In: Proceedings of the 6^{th} East-European Conference on Advances in Databases and Information Systems (ADBIS'2002), Bratislava, Slovakia, September 2002. Lecture Notes in Computer Science, Springer-Verlag.

Infrastructure for Information Spaces

Hans-J. Schek, Heiko Schuldt, Christoph Schuler, and Roger Weber

Database Research Group, Institute of Information Systems
Swiss Federal Institute of Technology (ETH)
ETH Zentrum, CH–8092 Zurich, Switzerland
Email: {schek,schuldt,schuler,weber}@inf.ethz.ch

Abstract. In the past, we talked about single information systems. In the future, we expect an ever increasing number of information systems and data sources, reaching from traditional databases and large document collections, information sources contained in web pages, down to information systems in mobile devices as they will occur in a pervasive computing environment. Therefore not only the immense amount of information demands new thoughts but also the number of different information sources. Essentially, their coordination poses a great challenge for the development of future tools that will be suitable to access, process, and maintain information. We talk about the continuous, "infinite" information, shortly called the "information space". Information in this space is distributed, heterogeneous and undergoes continuous changes. So, the infrastructure for information spaces must provide convenient tools for accessing information, for developing applications for analyzing, mining, classifying, and processing information, and for transactional processes that ensure consistent propagation of information changes and simultaneous invocations of several (web) services within a transactional workflow. As far as possible, the infrastructure should avoid global components. Rather, a peer-to-peer decentralized coordination middleware must be provided that has some self-configuration and adaptation features. In this paper we will elaborate some of the aspects related to process-based coordination within the information space and report on research from our hyperdatabase research framework and from experiences in ETHWorld, an ETH wide project that will establish the ETH information space. Nevertheless, this paper is rather visionary and is intended to stimulate new research in this wide area.

1 Introduction

Until recently, information processing was dominated by single, well-delimited albeit possibly distributed information systems. In the future, we see a strong reinforcement of the current trend from these single information systems towards an ever increasing number of information systems and data sources, reaching from traditional databases and large document collections, information sources contained in web pages, down to information systems in mobile devices and embedded information in mobile "smart" objects as they will occur in a pervasive computing environment. As a consequence of this development, the amount of stored information is exploding. However, not only the immense amount of information demands new thoughts but also the number of different information sources and their coordination poses a great challenge for the development of future tools that will be suitable to access, process, and maintain information. The reason being is that information will not be static but rather continuously updated, extended, processed and/or refined. We talk about the continuous, "infinite" information, shortly called the "information space". Information in this space is distributed, heterogeneous and undergoes continuous changes.

For all these reasons, the infrastructure for information spaces must provide convenient tools for accessing information via sophisticated search facilities and for combining or integrating search results from different sources (1), for developing applications for analyzing, mining, classifying and processing information (2), and for transactional processes that ensure consistent propagation of information changes and simultaneous invocations of several (web) services within a transactional workflow (3). The latter requirement is needed in order to automatically keep track of dependencies within the information space. Dependencies might exist when data is replicated at several sources or when derived data, e.g., in the form of sophisticated index structures, has to be maintained. As far as possible the infrastructure should avoid global components and thus also potential bottlenecks. Rather, a peer-to-peer decentralized coordination middleware must be provided that has some self-configuration and adaptation features to various applications and their load characteristics (4). This urgent requirement stems from the fact that the information space is not at all static but rather might change its character over time. Hence, an infrastructure that supports aspects like a high degree of flexibility and self-configuration can dynamically adapt to changes in the environment without explicit intervention and/or tuning.

In this paper we will elaborate some of the aspects in area (3) and (4) and report on research from our hyperdatabase research framework [16] and from experiences in ETHWorld [4,17], an ETH wide project that will establish the ETH virtual campus, a large-scale information space. In particular, we will present the prototype system OSIRIS (**O**pen **S**ervice **I**nfrastructure for **R**eliable and **I**ntegrated process **S**upport) and we will show how the advanced concepts identified for the management and organization of large-scale information spaces are realized in a peer-to-peer environment and by applying sophisticated publish/subscribe techniques. Nevertheless, this paper is rather visionary and is intended to stimulate new research in this wide area.

