraries

In what follows, the approaches presented use to variants of P2P architectures for the realization of DL management systems. First, these are super peer architectures that contain selected, specialized peers that manage sets of “normal” peers (the OFFIS super peer network and the TUC Knowledge Management approach). Second, MINERVA, GHT*, and OSIRIS consider P2P architectures with all equal peers. MINERVA is a P2P search engine that supports federated search over digital libraries and other information sources. GHT* addresses special queries (range and k-nearest neighbours queries) on metric space data. Finally, OSIRIS uses P2P data management for distributed process management, i.e., the execution of process-based DL services without any centralized control.

Super Peer Networks (OFFIS Oldenburg)

The research of OFFIS focuses on super peer networks. Super peer networks have some advantages in comparison to pure peer-to-peer networks. They combine the efficiency of the centralized client-server model with the autonomy, load balancing, and robustness of distributed search. They also take advantage of the heterogeneity of capabilities across peers. A super peer can independently route messages within its cluster. Queries to selected

organizational units do not flood the entire network, but can be routed directly. The hierarchical super peer network supports the flexibility and self-organization of widely distributed, loosely coupled, and autonomous digital library systems [21]. The architecture allows for the search over collections of arbitrary artefacts as for example traditional documents, on-line books, digital images, and videos, which is a basic service requirement for digital libraries. Beyond, the network enables library users to also store, administer, and classify their own artefacts. Thus, it supports scenarios like the construction of personal or group reference libraries and collaborative authoring. Other application areas for hierarchical super peer networks are the medical sector in order to solve the availability problem for distributed patient records [19] and the support of distributed software development [20].

P2P Knowledge Management (TU Crete)

The previously described Knowledge Management system is under development at TU Crete and it will be built with the principles of Service Oriented Architecture in mind. That is, the various back-end components (KB, Recommender, etc.) will be provided also as offered services to the users of the system. The system is currently extended from a centralized implementation to a P2P one. Some of the research issues related to the P2P Knowledge Management that are being examined are the following:

- Ontology management (insertion, maintenance, conflict resolution and utilization) in P2P systems combining MOF Repositories and Relational Databases at each peer following a Service Oriented Architecture.
- Business model and business process ontologies, environmental ontologies, domain specific ontologies and their use and interplay in a dynamic service environment.
- Distributed Semantic Recommendation and Service Composition mechanisms
- Self-organization of the P2P network.

The architecture follows the super peer network paradigm for the efficiency benefits that it presents and its capability of taking advantage of the heterogeneity of the peers by assigning greater responsibility to those peers that are more capable to handle it. However the choice of the super peer model does not solve all the problems that the Knowledge Management requirements poses, since the design should consider several of challenging issues like: dynamic self-organization of peers and super peers, performance trade offs, load-balancing among equivalent peers and among simple peers and super peers, avoidance of single-point of failure in the super peers, search performance using super peers, data placement and indexing across super peers and other research issues. The on-going research will address the above issues and will be based on state of the art semantic models as well as data representation interchange standards: OMG’s Model Drivel Architecture (MDA) that provides an open, vendor-neutral approach to the challenge of interoperability, building upon and leveraging the value of OMG's established modelling standards as well as the Unified Modelling Language (UML), Meta-Object Facility (MOF), XML Metadata Interchange (XMI) etc.

The Minerva P2P Search Engine (MPII)

The Minerva project [15], [16], [17] at the Max-Planck Institute of Computer Science pursues a P2P architecture for federated search over digital libraries and other information sources as well as Web search for advanced information demands. Each peer, for example, a digital library or a power-user's personal agent, is considered autonomous and has its own local search engine with a corresponding local index. Peers share their local indexes (or specific fragments of local indexes) by posting meta-information into the P2P network. This meta-

information contains compact statistics and quality-of-service information, and effectively forms a global directory. However, this directory is implemented in a completely decentralized and largely self-organizing manner. More specifically, we maintain it as a distributed hash table (DHT) using the (re-implemented and adapted) algorithms of the Chord system. Each per-peer engine uses the global directory to identify candidate peers that are most likely to provide good query results. A query posed by a user is first executed on the user's own peer, but can additionally be forwarded to other peers for better result quality. The local results obtained from there are merged by the query initiator.

Particular emphasis is paid to query routing, the decision to which other peers a given search request is forwarded. A "good" peer in this regard should have thematically relevant index contents, which could be measured by statistical notions of similarity between peers. Both query routing and the formation of "statistically semantic" overlay networks could greatly benefit from collective human inputs in addition to standard statistics about terms, links, etc.: knowing the bookmarks and query logs of thousands of users would be a great resource to build on.

A first prototype of the Minerva system is running and serves as an experimental platform for studying query routing strategies and other aspects of P2P information search.

GHT*: A Scalable P2P System (MUNI)

GHT* [12] is a scalable P2P system allowing execution of range and k-nearest neighbours queries on metric space data. The structure distributes data among network peers, utilizing additional peers as the size of the data-set scales up. The response to similarity queries remains practically constant, because the queries are executed in parallel on respective peers. In addition, the latency of the whole system is better as opposed to a centralized metric index, because different peers behave practically independently on each other and queries are solved only on a subset of peers.

In general, GHT* consists of network nodes, peers, that can insert, store, and retrieve objects using similarity queries. The GHT* architecture assumes that:

- Peers communicate through the message-passing paradigm.
- Each peer participating in the network has a unique Network Node IDentifier (NNID).
- Each peer maintains data objects in a set of buckets. Within a peer, the Bucket IDentifier (BID) is used to address a bucket.
- Each object is stored exactly in one bucket.

