TOWARDS CSPACES: A NEW PERSPECTIVE FOR THE SEMANTIC WEB

Francisco Martín-Recuerda
Digital Enterprise Research Institute (DERI)
Leopold-Franzens Universität Innsbruck, Austria

francisco.martin-recuerda@deri.org

Abstract: Information overload is mainly the result of the combination of four factors: the enormous amount of information available; the heterogeneity of information sources and information channels; the generation of a significant percentage of redundant information; and inefficient mechanisms for filtering, searching and classifying information. Given that the former factor cannot be changed, and current forecast expects that information grows exponentially in the next years, research and industry efforts are focusing to overcome the other three. The association of machine-understandable semantics to formally describe data published on the Web and the development of appropriate tools that can handle this method to describe data are the approaches that the promoters of the Semantic Web have suggested to overcome the problem of information overload in the Web. Although, the Semantic Web promises a new level of service with regard to the current Web, a more drastic approach is required. Conceptual Spaces (CSpaces) envision the future of the Semantic Web as a cooperative environment where communication between humans, machines, and human-and-machines will be reduced to the acts of publishing and reading machine processable semantics in a persistent collection of individual and shared information spaces. Decreasing the amount of syntactic data representation in the Semantic Web, and therefore, make machine processable semantics the prevalent representation formalism will facilitate interoperation between heterogeneous applications, web services, agents, humans and so on. Natural language generation and graphical knowledge visualization techniques will make possible that humans deal with this "purest semantic" Web. In addition, CSpaces will also decrease redundancy of the information stored and will provide a better organization of the data articulated around ontologies.

Key words: Information overload, Semantic Web, metadata, Triple Space Computing, Event-based, Ubiquitous Computing, Peer-to-Peer, Tuple Space, personal and distributed knowledge management.

1. INTRODUCTION

In the past decades the situation of a shortage of accessible information has been gradually changing thanks to new communication means like electronic mail and the World Wide Web. In fact, the exponential growth of the information available (e.g. in 2002 our society produced around 5 exabytes[1] of new information mostly stored in hard disks [1]) distributed in several heterogeneous information channels (i.e. emails, fax, instant messages, web pages, etc) have increased the difficulty to organize, find, integrate and reuse this flow of data. Furthermore, these information channels are not coordinated and generate a significant amount of redundant information. The combination of huge amount of information, diversity of information channels, a significant amount of redundant information, and inefficient mechanisms for filtering, searching and classifying information contribute to the appearance of a new phenomenon difficult to overcome: information overload. One approach to alleviate this situation in the concrete scenario of the Web came from the W3C, and it is called Semantic Web [2]. The Semantic Web stands for the idea of a future Web which aims to increase machine support for the interpretation and integration of information. Annotating the Web with formal semantic descriptions together with domain theories (i.e., ontologies) will enable a Web that provides more effective discovery, automation, integration, and reuse of the information currently stored [3]. Unfortunately, the development of the Semantic Web does not deal with the problems of diversity of information channels and redundancy. Moreover, the Semantic Web is an ongoing research effort where several relevant questions are still open. In the scope of this paper, how to physically and scaleably organize and store metadata in a global scenario, how to restrict the access to this metadata, how to guarantee trustworthiness of the available metadata, and how to facilitate the encoding, access and visualization of semantic data representations for human actors, are my major concerns. Also, the dichotomy of Web data (syntactic-textual representation) and semantic annotations will require permanent significant efforts to maintain coherence on both sides and will not reduce the problem of data-redundancy on the Web side.

Information overload can be achieved using a platform based on semantic web technologies that takes into account the following guidelines:

- Unify information channels and sources in a suitable one that can be shared by humans and machines.
- Increase machine support for information management by making machine processable semantics the prevalent representation formalism in this new infrastructure.
- Reduce heterogeneity by introducing mechanisms that allow humans to find agreements in the representation and definition of common terminology and semantic data specifications.
- Promote the use of ontologies as articulation means for organizing data and identifying redundant data.