The remainder of this paper is organized as follows: In Section 2, we introduce the notion of information space and highlight the problems associated with the management of data and services out of which the information space is built. Moreover, we discuss the basic foundations and prerequisites of the architecture we propose for managing large-scale information spaces. In Section 3, we present OSIRIS, our hyperdatabase prototype system in detail, and we introduce the ETHWorld project in which OSIRIS is currently being used. Section 4 discusses related work and Section 5 concludes.

2 Organizing and Maintaining Information Spaces: Problems & Challenges

In this section, we introduce the notion of *information space* and discuss the challenges for an infrastructure for information space maintenance and management and show how such an infrastructure can be implemented.

2.1 Information Spaces

Information spaces are collections of semantically related data and services. We differentiate between three abstractions of information spaces: Firstly, the *global information space* which comprises the universe of all documents and services that can be accessed, mainly via the Internet but also from other sources like databases, etc. Secondly, *community information spaces* (or, synonymously, *intranet information spaces*) which are defined by the data and services of a closed, coherent part of the global information space. Thirdly, personal information spaces which are individual views on a community information space tailored to the needs of a user in his/her current context.

Usually, the set of documents, data, and services of an information space is distributed and heterogeneous. In addition, information spaces are dynamic. This means that updates and deletions of data and documents as well as the new generation of documents have to be considered. To this end, the peers hosting information provide appropriate *services* by which these operations are encapsulated and made available to the users. Since documents within the information space are usually not independent, the local invocation of such services has several side-effects and therefore also affects other documents without the user being aware of these dependencies. Such dependencies occur, e.g. when data is replicated and indexed via search engines within the information space or when certain documents are derived from others. The task of *information space maintenance* is to keep track of these dependencies and to automatically re-establish consistency within the information space whenever necessary. A mayor difference between the global information space and community information spaces is that the latter are manageable in the sense that all existing dependencies are known such that they can be tracked. For this reason, we focus on the management of community information spaces; however, the basic mechanisms we propose for information space maintenance can also be applied to the global information space.

An example of a community information space is ETHWorld, the virtual campus of ETH that we will describe in more detail below. As a further and more general example for a community information space, consider the information space of a large-scale international company. The documents, out of which the company's information space is built, are product descriptions and technical documentation in some proprietary formats stored in file systems, product catalogs and inventory control data residing in databases, information within the company's intranet and on some information available over the Internet, as well as information stored in e-mail archives, etc. In addition, information is replicated at several places, and local caches exist, e.g., to support mobile devices. However, information may be added, changed, deleted by different individuals at different subsidiaries and no global control on the consistency of the overall community information space is possible. For instance, the update of some product description in a product database must be propagated to the replicas of all subsidiaries as well as to the information on the company web server.

Obviously, such dependencies within the information space have to be tracked and consistency between original and replicated and/or derived data has to be guaranteed. Therefore, the infrastructure of the information space has to support the execution of *coordination processes.* These are applications which are defined on top of the existing services, i.e., which use these services as building blocks. The goal of these coordination processes is to free the users of the information space from dealing with dependencies. Rather, appropriate processes have to be executed automatically after services are invoked so as to maintain the overall information space. The prerequisite for this is that i.) the necessary processes are properly defined and ii.) they are linked to the appropriate event necessitating coordination efforts. In particular, the latter requirement demands that service invocations can be monitored. Such services which seamlessly support the task of information space maintenance will be called *cooperative* services. Conversely, *non-cooperative* services are those that do not allow for automating coordination processes. Consequently, a basic requirement for sophisticated information space maintenance and management is that all services having side-effects within the information space are cooperative ones. Processes combining and virtually integrating single services can themselves be considered as high-level services within the information space.

2.2 Infrastructure for Information Spaces: Hyperdatabases

How must a proper infrastructure for information spaces look like? Clearly, we may think of a monolithic database that stores all information and that provides all the necessary services for all clients. However, this would not match the requirements for information spaces, mainly for two reasons. First, information is distributed by definition. It would not be feasible to force all information providers to store their data at a single place. Second, a monolithic system would not scale to the increasing demands of an information space. Recent advances in networking, storage, and computing power demonstrate that we are able to communicate and store vast amounts of data, but a single machine's power cannot catch up with the increased demands of larger data sets and larger communication rates. A useful metric for the rate of technological changes is the average period during which speed or capacity doubles or halves in price. For storage and networking, this period is around 12 and 9 months, respectively. The one for computing power lies at around 18 months. Obviously, computing power is falling behind. Consequently, a monolithic database solution for an information space is not appropriate, but we still want its benefits, yet at a higher level of abstraction. As a developer of such an information space, we want something