The GHT* exploits Generalized Hyperplane Trees (GHT) [130], which is a metric space indexing technique for centralized systems. Practically, a modified form of GHT called *Address Search Tree* (AST) is present in every participating peer – the structure is used to navigate the queries to particular peers holding the data. It is a binary search tree, where its inner nodes hold routing information and the leaf nodes represent pointers to the data. Specifically, the inner nodes always store a pair of pivots – these are some representative metric objects from the data-set – and respective pointers to the left and the right subtrees.

The data objects are stored in buckets that are held either locally (thus we can address the bucket by its BID) or on another peer, which can be identified by a proper NNID. Therefore, the AST has always one of those two types of pointers in leaf nodes.

When searching for a place where to store a new object, we start in the root of the AST of the peer that issued the query. We compute distances between the inserted object and the pivots in

inner nodes while traversing the tree using following rule. If the distance to the first pivot is smaller than the distance to the second one, we navigate to the left subtree of that inner node. Otherwise, the right subtree is considered. This process is recursively repeated until a leaf node is reached. Whenever the navigation procedure reaches the leaf node of the AST, the inserted object is stored either locally in the respective bucket (if a BID identifier is found) or on a remote peer (if an NNID identifier is encountered).

By analogy to insertion, the *range search* also starts by traversing the local AST of the querying peer. However, the traversing condition is modified. The specified radius of the range search is used when determining if the left of right subtree has to be considered. As opposed to insertion, both the paths can match and thus both the subtrees must be traversed. We again repeat this procedure recursively until all the matching leaves are found. Then we forward the query to all peers that are identified by NNIDs in the leaves. We execute range query in local buckets for all BID pointers in the leaves. In general, the query is usually forwarded to more peers, where the local buckets are searched. Thus, the query is effectively parallelized on different peers and the results are just concatenated together afterwards. Because the size of a bucket is limited, the parallelization grows with the size of the data-set and, therefore, the response time remains practically constant.

The GHT* structure is also able to perform *k-nearest neighbors* queries. The algorithm for executing kNN queries is based on the range searches. Let *q* be the query object and *k* the number of nearest neighbours to retrieve. The kNN algorithm first traverses the AST using the similar strategy as if it is inserting object *q*. With this strategy, the bucket, where *q* would be inserted, is found. Then, we search all the objects in this bucket and compute distance to the query object. If the bucket contains more than or exactly *k* objects, we get the distance to the k^{th} object and execute a range search with query object *q* and this distance. In the other case, we have to estimate the range search radius by another technique. Algorithms resolving this problem were presented in [13]. The result of the range search is then pruned so that there are exactly *k* objects left – they are the solution of the kNN query.

Peer-to-Peer Process Execution with the OSIRIS Hyperdatabase (ETH Zürich / UMIT)

The decentralized and distributed process engine OSIRIS follows the principles of a hyperdatabase system. The main emphasis of the OSIRIS design was to avoid any central component for process navigation. Rather, a process instance involves only nodes that provide a service for that process. To do so, each HDB layer (i.e., the software layer that has to be available with each service provider) requires global meta information, in particular information about other service providers and their current load. Therefore, the efficient replication of global meta information is important. Essentially, global repositories maintain the global meta information about the service providers in the community. Each HDB layer replicates those pieces of meta information it needs to fulfil its tasks.

Process execution in OSIRIS follows a true peer-to-peer approach touching only nodes that provide a service for the process, and accessing meta information only locally [114], [115]. Meta data replication, on the other hand, is based on a hierarchical organization with central repositories (although distributed over a set of nodes), and clients (= HDB layers) replicating from them. Most importantly for true peer-to-peer process management, process execution and meta data replication run independently from each other.

Usually, several providers offer semantically equivalent services. To simplify the discovery of services, OSIRIS deploys a publish and subscribe (pub/sub) mechanism for service

invocations: a service provider subscribes for the execution of its services at a global service repository (similar to a UDDI repository). If a client requests a service, it publishes this request with the service type as the topic, and the local HDB layer selects and invokes one of the available services. In OSIRIS, this means that a process instance is migrated by publishing the instance data with the service topics of subsequent steps. Essentially, there is no central pub/sub component routing publications. Rather, each hyperdatabase layer holds local replicas of the global subscription lists and migrates process instances in a peer-to-peer way, i.e., plays the role of a pub/sub broker.

2.4 Grid Infrastructures for Digital Libraries

The application of Grid infrastructures for DLs either considers i.) the "gridification" of selected DL functionality (the TUC approach to index and access 2D and 3D data in a grid), ii.) the application of grid features like on-demand deployment of services and load balancing to the management and execution of DL processes in the OSIRIS hyperdatabase approach, and iii.) the implementation of a DL on top of a Grid environment as it is done in the DILIGENT project.

Grid Infrastructures for Scientific and Engineering Applications (TU Crete)

TUC is currently surveying the state of the art in the field of exploiting Grid infrastructures for supporting scientific and engineering applications based on big digital libraries of 2D and 3D content that are stored on the Grid. The main areas of research interest include:

- Semantic modelling of 2D and 3D objects using domain specific ontologies that capture the semantics of specific application domains (e.g. meteorology, engineering, health) including the modelling of simulation environments.
- Query and manipulation languages for 2D and 3D objects that are exploiting the semantics of the content.
- Semantic indexing of the content to provide efficient access and retrieval of 2D and 3D objects that reside on the Grid.

OSIRIS: A Grid-Enabled Hyperdatabase System for Processes (ETH Zürich / UMIT)

The OSIRIS hyperdatabase implementation supports several grid features, especially in terms of sophisticated routing of service requests, load balancing, and service management.