Conceptual Spaces (CSpaces) envision the future of the Semantic Web as a cooperative environment where communication between humans, machines, and human-and-machines will be reduced to the acts of publishing and reading machine processable semantics in a persistent collection of individual and shared information spaces. Applications and humans can be owners of several Individual CSpaces and can be members of several shared CSpaces (preferably one for each Individual Cspace). Access rights can be easily associated to CSpaces, and interoperability can be improved through the use of Shared CSpaces. Decreasing the amount of syntactic data representation in the Semantic Web, and therefore, make machine processable semantics the prevalent representation formalism will facilitate interoperation between heterogeneous applications, web services, agents, humans and so on. Natural language generation and graphical knowledge visualization techniques will make possible that humans deal with this "purest semantic" Web.

The evolution of present Semantic Web proposal into CSpaces can increase the range of applications and benefits for industry, research and education areas. In particular, I am investigating the applicability of CSpaces in personal and distributed knowledge management, enterprise application integration, Semantic Web services, software components coordination and ubiquitous computing.

The paper is organized as follows. Section 2 provides a short overview of current state of the art and open issues on related Semantic Web technologies. Section 3 describes some relevant building blocks that will constitute Conceptual Spaces. Section 4 discusses some possible applications of this technology. Finally, conclusions and future work are provided in section 4.

2. STATE OF THE ART IN RELATED SEMANTIC WEB TECHNOLOGIES

CSpaces have been invented to minimize the problem of information overload, and as a response to the questions: how to organize and store metadata, how to restrict the access to this metadata, how to guarantee trustworthiness of the available metadata, and how to facilitate the encoding, access and visualization of semantic data representations for human actors, that Semantic Web does not address yet. In this section, a review of the state of the art of relevant Semantic Web technologies is described and significant gaps are identified. This analysis will drive to the necessity to develop a new proposal for organizing, protecting and sharing machine processable semantics.

2.1 Machine processable semantics organizational model

There is not agreement in the Semantic Web community about how to organize ontologies, instances, rules and mappings (alignments) between them. [4] suggests an hybrid approach called ontology islands, where ontologies are mapped to concrete influential domain ontologies , where within the island there is a form of global integration; one ontology would be the global ontology of the islands and a number of local ontologies are mapped to this global ontology. Unfortunately this approach mainly focuses in the mapping problem, and more general model is required.

HCONE [5]introduces three different spaces in which ontologies can be stored: Personal Space, Shared Space and Agreed Space. Ontologies are created in Personal Spaces and shared in Shared Spaces where users can discuss ontological decisions. When users reach an agreement, ontologies are moved to Agreed Spaces. Although closely related with CSpace approach, HCONE is more oriented for collaboratively building ontologies in a restricted community of users than for a Semantic Web scale organizational model.

Outside of the Semantic Web, [6, 7] proposes a distributed knowledge management infrastructure based on Kpeers. An interesting point of this solution is that knowledge is created and shared using a bottom-up approach. Users create their own knowledge that they share with other users, creating bigger knowledge bases.

Finally, CO4 (Collaborative construction of consensual knowledge bases) [8] is an infrastructure enabling the collaborative construction of a knowledge base through the web. The main contribution of this approach is a proposal for organizing KBs in a tree structure. The leaves are called user KBs, and the intermediate nodes, group KBs. Each group knowledge base

represents the knowledge consensual among its sons (called subscriber knowledge bases). This solution is closely related with the idea of ontology islands presented before, and it can be easily adapted as a organizational model of machine processable semantics for the Semantic Web.

2.2 Coordination model for distributed systems

Middleware is the "glue" that facilitates and manages the interaction acroos heterogeneous computing platforms. During the last three decades, several coordination models and infrastructures like RPC [9], Message passing, Message Queues [10], Tuple Space [11] and Publish- Subscribe [12] bring the attention of developers of concurrent and distributed systems. Currently, major efforts are focused on Web Services[2] and adding semantics to Web Services[3]. However, [3] and [13] criticize the use of message passing paradigm for Web Services because does not rely in Web principles (asynchronous publish and read model). Thus, they propose an alternative coordination model, called Triple Space, that enriches Tuple Space model using RDF triples instead of unformatted tuples. Parallel work published in [14] reached the same conclusion.