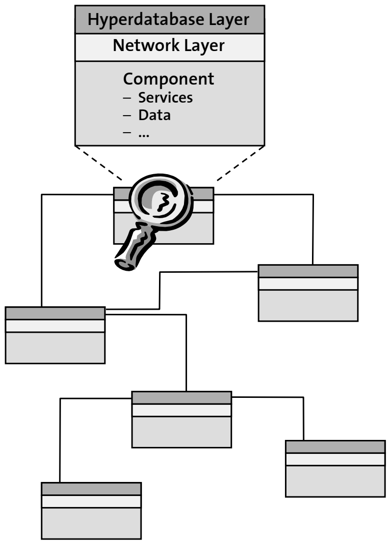

Fig. 1. The Hyperdatabase (HDB) is a Layer on top of Networking

similar to data independence but for services. We want transactional guarantees but for processes (workflows) over distributed components using existing services. We want recovery and fault tolerance but at the process level. In a nutshell, we would like to have a database over components and services, and a database over databases: a Hyperdatabase (HDB) [16,17].

Essentially, organizing an information space means providing a coherent, integrated, and consistent view on a vast amount of information that is produced, stored, and even updated by distributed components within a networked environment. Due to the physical distribution of information sources and services, a core functionality of an HDB for the information space is coordination. This means that the HDB has to keep track of changes, to continuously propagate these changes, and to derive and update particular views of the overall (community) information space. As a consequence, HDB functionality is distributed over the participating components and located as additional layer on top of the network layer (see Figure 1). Following the analogy of transactional processes being the HDB abstraction of database transactions, a HDB as coordinator has to run such transactional processes and to guarantee their correct termination. In addition, the HDB offers a simple way to add and administer specialized components (cf. Figure 2). The services provided by these components can be combined to form application-aware processes with specific execution guarantees (recovery, exception handling, alternative executions, consistency, concurrency control, etc.). Commonly, such processes are triggered by events like insertion, update, or connect/disconnect. Computationally expensive tasks may easily be scaled out to a cluster of workstations (computational services).

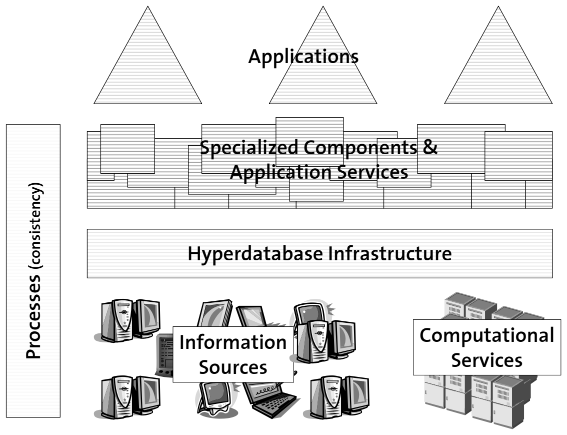

Fig. 2. Architecture of a Community Information Space

The traditional approach to implementing systems providing functionality similar to the functionality required from an HDB consists of *decentralized* storage and service components and a *centralized* control component (coordinator, workflow or process engine). When enriching such a process engine with transactional semantics (e.g., [3,12,18]), the result will be a system that provides all the required execution guarantees and thus will seamlessly support the task of information space maintenance. However, the central control component disallows for scaling the system to the demands and sizes of an information space as all steps of all processes are routed through this (bottleneck) component. As insinuated above, computational power does not advance as quickly as storage and networking technologies. Consequently, the maximum number of clients, processes, and information providers will be limited by the computational capabilities of the central control component. Another problem arises from the lack of resource management. Computational expensive processes demand for an infrastructure as provided by grids. A grid infrastructure pools complex tasks which are assigned to resources whenever they signal "idle state" or "finished last task". But like in workflow systems, a grid relies on a central component which assigns tasks to is subsidiary components. On the other hand, this central component holds enough global metadata on the tasks and resources to ensure an optimal assignment of tasks to resources and achieve a maximum throughput of tasks per time unit. But still, communication mainly incurs between the central component and the decentralized resources.