The OSIRIS load balancer chooses the provider that is best suited to offer the service to be invoked next in the context of a particular process instance. For that purpose, the Load Balancing module replicates data from the global Load Repository via the Replication Manager. Since load information is not critical, approximate load information at each peer about other service providers is sufficient. Therefore, less strict freshness predicates can be used as in the case of replicating process information (i.e., information about the next service to be executed in the context of a process). The use of a lazy replication approach for load balancing information is a key characteristic of OSIRIS and important to achieve a high degree of scalability in decentralized peer-to-peer process execution [114], [115].

OSIRIS also supports the management of executables of services. When no or only highly loaded providers for a certain service exist, a particular service instance can be installed on a less loaded node in the network (given that this node has the required processing, network,

and/or storage capabilities), thereby also increasing the degree of scalability that can be achieved for decentralized peer-to-peer process execution.

DILIGENT: a DIgital Library Infrastructure on Grid ENabled Technology (CNR-ISTI)

The DILIGENT project [41] aims to create a knowledge infrastructure that will allow members of dynamic user communities to build *on-demand transient digital libraries* (DLs) capable to satisfy their needs. These DLs will be created by exploiting shared resources, i.e. content repositories, applications, services, storage and computing elements, etc, offered by the infrastructure.

From an architectural point of view the DILIGENT infrastructure will be composed of a set of interacting services providing: i) a set of typical DL functions, like search, annotation, personalization, document visualization; ii) access to information sources and applications provided by third-parties; iii) features necessary for handling the shared content and application resources; and iv) support for the creation and operation of on-demand, transient DLs. These services will exploit the high computational and storage capabilities of the Grid infrastructure released by the EGEE project [43] in order to support complex and time consuming functionalities, while focusing on optimizing resource usage and satisfying QoS contracts.

The DILIGENT services will be logically partitioned into three layers:

- *DILIGENT Collective Layer*. This layer is composed by services that enhance existing Grid collective services, i.e. those global services needed to manage interactions among resources, with functionalities capable to support the complex services interactions required by the Digital Library Layer. It will consists of (a) an Information Service for discovering and monitoring distributed resources, (b) a Broker&Matchmaker Service for obtaining an optimal distribution of services and resources across the infrastructure, (c) a Keeper Service for bridging together the set of resources belonging to a DL and (d) a Dynamic VO Support Service for the creation of the operational secure context associating users, user requests and resources belonging to the DL according to the sharing policies and agreements.
- *Digital Library Layer*. It consists of a set of reliable and dependable services covering the core functionalities required by DL applications. This set provides submission, indexing and discovery of mixed-media objects and the management and processing of these objects through annotation, composition, cooperative editing, etc. Each service of this area will likely represent an enhancement of the functionalities provided by the equivalent no-Grid-aware service and will be designed to take full advantage of the scalable, secure, and reliable Grid infrastructure.
- *Application-Specific Layer*. This layer contains the set of services provided by user communities that decided to share their legacy content and application-specific resources.

3 Discussion and Open Issues

Service-oriented architectures, peer-to-peer architectures and Grid infrastructures provide powerful support for the realization of next generation DLs.

Service-oriented architectures SoA are more and more becoming the core backbone of next generation DLs and DL management systems. In addition, the definition, invocation, and

description of services (service directories) as well as the possibility to build sophisticated applications by means of service composition will significantly support DL applications. However, in addition to the underlying SoA-based DL infrastructure, significant work needs to be done in the context of defining and implementing self-contained, modular DL services that can be re-used in different DL applications.

Peer-to-peer architectures allow for a complete decentralization, i.e., DLs that do neither have any global control nor any kind of censorship. Essentially, P2P aspects are applied to data and service management. In addition, P2P data management provides self-organization capabilities that are highly important in the presence of many failures and that allow to address the high dynamics of large-scale DLs.

Grid infrastructures, finally, support the efficient use of resources by means of automatic service installation and deployment (self-adaptability), load balancing, and scheduling. Due to dynamic replication of content and services, a high degree of availability can be achieved. In addition, based on the features of existing Grid infrastructures, authentication and authorization can be seamlessly integrated into a grid-enabled DL.

Despite of the separate introduction and discussion of the three technologies, there are many similarities and analogies. First, all three approaches are distributed ones that make use of functionality offered by different providers and that make this distribution transparent to the user. Second, from a DL point of view, they support in a way or another the notion of sharing/reuse (of content, services, etc.). Third, although we have independently discussed service-oriented architectures, P2P architectures and Grid infrastructures, it has to be noted that these are by far no orthogonal technologies. Rather, all three strongly converge and differences between technologies more and more diminish. This convergence is mostly driven by applying/extending existing standards towards the common notion of Web services. Current tends in Grid infrastructures, for instance, make intensive use of service-oriented concepts (service grids actually apply web service standards for the definition, description, etc. of grid services). Similarly, P2P approaches are more and more penetrating (meta) data management in Grid infrastructures.

While most requirements that have been identified in the introduction can be directly supported by either of these technologies, there are still requirements that are not directly covered by these technologies and require further consideration. This either calls for the implementation of specialized (grid-aware, P2P) services (e.g., context- and location-aware services, complex queries across media types, etc.) or requires orthogonal extensions (e.g., support for highly dependable and reliable systems, continuous processing of sensor data, monitoring of users, or support for mobile devices that allows switching between connected/disconnected modes, etc.).

Since different DL applications have different requirements, there is not a universal DL architecture that supports everything, nor is there a unique recipe on how to construct a novel DL management system. Rather, as we have pointed out in this survey, service-oriented architectures, peer-to-peer architectures, and Grid infrastructures, provide the basic building blocks that can be combined based on the concrete requirements of a DL application.