[12] identifies three desirable orthogonal dimensions for coordination models that are extended to four in [15]:

- Space decoupling: processes involved in the interaction can run in completely different computational environments.
- Reference decoupling: processes involved in the interaction do not need to know each other (anonymous).
- Time decoupling: processes do not need to be up at the same time during the interaction (asynchronous).
- Flow decoupling: main flows of process are not affected for the generation of reception of data (no blocking read (receive) and write (send) operations).

Table 1 (partially adapted from [12]) summarizes decoupling dimensions of five coordination models.

Table 1. Decoupling dimensions of several coordination models

Abstraction	Space	Time	Reference	Flow
Message passing	Yes	No	No	Producer-side
RPC	Yes	No	No	Producer-side[4]
Message queues	Yes	Yes	Yes	Producer-side
Tuple Space	Yes	Yes	Yes	Producer-side

[2] http://www.w3.org/2002/ws/

[3] http://www.daml.org/services/, www.wsmo.org and http://lsdis.cs.uga.edu/projects/meteor-s/

[4] In the case of RPC is not clear which process is producer and which is consumer. [9] suggests that consumer plays the role of invoker and producer lays the role of invokee.

Abstraction	Space	Time	Reference	Flow
Publish - Subscribe	Yes	Yes	Yes	Yes

[15] identifies some drawbacks in Tuple/Triple Space. For instance it does not provide flow decoupling dimension, and the model expects a tacit agreement between producers and consumers regarding the format of the data that it will be published in the space (in other words, there is no way to know before hand which is the format of the data that producers will publish information in the space, and therefore, there is no way to know which data format the consumers expect). [15] proposes a new model that combines main features of Tuple/Triple Space and Publish-Subscription models. Further on, I provide a detailed description of this paradigm and how can become in the coordination model for CSpaces. As a consequence of this decision, the architecture proposal for CSpaces has to take into account the requirements of this new coordination mechanism.

2.3 Semantic interoperability

Data integration approaches aim to provide a unified (or reconciled) view, called global or mediated schema of different heterogeneous data sources (local schemas) [16]. Two basic approaches have been used to specify the mapping between local schemas and global schema: global-as-view (GAV) and local-as-view (LAV). The first approach, defines the global schema in terms of the local schemas. In the second approach, the global schema is defined independently from the local schemas, and those local schemas have associated a description of themselves in terms of the global schema. Examples of the GAV approach are TSIMMIS [17], InterViso [18] and Garlic [19] while examples of the LAV approach are IM [20] and Agora [21].

[22] reports that the major disadvantage of GAV is that it is complex and expensive to support evolution of local schemas. On the other hand, it is also well known that processing queries using LAV approach is a difficult task ([22] and [23]).

[24] and [16] propose alternative approaches to combine the advantages of GAV and LAV. The former, called "both-as-view" (BAV), is built on top of a low level hyper-graph-based data model (HDM) and a set of primitive schema transformations for this model. The second approach proves that its is possible to derive LAV view definitions in to GAV view definitions in very restricted scenarios where views are expressed as conjunctive queries, and the global schema is defined in the relational data model with inclusion dependencies.

The Semantic Web devises a more complex scenario for data integration where LAV, GAV and BAV can also be applied [25] and [26]. Ontologies define a more sophisticated schema specifications base on rich formal representation languages. Given the fact that reasoning tasks over expressive languages are very expensive in terms of computational resources, and scenario of mapped heterogeneous ontologies would drive to unacceptable time response of inference systems. Moreover, there would be cases in which semantic heterogeneity of several mapped ontologies cannot be completely reconciliated, and thus inconsistent problem can arise ([25] and [27]). Deal with inconsistencies would require the use of concrete techniques that can degrade even more performances of inference systems.