Peer-to-peer systems have recently attracted large interest from researches, companies and Internet users. The first successful application of peer-to-peer systems, file sharing, demonstrated the potential of an entirely decentralized architecture. Protocols like the ones of Gnutella [6] or FastTrack [5] require no central control or repository component but still provide clients with sufficiently accurate information about the state of their specific information space. Moreover, since communication takes place without any central component involved, peer-to-peer systems easily scale up to any size. Clearly, peer-to-peer communication is a demand for an HDB that must scale to the sizes of large information spaces. However, unlike in file sharing applications, an HDB requires a more or less consistent, complete, and accurate view of the overall (community) information space. While clients of a file sharing system will tolerate missing entries in their results, this clearly is not the case for applications and processes in the information space for which overall consistency is of primary concern.

Peer-to-peer communication enables unlimited scalability, but to ensure correctness and completeness, an HDB must rely on accurate global metadata. But peers should not have to access global metadata on a centralized component whenever they want to perform an action. Rather, the peers should be provided with replicas of the portion of global metadata they need to fulfill their tasks. Clearly, this demands for a sophisticated replication scheme within the HDB. Although replication appears to be expensive at first glance, the characteristics of an information space and recent advances in networking technologies enable such a solution. For instance, if a peer must invoke a service s, it must know which providers of s currently exist. Hence, the peer is given a replica of the list of providers of s from the global metadata. In particular, we assume that the number of processes to be executed and therefore the number of service invocations considerably exceeds the number of changes of global metadata such that metadata replication pays off. In addition, the freshness of replicated data may in certain cases be relaxed. For instance, a peer can live with a slightly outdated list as long as it can find at least one service provider.

Summarizing the previous discussion, an implementation of an HDB should encompass the following five essential concepts (in brackets, we have listed the areas from which we have borrowed the respective concepts):

– Sophisticated process management enriched by execution guarantees (transactional workflow or process management).
– Resource management to enable optimal execution of computational expensive tasks (grid computing).
– Peer-to-peer communication at the process level. No central component shall be involved in the navigation of processes or routing of process steps (peer-to-peer systems).
– Accurate global metadata to provide a consistent and complete view (transactional process management and grid computing).
– Decentralized replication of metadata with freshness guarantees.

3 Managing the Information Space of ETHWorld

In the following, we describe the OSIRIS system developed at the database group of ETH. Moreover, we highlight the problems associated with maintaining the information space of a virtual campus, an environment where the OSIRIS system is currently being applied.

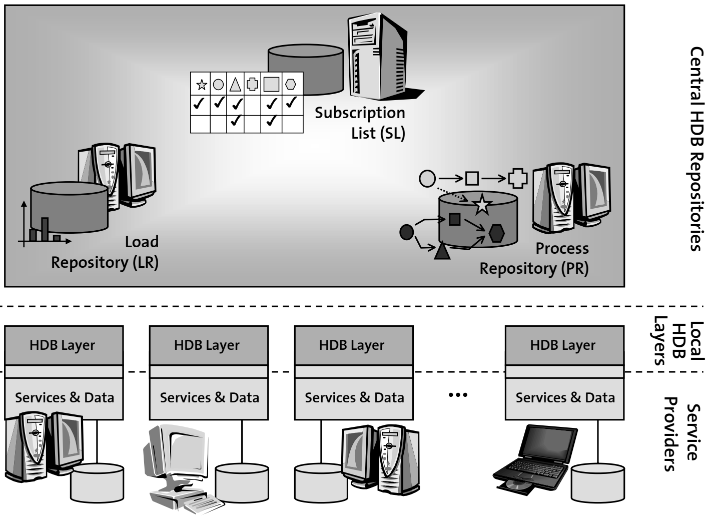

Fig. 3. The OSIRIS Architecture

3.1 The OSIRIS System

When implementing an HDB infrastructure for the management and maintenance of an information space, support for the various requirements we have identified in the previous section is vital for its success. To this end, the OSIRIS system (**O**pen **S**ervice **I**nfrastructure for **R**eliable and **I**ntegrated process **S**upport) [19] provides basic support for executing processes with dedicated execution guarantees but also accounts for the inherent distribution that can be found within the information space and the need for a highly scalable system. The latter requirement is stemming from the fact that services may be dynamically added to or revoked from the system such that the infrastructure should provide certain self-adaptation features [21].