Based on this survey on SoA, P2P and Grid for Digital Libraries, a particularly important next step will be the definition and introduction of a model for DL management systems (*A Reference Model for Digital Library Management Systems*), i.e., a formal and conceptual framework describing the characteristics of these systems. Essentially, this reference model has to identify the different components of a DL management system and to define and specify these components and their interrelations.

References

[1] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, R. Schmidt. *P-Grid: A Self-organizing Structured P2P System*. SIGMOD Record, Volume 32, 2003.

[2] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, T. Van Pelt. *GridVine: Building Internet-Scale Semantic Overlay Networks*. In: Proceedings of the 3rd International Semantic Web Conference, 2004.

[3] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J .R. Douceur, J. Howell, J. R. Lorch, M. Theimer, R. Wattenhofer. *FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment*. In: Proceedings of the OSDI Conference, 2002.

[4] M. Agosti, N. Ferro. *An Information Service Architecture for Annotations*, In: M. Agosti, H.-J. Schek, and C. Türker (Eds), Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation, Pre-proceedings of the 6th Thematic Workshop of the EU Network of Excellence DELOS. Edizioni Libreria Progetto, Padova, Italy, 2004, 115-126.

[5] M. Agosti, N. Ferro. *Annotations: Enriching a Digital Library*, In: T. Koch, I.T. Solvberg (Eds). Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2003), Trondheim, Norway, August 2003. Springer, Berlin/Heidelberg, 2003, 88-100.

[6] M. Agosti, N. Ferro, I. Frommholz, U. Thiel. *Annotations in Digital Libraries and Collaboratories - Facets, Models and Usage*, In: Proceedings of the 8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2004). Springer, Berlin/Heidelberg, Germany, LNCS 3232, 2004, 244-255.

[7] G. Alonso, F. Casati, H. Kuno, V. Machiraju. *Web services - Concepts, Architectures and Applications*, Springer, 2004.

[8] G. Alonso, U. Fiedler, C. Hagen, A. Lazcano, H. Schuldt, N. Weiler: *WISE: Business to Business E-Commerce*. In: Proceedings of the 9th International Workshop on Research Issues in Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE'99), Sydney, Australia, March 1999.

[9] M. Atkinson, D. DeRoure, A. Dunlop, G. Fox, P. Henderson, T. Hey, N. Paton, S. Newhouse, S. Parastatidis, A. Trefethen , and P. Watson. *Web Service Grids: An Evolutionary Approach*. http://omii.ac.uk/WSG/WebServiceGrids.pdf

[10] ASAP: The OASIS Asynchronous Service Access Protocol (ASAP). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=asap

[11] D. Bainbridge, K. J. Don, G. Buchanan, I. H. Witten, S. Jones, M. Jones, M. I. Barr, *Dynamic Digital Library Construction and Configuration*. In Proceedings of 8th European Conference Research and Advanced Technology for Digital Libraries, ECDL 2004, pages 1-13, Bath, UK. LNCS 2004.

[12] M. Batko, C. Gennaro, P. Savino, P. Zezula: *Scalable Similarity Search in Metric Spaces*. In: Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation, Pre-proceedings of the 6th Thematic Workshop of the EU Network of Excellence DELOS, S. Margherita di Pula, Cagliari, Italy (2004), 213–224.

[13] M. Batko, C. Gennaro, P. Zezula: *A Scalable Nearest Neighbour Search in P2P Systems*. In: VLDB Workshop On Databases, Information Systems and Peer-to-Peer Computing, Toronto, Canada (2004).

[14] M. Bawa, R. J. Bayardo Jr., S. Rajagopalan, E. Shekita. *Make it Fresh, Make it Quick - Searching a Networks of Personal Webservers*, In: WWW Conference, 2003.

[15] M. Bender, S. Michel, C. Zimmer, G. Weikum: Towards Collaborative Search in Digital Libraries Using Peer-to-Peer Technology. In: M. Agosti, H.-J. Schek, and C. Türker (Eds), Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation, Pre-proceedings of the 6th Thematic Workshop of the EU Network of Excellence DELOS. Edizioni Libreria Progetto, Padova, Italy, 2004, pages 61-72.

[16] M. Bender, S. Michel, G. Weikum, C. Zimmer: Bookmark-driven query routing in peer-to-peer web search. In: Proceedings of the SIGIR Workshop on Peer-to-Peer Information Retrieval: pages 46-57.

[17] M. Bender, S. Michel, G. Weikum, C. Zimmer: The MINERVA Project: Database Selection in the Context of P2P Search. In: Proceedings of the 11th German Database Conference, Karlsruhe, Germany, 2005

[18] F. Berman, G. Fox, and A. Hey, *Grid Computing: Making the Global Infrastructure a Reality.* John Wiley & Sons, April 2003. ISBN: 0470853190.

[19] L. Bischofs, W. Hasselbring: *A Hierarchical Super Peer Network for Distributed Software Development*. In: Proceedings of the Workshop on Cooperative Support for Distributed Software Engineering Processes (CSSE 2004). Linz, Austria, 2004, September.

[20] L. Bischofs, W. Hasselbring, H. Niemann, H. Schuldt, M. Wurz: *Verteilte Architekturen zur intra- und inter-institutionellen Integration von Patientendaten*. In: Tagungsband der 49. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS 2004) (2004), September

[21] L. Bischofs, W. Hasselbring, J. Schlegelmilch, U. Steffens: *A Hierarchical Super Peer Network for Distributed Artifacts*. In: Pre-proceedings of the 6th Thematic Workshop of the EU Network of Excellence DELOS. S. Margherita di Pula (Cagliari), Italy, 2004, pp. 105-114

[22] P. Boldi and B. Codenotti and M. Santini and S. Vigna. *Ubicrawler: A scalable fully distributed web crawler*. In: Proceedings of the 8th Australian World Wide Web Conference (AusWeb02), 2002.