[27] discusses an alternative approach to deal with queries for mapped heterogeneous ontologies. The authors propose the generation of tailored reasoning spaces for each of the queries that the system receives. The drawback of this approach is that the time necessary to generate such space can be more expensive that perform the query over the relevant mapped ontologies using query rewriting techniques. However, the idea of improving reasoning performance by creating reasoning spaces is one of the motivations because I propose to organize data semantics in the Semantic Web in individual and shared conceptual spaces.

2.4 Knowledge visualization and natural language generation (NLG)

Knowledge visualization comprises all the techniques and mechanisms that facilitate the exploration and visualization of semantic formal representation of information stored in knowledge bases. Knowledge visualization aims to improve the creation, comprehension and transfer of knowledge by exploiting graphical and natural language processing means.

Graphical representation of knowledge was intensively studied in the previous decades and is still ongoing research (please refer [28] for a survey). The popularization in the use of Ontologies brings into focus the necessity to provide graphical visualization as an essential feature for every tool for ontology editing and browsing. Tree and graph visualization approaches are the more common techniques to graphically represent ontologies. A concrete solution for displaying large tree structures, called *hyperbolic tree* [29], was developed in 1995 in Xerox Parc Laboratories and commercialized by Inxight[5] . This technique is used in tools like KAON[6] and KIM[7].

[5] http://www.inxight.com
[6] http://kaon.semanticweb.org/
[7] http://dell.sirma.bg/kim/graph/Graph.jsp

A complementary approach for knowledge visualization in which semantic data descriptions are presented in a user friendly way is natural language generation (NLG). "*NLG takes structured data in a knowledge base as an input and produces Natural Language text, tailored to the presentational and the target reader*" [30]. NLG mechanisms can constantly keep up-to-date text descriptions of data semantics and can automatically provide those text descriptions in multiple languages [31]. Current efforts in NLG have two main focuses. The first one is to provide tools specific oriented to semantic web platforms, and the second one is to design NLG systems that keep the system simple enough to be maintained by non-NLG experts, but without losing quality of the text output ([32], [33], [34], and [35]). Since I expect that most of the users will not be NLG experts, these proposals can be very useful for the Semantic Web.

2.5 Distributed Architectures for storing and sharing data

The Web has been described using an abstract model called REST (Representational State Transfer) [36]. The fundamental principle of REST is that resources are stateless and identified by URIs. [37] demonstrates that it is not possible for a server to transmit any information to a client asynchronously in REST because every representation transfer must be initiated by the client, and every response must be generated as soon as possible (the statelessness requirement). Asynchronous communication is a requirement that CSpaces will require, so we have studied several extensions of REST, like ARRESTED [37].

Peer-to-Peer system is an interesting proposal for decentralized, distributed, self-organized systems, capable of adapting to changes such and failure [38]. Although there are several open issues regarding scalability, shared resources management, security and trust [39], current efforts in the field [40,41] are progressively overcoming these problems.

P-Grid[8], Edutella[9] and OceanStore[10] are current state of the art in P2P systems. They are examples of decentralized architecture that address several interesting problems like optimizing message flooding (HyperCup in Edutella), increasing fault tolerance using replication (OceanStore), improving security via cryptography techniques (OceanStore) and generation of public key in a decentralized manner (P-Grid). However, none of them provide all features that can be desirable, so implementation of extensions to one of these systems should be studied.

[8] http://www.p-grid.org/
[9] http://edutella.jxta.org/
[10] http://oceanstore.cs.berkeley.edu/

3. BUILDING BLOCKS TOWARDS CONCEPTUAL SPACES (CSPACES)

CSpaces are built around six building blocks. Given space limitation of this paper, I will introduce briefly all of them and in the following subsections will describe in more detail four of them.