In a nutshell, OSIRIS consists of two kinds of components: i.) a small HDB layer added to each component providing a service within the community information space (these components will be shortly denoted as 'service providers') and ii.) a canonical set of basic repositories managing global metadata describing the system configuration and the applications (processes) to be executed within the information space (c.f. Figure 3).

Following the core characteristics of an HDB, the OSIRIS system provides support for process execution in a peer-to-peer style. To this end, metadata on the processes and on the system configuration is replicated. This allows each HDB layer of a service provider to locally drive the execution of a process, i.e., to invoke the next process step after a local service invocation has terminated. As a consequence, no centralized control is involved during process execution.

The first step in providing support for the maintenance of an information space is that the processes to be executed in the information space have to be properly specified. This is usually done by the administrator of the community information space and is supported in OSIRIS by the graphical workflow process modeling tool IvyFrame [9]. After specification, the process information is stored in a global process repository (PR). Since processes are the applications that OSIRIS executes within the information space, the process repository can be considered as a kind of software archive. Similarly, another global component (subscription list, SL) manages the different service providers of the system, i.e., the system configuration. To this end, each service provider has to register its service(s). Both PR and SL are then used as sources for metadata distribution. After a provider p registers its service s_i with SL, it receives in return the parts of all process models in which s_i appears. In particular, these parts contain pairs of subsequent services (s_i, s_k), i.e., specify the services that have to be immediately invoked after s_i has terminated. In addition, service provider p will also receive the list of all providers offering service s_k which has to be invoked after s_i. Process execution then takes place only at the level of the HDB layers of the respective service providers: when a process is started, the first service is invoked. After its termination, the HDB layer of the provider directly invokes the subsequent service, thereby transferring control to its HDB layer. These mechanisms are stepwise applied until the successful termination of a process.

Essentially, OSIRIS uses publish/subscribe techniques to execute processes. Conceptually, each HDB layer generates (*publishes*) an event after the termination of a service s_i. This event is converted to an invocation of the subsequent service s_k of the current process and transferred to the HDB layer of a service provider which has previously registered a service s_k, i.e., has made a *subscription* of the appropriate event indicating that s_k has to be executed. But instead of matching publication and subscriptions by a centralized publish/subscribe broker, this is done locally by the HDB layer. To this end, the replicated metadata allows for a distributed and decentralized implementation of publish/subscribe functionality. Each local HDB layer is equipped with a publish/subscribe broker such that events can be handled locally, based on the replicas of PR and SL. However, a question that immediately arises is how this replicas are kept consistent. For this problem, again publish/subscribe techniques are applied. When the initial subscription of a service provider for service s_i is done, it is not only registered as provider in SL. Implicitly and transparent to the provider, a second subscription is generated by the system. This subscription declares interest on all metadata of the system associated with this service. Initially, it corresponds to the copy of the metadata on process models and subscribers. However, since the second, implicit subscription is held until the provider revokes its service, it guarantees that each change in the metadata of PR and SL relevant to this particular provider is updated. This is essentially necessary in the following cases: i.) new processes are defined in which s_i is to be executed, ii.) processes are up-