[23] BPEL4WS: The Business Process Execution Language for Web Services. http://www-128.ibm.com/developerworks/library/ws-bpel/

[24] BPEL: The Business Process Execution Language for Web Services. http://www-106.ibm.com/developerworks/library/ws-bpel/

[25] BRICKS Community. http://www.brickscommunity.org

[26] J. Callan. *Distributed Information Retrieval.* Advances in Information Retrieval. Kluwer Academic Publishers, 2002.

[27] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, M. Shan. *Adaptive and Dynamic Service Composition in eFlow*. In B.Wangler and L. Bergman, editors, Advanced Information Systems Engineering, Proc. of the 12th Int. Conf., CAiSE 2000, Stockholm, Sweden, June 5–9, 2000, Lecture Notes in Computer Science, Vol. 1789, pages 13–31. Springer-Verlag, Berlin, 2000.

[28] E. Cerami. *Web Services Essentials*. O'Reilly, 2002.

[29] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 2002.

[30] Q. Chen, U. Dayal. *A Transactional Nested Process Management System*. In: Proceedings of the 12th International Conference on Data Engineering (ICDE'96), pages 566-573, New Orleans, Louisiana, USA, 1996.

[31] E. Cohen, A. Fiat, H. Kaplan. *Associative Search in Peer to Peer Networks: Harnessing Latent Semantics*. In: Proceedings of the IEEE INFOCOM'03 Conference, 2003.

[32] Condor: The Condor® Project Homepage. http://www.cs.wisc.edu/condor/

[33] A. Crespo, H. Garcia-Molina. *Routing Indices for Peer-to-Peer Systems*. In: Proceedings of the 28th Conference on Distributed Computing Systems, 2002.

[34] A. Crespo, H. Garcia-Molina. *Semantic Overlay Networks for P2P Systems*. Technical report, Computer Science Department, Stanford University, October 2002.

[35] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen: *PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities*. In: Proceedings of the 12th International Symposium on High Performance Distributed Computing (HPDC), 2003.

[36] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukin, D. Snelling, S. Tuecke, and W. Vambenepe. *The WS-Resource Framework*. White paper, 2004.

[37] DAML+OIL: The W3C DAML+OIL Reference Description. http://www.w3.org/TR/daml+oil-reference

[38] DataGrid: The DataGrid Project website. http://www.eu-datagrid.org

[39] DECmessageQ. http://h18000.www1.hp.com/info/SP3409/SP3409PF.PDF

[40] H. Deitel, P. Deitel, B. DuWaldt, L. Trees *Web Services: A Technical Introduction*. Prentice Hall, 2002.

[41] DILIGENT website. http://www.diligentproject.org

[42] DSpace official website. http://www.dspace.org/

[43] J. Eder, H. Groiss, W. Liebhart. *The Workflow Management System Panta Rhei*. In: A. Dogac, L. Kalinichenko, T. Özsu, A. Sheth. Workflow Management Systems and Interoperability, Proceedings of the NATO Advanced Study Institute (ASI) on Workflow Management Systems, pp. 129-144. Istanbul, Turkey, Springer, 1998.

[44] EGEE: The EGEE Project website. http://www.eu-egee.org

[45] EPrints website. http://software.eprints.org

[46] A. Elmagarmid, Y. Leu, W. Litwin, M. Rusinkiewicz. *A Multidatabse Transaction Model for InterBase*. In: Proceedings of the 16th International Conference on Very Large Databases (VLDB'90), pages 507-518, Brisbane, Australia, August 1990.

[47] R. Fagin. *Combining fuzzy information from multiple systems*. SIGMOD Record 31, 2002.

[48] Fedora website. http://www.fedora.info/

[49] I. Foster. *What is the Grid? A Three Point Checklist*. GRID Today, July 20, 2002.

[50] I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. F. Ferguson, F. Leymann, M. Nally, T. Storey, W. Vambenepe, and S. Weerawarana. *Modeling Stateful Resources with Web Services*. White paper, 2004.

[51] I. Foster, C. Kesselman. *Globus: A Metacomputing Infrastructure Toolkit*. International Journal of Supercomputer Applications, 11(2):115-128, 1997.

[52] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, Second edition, November 2003. ISBN: 1558609334.

[53] I. Foster, C. Kesselman, J. Nick, S. Tuecke. *The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration*. Open Grid Service Infrastructure Working Group, Global Grid Forum, June 2002.

[54] I. Foster, C. Kesselman, S. Tuecke. *The Anatomy of the Grid: Enabling Scalable Virtual Organization*. The International Journal of High Performance Computing Applications, 15(3):200-222, 2001.

[55] I. Foster and A. Iamnitchi. *On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing*. In M. F. Kaashoek and I. Stoica, editors, Peer-toPeer Systems II, 2nd International Workshop, IPTPS 2003, Revised Papers, volume 2735, pages 118-128, 2003.

[56] N. Fuhr. *A Decision-Theoretic Approach to Database Selection in Networked IR*. ACM Transactions on Information Systems, 1999.

[57] D. Georgakopoulos, M. Hornick, A. Sheth. *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. In: Distributed and Parallel Databases, 3(2), pages 119-153, April 1995.

[58] M. Gillmann, G. Weikum, W. Wonner. *Workflow Management with Service Quality Guarantees*. In M. J. Franklin, B. Moon, A. Ailamaki, editors, Proc. of the 2002 ACM SIGMOD Int. Conf. on Management of Data, Madison, Wisconsin, June 3–6, 2002, pages 228–239, ACM Press, 2002.