Semantic data, schema and organizational model. Data elements and their relations should be described using formal representation languages (e.g. RDF and RDF(S) for the Semantic Web) that include a set of modeling primitives. The relations between data elements and data properties should be constructed as Ontologies. Ontologies are distributed in Individual and Shared CSpaces where heterogeneity should be overcome by finding a unified representation of the information in which mismatches are identified and transformation rules are implemented.

Coordination model. CSpaces is a middleware infrastructure for applications and a cooperation infrastructure for humans that aims to unify on the one hand, several communication means like e-mail, faxes, weblogs, etc, and on the other hand, coordination mechanisms like tuplespace, publish-subscribe, message queues, etc. The coordination model combines two metaphors: "persistent publish and read" and "publish and subscribe".

Semantic interoperability and consensus making model. The identification and representation of mismatches, and the definition of transformation rules will be required to ensure interoperability between the participants in different CSpaces. Moreover, consensus techniques are required to build Shared CSpaces. Humans need to reach an agreement about which information will be shared (content agreement) and how it will be represented (semantic agreement).

Security and Trust model. The protection of private and restricted information stored on spaces and the inclusion of trusted mechanisms to guarantee the validity, or trustworthiness of the information accessed are critical requirements for a successful development of a distributed information infrastructure.

Knowledge visualization model. Given the fact that the final goal of CSpaces is to minimize the amount of syntactic data representation (current Web) and maintain mostly semantic descriptions of data, it is necessary an infrastructure with knowledge visualization facilities to easily interpret the information stored through graphical and natural language generation mechanisms.

Architecture model: scalable, decentralized, distributed and secure are four design goals that CSpace infrastructure has to achieve. It would be also interesting the implementation of replication, searching, routing and data allocation (besides others) services.

3.1 Defining Conceptual Spaces: semantic data, schema and organizational model

Nowadays, there is a debate in the knowledge representation field about how ontologies, rules and alignment specifications should coexist. [45] argues that a set of ontologies and alignment specifications is more suitable for the Semantic Web because provides a loosely coupled configuration where updates in the mapped ontologies can be easily incorporated. On the other hand, a merge configuration of several ontologies can provide a more coherent, compact and consistent reasoning space and avoid the necessity to include reasoning mechanisms that have to deal with inconsistencies. Also the applicability in real scenarios of query rewriting to query mapped ontologies is limited in practice [27]. Based on previous experiences, I propose a new organization of the machine processable semantics that will be provided by the Semantic Web. This organization is articulated around Conceptual Spaces.

A **Conceptual Space** (CSpace) is a finite set of ontologies, their instances, and mapping and transformation rules (alignment specification). All these elements are represented using a common formal language that allows ontologies to be enriched with rules[11], and exhibit some degree of semantic autonomy[12].

Ontologies, which provide an unambiguous definition of the meaning of terminology/vocabulary used to describe a concrete domain, are used as a skeletal foundation for a Knowledge Base. An ontology can be formally defined as a 4-tuple < C, R, I, A > [45] where C is a set of concepts, R is a set of relations, I is a set of instances and A is a set of axioms. An ontology has to be specified in a formal logical language, i.e. a formalism with a well-defined semantics (OWL[13] and RDF(S)[14], are current relevant standards for Semantic Web).

[11] Although we have not considered at this moment representation formalisms to characterize uncertainty or vagueness, future work should evaluate and integrate these mechanisms.

[12] Semantic autonomy represents a particular perspective of the world of an individual or group of agents (humans or not). This semantic autonomy is represented by a set of schema specifications that organize and classify information according with an individual or shared interpretation.

[13] http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[14] http://www.w3.org/TR/rdf-schema/

Rules are generally of the form of an implication between an antecedent (body) and consequent (head). The meaning of a rule can be informally described as: "*whenever (and however) the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold*"[46]. A rule has the form: *consequent* ← *antecedent*, where both are a conjunction of atoms, R(t1, ..., tn) composed by variables and/or constants. Rules are becoming very popular in the Semantic Web area because they enable one to enrich ontological specifications ("*build ontologies on top of rules*" [46]) and to build rules using the vocabulary defined by ontologies ("*build rules on top of ontologies*" [46]). Rules can also be very useful to improve query capabilities and provide mapping descriptions and data transformation specification for data integration.