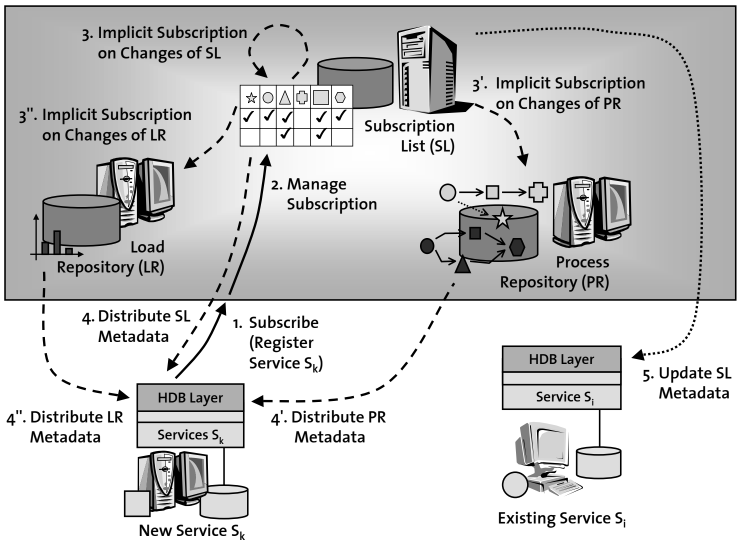
3. Implicit Subscription on Changes of SL
3". Implicit Subscription on Changes of LR
Subscription List (SL)
3'. Implicit Subscription on Changes of PR
2. Manage Subscription
Load Repository (LR)
Process Repository (PR)
4. Distribute SL Metadata
1. Subscribe (Register Service S_k)
HDB Layer
Services S_k
4". Distribute LR Metadata
4'. Distribute PR Metadata
New Service S_k
HDB Layer
Service S_i
5. Update SL Metadata
Existing Service S_i

Fig. 4. DIPS Metadata Replication Management

dated such that the subsequent service after s_i changes or disappears, iii.) new providers register which offer a service s_k that is to be invoked after s_i in some process, or iv.) service providers revoke their service. In all cases, changes are submitted to the central repositories and generate an event there. This event is then, due to the second, implicit subscription, transferred to the respective HDB layers of service providers which then update their local replica. The repeated application of publish/subscribe techniques is termed DIPS (Doubled Implicit Publish/Subscribe). The DIPS-based metadata replication management is depicted in Figure 4. The explicit subscription of a service is illustrated by solid arcs, the implicit second subscription by dashed arcs, and the metadata distribution to existing services by dotted arcs.

When the local HDB layer has to process the event indicating that a local service s_i has terminated, it has to choose one concrete service provider among the local replica containing the list of providers having registered for the subsequent service s_k. In order to allow for sophisticated load balancing within the community information space, the local replicas not only contain metadata on processes and service providers but also on the (approximated) load of the latter. Hence, the local publish/subscribe engine within the HDB layer is able to choose the provider with the lowest current load. The distribution of metadata on the load of a provider is again done by publish/subscribe techniques. The initial, explicit subscription does not only generate an implicit subscription on changes of PR and SL but also on their loads. These loads are, similarly to PR and SL globally maintained by the load repository, LR (see Figure 4). However, in contrast to the replicas of metadata of PR and SL that have to be kept consistent, load information may only be an approximation of the actual load of the provider. To this end, not each minor change of the load of a provider is published via an appropriate event but only significant changes exceeding a pre-defined threshold. This information is then made available to all local HDB layers for which the appropriate provider is of interest.

In addition to the basic DIPS support for distributing process information, system configuration (registered providers), and load information, OSIRIS provides dedicated transactional execution guarantees by following the ideas of transactional process management [18]. Essentially, this includes a notion of atomicity that is more general than the traditional "all-or-nothing" semantics and guarantees that exactly one out of several alternative executions that are specified within a process is correctly effected.

Finally, a last task to be solved is that processes are automatically started whenever coordination efforts are required in the information space. Each HDB layer of a component offering services which have side-effects such that their invocation necessitates coordination activities has to monitor the invocation of these services. Whenever an invocation is detected, the appropriate event is published such that the first step of the respective process needed to enforce consistency is started (in the OSIRIS peer-to-peer way as presented above). Yet, a basic requirement is that each of these services necessitating coordination processes are cooperative ones.

3.2 The ETHWorld Virtual Campus

A concrete application of the OSIRIS HDB infrastructure is in the context of coordinating multimedia information components in the ETHWorld project [4,17]. ETHWorld was established by ETH Zürich to create a virtual campus where students, assistants, professors, researches inside and outside ETH can meet, discuss, exchange information, or work together.

A virtual campus, as an example of a large-scale community information space, consists of various documents distributed over a large number of peers. A particular sub-project of the ETHWorld initiative addresses multimedia similarity search with focus on image retrieval and relevance feedback. The goal is to allow for the interactive exploration of the community information space of the virtual campus. To this end, we have built the ISIS system (**I**nteractive **SI**milarity **S**earch) which provides efficient and effective search methods [8]. Its goals are to i.) identify and build effective content descriptors for various document types, to ii.) develop efficient search methods that allow for complex similarity retrieval [20], and to iii.) build easy-to-use and powerful relevance feedback methods to support query formulation.