[59] GGF: The Global Grid Forum website. http://www.ggf.org

[60] Globus: The Globus Alliance website. http://www.globus.org

[61] GlobusToolkit: The Globus Toolkit website. http://www.globus.org/toolkit/

[62] Greenstone website. http://www.greenstone.org

[63] P. Grefen, K. Aberer, H. Ludwig, Y. Hoffner. CrossFlow: *Cross-Organizational Workflow Management for Service Outsourcing in Dynamic Virtual Enterprises*. IEEE Data Engineering Bulletin, 24(1):52–57, 2001.

[64] GridSphere: The GridSphere website. http://www.gridsphere.org

[65] T. Grabs, K. Böhm, H.-J. Schek. *PowerDB-IR: Information Retrieval on Top of a Database Cluster*. In Proceedings of the 10th International Conference on Information and Knowledge Management, 2001.

[66] G. Hohpe, B. Woolf. E*nterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*, Addison-Wesley, 2003.

[67] R. Huebsch, J. M. Hellerstein, N. Lanham, B. Thau Loo, S. Shenker, I. Stoica. *Querying the Internet with PIER*, In Proceedings of the VLDB Conference, 2003.

[68] IBM MQseries. http://www.ibm.com/software/mqseries/

[69] IBM On Demand Computing. http://www-1.ibm.com/services/us/index.wss/rs/imc/a1002907?cntxtId=a1000074

[70] IBM WebSphere Application Server Enterprise Process Choreographer. http://www7b.software.ibm.com/wsdd/zones/was/wpc.html

[71] Jetspeed: The Jetspeed website. http://portals.apache.org/jetspeed-2/

[72] JMS: The Java Message Service website. http://java.sun.com/products/jms/

[73] JSR168: Portlet Specifications. http://jcp.org/en/jsr/detail?id=168

[74] M. Kamath, K. Ramamritham. *Correctness Issues in Workflow Management*. In: Distributed Systems Engineering Journal, 3(4), pages 213-221, December 1996.

[75] M. Keidl, S. Seltzsam, K. Stocker, A. Kemper. *ServiceGlobe: Distributing E-Services Across the Internet*. In Proc. of the 28th Int. Conf. on Very Large Data Bases, VLDB 2002, Hong Kong, China, pages 1047–1050. Morgan Kaufmann Publishers, 2002.

[76] D. Krafzig, K. Banke, D. Slama. *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall, 2004.

[77] C. Lagoze, S. Payette, E. Shin, C. Wilper, *Fedora: An Architecture for Complex Objects and their Relationships*. Forthcoming in Journal of Digital Libraries, Special Issue on Complex Objects, Springer 2005.

[78] C. A. Linch, *Institutional Repositories: Essential Infrastructure for Scholarship in the Digital Age*. ARL, no. 226 (February 2003): 1-7. http://www.arl.org/newsltr/226/ir.html

[79] F. Leymann. *Supporting Business Transactions via Partial Backward Recovery in Workflow Management Systems*. In: Proceedings of Datenbanksysteme in Büro, Technik und Wissenschaft (BTW'95), pages 51-70, Dresden, Germany, March 1995.

[80] A. Löser, F. Naumann, W. Siberski, W. Nejdl, U. Thaden. *Semantic Overlay Clusters within Super-Peer Networks*. In: Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing, 2003.

[81] J. Lu, J. Callan. *Content-based Retrieval in Hybrid Peer-to-Peer Networks*. In: Proceedings of the 12th International Conference on Information and Knowledge Management, 2003.

[82] S. Melnik, S. Raghavan, B. Yang, H. Garcia-Molina. *Building a Distributed Full-Text Index for the Web*. ACM Transactional Information Systems, 2001.

[83] J. Meng, S. Su, H. Lam, A. Helal. *Achieving Dynamic Inter-Organizational Workflow Management by Integrating Business Processes, Events and Rules*. In Proc. of the 35th Hawaii International Conference on System Sciences (HICSS-35 2002), January, Big Island, HI, USA, IEEE Computer Society, 2002.

[84] W. Meng, C. T. Yu, K.-L. Liu. *Building Efficient and Effective Metasearch Engines*. ACM Computing Surveys 34, 2002.

[85] Message Oriented Middleware Resources: http://www.huihoo.com/middleware/mom/mom_links.html

[86] B. Metha, M. Levy, G. Meredith, T. Andrews, B. Beckman, J. Klein, A. Mital. *BizTalk Server 2000 Business Process Orchestration*. IEEE Data Engineering Bulletin, 24(1):35–39, 2001

[87] M. Mlivoncic, C. Schuler, C. Türker: *Hyperdatabase Infrastructure for Management and Search of Multimedia Collections*. Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation, Proc. of the 6th Thematic Workshop of the EU Network of Excellence DELOS on Digital Library Architectures, S. Margherita di Pula (Cagliari), Italy, June 2004.

[88] Microsoft DCOM. http://www.microsoft.com/com/default.mspx

[89] Microsoft Message Queuing. http://www.microsoft.com/windows2000/technologies/communications/msmq/default.asp

[90] MOWS: The OASIS Web Services Distributed Management: Management of Web Services. http://www.oasis-open.org/committees/download.php/6255/cd-wsdm-mows-0.5-20040402.pdf

[91] MUWS: The OASIS Web Services Distributed Management: Management Using Web Services. http://www.oasis-open.org/committees/download.php/6234/cd-wsdm-muws-0.5.pdf

[92] E. Newcomer, G. Lomow. *Understanding SOA with Web Services*. Addison-Wesley, 2004.

[93] H. Nottelmann, N. Fuhr. *Evaluating Different Methods of Estimating Retrieval Quality for Resource Selection*. In: Proceedings of the 26th annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2003.

[94] OASIS: The OASIS Business Transactions. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=business-transaction

[95] OGSA-DAI: The OGSA-DAI website. http://www.ogsadai.org

[96] OMG Corba. http://www.omg.org/gettingstarted/corbafaq.htm

[97] OpenDLib: http://www.opendlib.com

[98] Oracle Database 10g: The Database for the Grid. http://otn.oracle.com/products/database/oracle10g/

[99] A. Oram. *Peer-to-peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[100] OWL: The W3C OWL Web Ontology Language. http://www.w3.org/TR/2004/REC-owl-features-20040210/

[101] The Peers Group. *Peer-to-Peer Research at Stanford*. ACM SIGMOD Record, September 2003.

[102] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker. *A Scalable Content-Addressable Network*, In: Proceedings of ACM SIGCOMM, 2001.

[103] RDF: The W3C Resource Description Framework (RDF). http://www.w3.org/RDF/

[104] P. Reynolds, A. Vahdat: *Efficient Peer-to-Peer Keyword Searching*. ACM/IFIP/USENIX International Middleware Conference, 2003.

[105] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. *Pond: The Oceanstore Prototype*. In: Proceedings of USENIX File and Storage Technologies FAST, 2003.

[106] A. Rowstron, P. Druschel. *Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems*. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001.

[107] SAML: The OASIS Security Services. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[108] H.-J. Schek, K. Böhm, T. Grabs, U. Röhm, H. Schuldt, R. Weber. *Hyperdatabases*. In Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE'00), pages 14–23, Hong Kong, China, June 2000.

[109] H.-J. Schek, H. Schuldt, C. Schuler, R. Weber. *Infrastructure for Information Spaces*. In Proceedings of Advances in Databases and Information Systems, 6th East European Conference, ADBIS 2002, volume 2435 of Lecture Notes in Computer Science, pages 23–36, Bratislava, Slovakia, September 2002. Springer.

[110] H.-J. Schek, H. Schuldt, R. Weber. *Hyperdatabases – Infrastructure for the Information Space*. In Proceedings of the 6th IFIP 2.6 Working Conference on Visual Database Systems (VDB'02), Brisbane, Australia, May 2002.

[111] D. Schoder, K. Fischbach, R. Teichmann. *Peer-to-Peer*. Springer, 2002.

[112] H. Schuldt. *Process Locking: A Protocol based on Ordered Shared Locks for the Execution of Transactional Processes*. In Proceedings of the 20th ACM Symposium on Principles of Database Systems (PODS'01), pages 289–300, Santa Barbara, California, USA, May 2001. ACM Press.

[113] H. Schuldt, G. Alonso, C. Beeri, H.-J. Schek: *Atomicity and Isolation for Transactional Processes*. In: ACM Transactions on Database Systems (TODS) 27(1), March 2002.

[114] C. Schuler, R. Weber, H. Schuldt, and H. -J. Schek: *Peer–to–Peer Process Execution with OSIRIS*. In: Proceedings of the 1st International Conference on Service-Oriented Computing, Trento, Italy, December 2003.

[115] C. Schuler, R. Weber, H. Schuldt, H.-J. Schek: *Scalable Peer-to-Peer Process Management - The OSIRIS Approach*. In: Proceedings of the 2nd International Conference on Web Services (ICWS'2004), pages 26-34, San Diego, CA, USA, July 2004. IEEE Computer Society.

[116] SOA and Web Services: http://www-106.ibm.com/developerworks/webservices/newto/

[117] SOAP: W3C SOAP 1.2 Primer: http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[118] SOAP: The W3C SOAP. http://www.w3.org/TR/soap/

[119] SOAP Message Security: The OASIS Web Services Security. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[120] SOAP-MTOM: The W3C SOAP Message Transmission Optimization Mechanism (SOAP-MTOM). http://www.w3.org/TR/2004/WD-soap12-mtom-20040608/

[121] L. Smarr and C. E. Catlett. *Metacomputing*. In Communications of the ACM , 35 (6): 44-52, 1992.

[122] SRB: The SDSC Storage Resource Broker (SRB) Homepage. http://www.npaci.edu/DICE/SRB/

[123] T. Suel, C. Mathur, J. Wu, J. Zhang, A. Delis, M. Kharrazi, X Long, and K. Shanmugasundaram. *ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval*, In: Proceedings of the 6th International Workshop on the Web and Databases (WebDB), 2003.

[124] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan. *Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications*. IEEE/ACM Transactions on Networking, 2003.

[125] Z. Stojanovic, A. Dahanayake. *Service-oriented Software System Engineering Challenges and Practices*. Idea Group Publishing, 2004.

[126] C. Tang, Z. Xu, S. Dwarkadas. *Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks*. In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, 2003.

[127] R. Tansley, M. Bass, D. Stuve, M. Branschofsky, D. Chudnov, G. McClellan, M. Smith, *The DSpace Institutional Digital Repository System: Current Functionality*. In Proceedings of the third ACM/IEEE-CS Joint Conference on Digital libraries (JCDL '03), pages 87-97, Houston, Texas. IEEE Computer Society, 2003.

[128] D. Tombros, K. Dittrich. *SWORDIES - Swiss Workflow Management in Distributed Environments*. In Informatik/ Informatique, Journal of the Swiss Computer Science Society. No. 3 June 1999.

[129] UDDI: The UDDI website. http://www.uddi.org/

[130] J. Uhlmann: *Satisfying General Proximity / Similarity Queries with Metric Trees*. IPL: Information Processing Letters, 40 (1991), 175–179.