Since information may be added at any place in the information space but should be accessible by the ISIS search engine as soon as possible (without the delay known from search engines in the Internet), ISIS requires support from OSIRIS with respect to coordination efforts. As a concrete example, the OSIRIS infrastructure has to monitor local insertions of new documents (e.g., in some

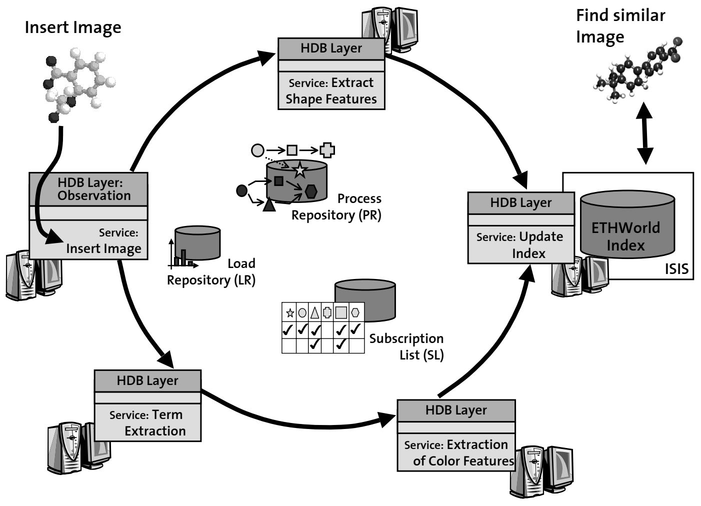

Fig. 5. OSIRIS: Distributed Process Support Infrastructure in ETHWorld

image database maintained at the ETH library) and to automatically start the execution of a coordination process by publishing its start event. Within this process, a well-defined sequence of steps is executed; each step corresponds to a service invocation (e.g., for extracting color and shape features and for term extraction from text). These features are required to finally maintain the index allowing for sophisticated search techniques within the information space. Hence, OSIRIS is the infrastructure that, under the cover, provides users with quality of service guarantees for the information she/he wants to access (both in terms of the users of ISIS being served with up-to-date data and in terms of the users of the ETHWorld community information space which do not have to explicitly care about coordination efforts).

In Figure 5, the OSIRIS infrastructure for the `InsertDocument` process is depicted: process execution is driven by the HDB layers of the individual service providers (outer circle) while meta information required for process execution is collected by the global repositories (within the circle). Replication management from the global repositories to the service providers of the outer circle is done using DIPS techniques.

4 Related Work

The dynamic character of the information space necessitates the management of metadata on the services of different providers. Our approach which uses a global subscription list component allows the infrastructure to dynamically choose a suitable service instance at process execution time. This allows the system to execute processes in a dynamic way. Approaches dealing with similar paradigms are eFlow [2] and CrossFlow [7]. Existing service description directories like the UDDI Repository [1] could be used to discover external, non-cooperative service providers. To find services matching a complex service definition, additional effort has to be taken. The ISEE [13] system shows how e–service constraints can be used to match suitable service instances. In medical information systems, even instances of long–running patient treatment processes have to be continuously updated to the most recent process template definition so as to provide up-to-date treatment knowledge. Therefore, systems from this field have to deal with an additional type of dynamic behavior. Running processes have to be migrated to newer definitions. Systems like HematoWork [14] and $\text{ADEPT}_{\text{flex}}$ [15] address these aspects. Such mechanisms are however orthogonal to OSIRIS and could be seamlessly integrated to additionally enable this kind of applications.

OSIRIS provides the guarantee of automated update propagation between autonomous data sources (and even application systems) in the information space. Processes are used to convert data formats and semantics from the source to the target system. The set of services used to define such update processes provide the functionality needed. Similar conversion flows are used to define "XML Pipelines" [22], or "Infopipes" [11].

The OSIRIS System uses publish/subscribe techniques both for replication management of system metadata and for process navigation. The approach presented in [10] shows how replication management can be implemented in an efficient way. Using this rule matching algorithm, a large amount of clients can be served which accurate information.

5 Conclusions

In this paper, we have discussed the various requirements for an infrastructure aiming at maintaining and managing community information spaces. The most important task is to guarantee consistency and correctness within the information space by executing appropriate coordination processes whenever necessary. The core of the infrastructure is formed by a hyperdatabase (HDB) which allows for the seamless combination of existing services into processes.

Furthermore, we have presented the OSIRIS system, an HDB implementation, which accounts for all these challenging requirements. In OSIRIS, process execution takes place in a peer-to-peer way. OSIRIS distributes replicas of global metadata at each peer in a way that is transparent to the users of the information space. This is realized by applying publish/subscribe techniques for both service providers of the system and for the global metadata repositories.

Finally, we have introduced the ETHWorld project aiming at providing the ETH virtual campus. In here, OSIRIS is used as the underlying infrastructure for the community information space of ETHWorld.

References

1. Ariba, IBM, and Microsoft. UDDI Technical White Paper. `http://www.uddi.org`.

2. F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan. Adaptive and Dynamic Service Composition in eFlow. In *Proc. Conf. on Advanced Information Systems Engineering*, Stockholm, 2000.
3. Q. Chen and 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.
4. ETHWorld – The Virtual Campus of ETH Zürich. `http://www.ethworld.ethz.ch`.
5. FastTrack – P2P Technology. `http://www.fasttrack.nu`.
6. Gnutella RFC. `http://rfc-gnutella.sourceforge.net`.
7. P. Grefen, K. Aberer, H. Ludwig, and Y. Hoffner. CrossFlow: Cross–Organizational Workflow Management for Service Outsourcing in Dynamic Virtual Enterprises. *IEEE Data Engineering Bulletin*, 24:52–57, 2001.
8. ISIS – Interactive SImilarity Search. `http://www.isis.ethz.ch`.
9. IvyTeam. IvyFrame: Process Modeling & Simulation. `http://www.ivyteam.com`.
10. M. Keidl, A. Kreutz, A. Kemper, and D. Kossmann. A Publish & Subscribe Architecture for Distributed Metadata Management. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, San Jose, USA, 2002.
11. R. Koster, A. Black, J. Huang, J. Walpole, and C. Pu. Infopipes for Composing Distributed Information Flows. In *Proceedings of the International Workshop on Multimedia Middleware (M^3W'01)*, Ottawa, Canada, October 2001.
12. F. Leymann. Supporting Business Transactions via Partial Backward Recovery in Workflow Management Systems. In *Proceedings of BTW'95*, pages 51–70, Dresden, Germany, March 1995. Springer Verlag.
13. J. Meng, S. Su, H. Lam, and A. Helal. Achieving Dynamic Inter-organizational Workflow Management by Integrating Business Processes, Events, and Rules. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, Big Island, Hawaii, USA, January 2002.
14. R. Müller and E. Rahm. Rule-Based Dynamic Modification of Workflows in a Medical Domain. In *Proceedings of BTW'99*, pages 429–448, Freiburg, Germany, March 1999. Springer Verlag.
15. M. Reichert and P. Dadam. $\text{ADEPT}_{\text{flex}}$ — Supporting Dynamic Changes of Workflows without Losing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, March 1998.
16. H.-J. Schek, K. Böhm, T. Grabs, U. Röhm, H. Schuldt, and 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.
17. H.-J. Schek, H. Schuldt, and 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.
18. H. Schuldt, G. Alonso, C. Beeri, and H.-J. Schek. Atomicity and Isolation for Transactional Processes. *ACM TODS*, 27(1), March 2002.
19. C. Schuler, H. Schuldt, and H.-J. Schek. Supporting Reliable Transactional Business Processes by Publish/Subscribe Techniques. In *Proc. of the 2nd International Workshop on Technologies for E–Services (TES'01)*, Rome, Italy, September 2001.
20. R. Weber, H.-J. Schek, and S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proc. of 24rd International Conference on Very Large Data Bases*, New York, USA, August 1998.
21. I. Wladawsky-Berger. Advancing the Internet into the Future. Talk at the *International Conference Shaping the Information Society in Europe 2002*, April 2002. `http://www-5.ibm.com/de/entwicklung/academia/index.html`.
22. XML pipeline Definition Language. `http://www.w3.org/TR/xml-pipeline`.