[131] H. Wächter, A. Reuter. *The ConTract Model*. In: A. Elmagarmid (ed.), Database Transaction Models for Advanced Applications, chapter 7, Morgan Kaufmann Publishers, 1992.

[132] Y. Wang, L. Galanis, D. de Witt. *GALANX: An Efficient Peer-to-Peer Search Engine System*. Available at http://www.cs.wisc.edu/~yuanwang

[133] D. Wodtke, J. Weißenfels, G. Weikum, A. Kotz Dittrich: *The Mentor Project: Steps Toward Enterprise-Wide Workflow Management*. In: Proceedings of the 12th International Conference on Data Engineering, pp 556-565, New Orleans, 1996.

[134] WS-Addressing: The Web Services Addressing. http://www-106.ibm.com/developerworks/library/specification/ws-add/

[135] WS-Agreement: The Web Services Agreement. http://www.gridforum.org/Meetings/GGF11/Documents/draft-ggf-graap-agreement.pdf

[136] WS-AtomicTransaction: The Web Services Atomic Transaction. http://www-106.ibm.com/developerworks/library/ws-atomtran/

[137] WS-BaseNotification, WS-BrokeredNotification, WS-Topics: The Web Services Notification Family Standards. http://www-106.ibm.com/developerworks/library/specification/ws-notification/

[138] WS-BusinessActivity: The Web Services Business Activity Framework. http://www-106.ibm.com/developerworks/webservices/library/ws-busact/

[139] WS-CAF: The Web Services Composite Application Framework. http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CAF-Primer.pdf

[140] WS-CF: The Web Services Coordination Framework. http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CF.pdf

[141] WS-Choreography: The W3C Web Services Choreography (WS-Choreography). http://www.w3.org/2002/ws/chor/

[142] WS-Coordination: The Web Services Coordination. http://www-106.ibm.com/developerworks/library/ws-coor/

[143] WS-CTX: The Web Services Context. http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CTX.pdf

[144] WS-Discovery: The Web Services Dynamic Discovery Specification. http://msdn.microsoft.com/ws/2004/02/discovery

[145] WS-Eventing: The Web Services Eventing. http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-Eventing.asp

[146] WS-Federation: The Web Services Federation Language. http://www-106.ibm.com/developerworks/library/ws-fedworld/

[147] WS-GAF: The Web Services Grid Application Framework. http://www.neresc.ac.uk/ws-gaf/

[148] WS-I basic profile: http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

[149] WS-IL: The Web Services Inspection Language. http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html

[150] WS Interoperability: The Web Services Interoperability Organization Website. http://www.ws-i.org/

[151] WS-Manageability: ftp://www6.software.ibm.com/software/developer/library/ws-manage.pdf

[152] WS-MessageDelivery: The W3C Web Services Message Delivery. http://www.w3.org/Submission/ws-messagedelivery/

[153] WS-MetadataExchange: The Web Services Metadata Exchange. http://www-106.ibm.com/developerworks/library/specification/ws-mex/

[154] WS-Policy: The Web Services Policy Framework. http://www-106.ibm.com/developerworks/library/specification/ws-polfram/

[155] WS-Reliability: The OASIS Web Services Reliable Messaging. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm

[156] WS-RM: The Web Services Reliable Messaging. http://www-106.ibm.com/developerworks/webservices/library/ws-rm/

[157] WS-Routing: The Web Services Routing Protocol. http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-routing.asp

[158] WS-Security: http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

[159] WS-SecureConversation: The Web Services Secure Conversation Language. http://www-106.ibm.com/developerworks/library/specification/ws-secon/

[160] WS-SecurityPolicy: The Web Services Security Policy. http://www-106.ibm.com/developerworks/library/ws-secpol/

[161] WS-TXM: The Web Services Transaction Management. http://www.arjuna.com/library/specs/ws_caf_1-0/WS-TXM.pdf

[162] WS-Trust: The Web Services Trust Language. http://www-106.ibm.com/developerworks/library/specification/ws-trust/

[163] WSCL: The W3C Web Services Conversation Language (WSCL). http://www.w3.org/TR/wscl10/

[164] WSDL: The W3C Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl

[165] WSDM: The OASIS Web Services Distributed Management. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm

[166] WSFL: Web Services Flow Language. http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[167] WSRP: The OASIS Web Services for Remote Portlets. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp

[168] Z. Wu, W. Meng, C. T. Yu, Z. Li. *Towards a Highly-Scalable and Effective Metasearch Engine*. Word Wide Web Conference, 2001.

[169] M. Wurz, G. Brettlecker, H. Schuldt. *Data Stream Management and Digital Library Processes on Top of a Hyperdatabase and Grid Infrastructure*. In: Pre-Proceedings of the 6th Thematic Workshop of the EU Network of Excellence DELOS: Digital Library Architectures - Peer-to-Peer, Grid, and Service-Orientation (DLA 2004), pages 37-48, Cagliari, Italy, June 2004, Edizioni Progetto Padova.

[170] XACML: The OASIS eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[171] XLANG – Web Services for Business Process Design. http://www.gotdotnet.com/team/xml wsspecs/xlang-c/default.htm

[172] XML: eXtended Markup Language. http://www.w3.org/TR/REC-xml

[173] XML Schema: The W3C XML Schema. http://www.w3.org/XML/Schema

[174] B. Yang, H. Garcia-Molina. *Improving Search in Peer-to-Peer Networks*. Proceedings of the 22nd International Conference on Distributed Computing Systems, 2002.

[175] A. Zhang, M. Nodine, B. Bhargava. *Global Scheduling for Flexible Transactions in Heterogeneous Distributed Database Systems*. In: IEEE Transactions on Knowledge and Data Engineering (TKDE), 13(3), pages 439-450, 2001.