Alignment specification, a set of mapping descriptions and transformation rules to handle heterogeneity between two or more ontologies. Mappings are typically expressed by some form of logic programming style rules, offering the expressivity of a powerful query language are a natural choice. However, the mapping language to be used does not provide any formal grounding, leaving open the choice of the appropriate semantics for the mappings; as a consequence, a formal semantics compatible with Description Logics and/or Logic Programming can be defined [27].

All logical statements have associated three identifiers. The first one is the identifier of the context where they were created (**id_context**). The second identifier is a unique id for the logical statement (**id_statement**, which can simplify reification, and make the code more compact, and the third one is a unique version id (**id_version**) that identifies each version of a logical statement.

In a CSpace, I can distinguish between raw and reasoning sub-space. A **raw** sub-space stores imported or local data, schemas, and alignment specifications (mapping and transformation rules) between these schemas. On the contrary, a **reasoning** sub-space provides a compact representation of an associate raw sub-space. The main goal of this compact representation is to maximize reasoning performances.

Ontologies, rules and alignments are maintained and updated on raw sub-spaces (one for each CSpace). The associated reasoning sub-spaces are periodically re-generated with the last version of the raw sub-space. In this regeneration (or initial generation) process, ontologies and rules are merged based on the alignment specification stored in the raw sub-space. A refinement and validation process identifies inconsistencies in the merged ontology and supports the user during the resolution of those inconsistencies. The resultant knowledge base is optimized in order to achieve better reasoning performances. Current proposals are studying the applicability of

language weakening, knowledge compilation and approximate deduction in the context of the Semantic Web.

I distinguish two types of CSpaces, individual CSpaces and Shared CSpaces. An Individual CSpace is a formal representation of the perception that each individual (human or not) has about the Semantic Web (or a limited part of it). The machine processable semantics stored in an Individual CSpaces can be private (only the owner of the space can access it), restricted (a limited number of individual can access it) or public (the information can be accessed without restriction). The combination of restricted and public data can be used to create Shared Conceptual Spaces. Shared CSpaces are conceptual spaces shared by several participants that have reached an agreement on how to represent semantically common concepts. This requirement is fundamental to ensure interoperability between participants.

CSpaces can be viewed as leaves and shared spaces can be graphically considered as the branches and the trunk of a fictitious tree following a very similar organization proposed in CO4 (Collaborative construction of consensual knowledge bases) [8]. CO4 is an infrastructure enabling the collaborative construction of a knowledge base through the web. The main contribution of this approach is a proposal for organizing KBs in the same way that we proposed for CSpaces, as a tree structure. The leaves are called user KBs, and the intermediate nodes, group KBs. Each group knowledge base represents the knowledge consensual among its sons (called subscriber knowledge bases). When a subscriber wants to extend their group knowledge base, he/she submits a proposal with the modifications to the other subscribers. In response, users must answer by one of the following: accept when they consider that the knowledge must be integrated in the consensual knowledge base, reject when they do not, and challenge when they propose another change.

On the contrary of CO4 where modifications need to be approved by all subscribers, the updates proposed by the members of a Shared CSpace are automatically included, and versioning mechanisms are in charge to track changes and provide rollback features if one of the members disagrees with the included updates.

To join a Shared CSpaces and publish and retrieve data on it the new members should first complete a registration procedure in which one of the main tasks is to provide a semantic and alignment specification between the data that each new candidate want to share and the data that previous members have published beforehand.

3.2 Coordination model: "publish, read and subscribe"

Thanks to the Web, humans can *persistently publish and read* information at any time stored on servers spread around the World. The "*persistent publish and read*" metaphor have been also applied successfully as a simple coordination model for parallel computing [47], and more recently to Semantic Web Services [3]. Tuple-Space [47] is a coordination mechanism in which synchronization and communication between participants take place through the insertion and removal of tuples to/from a common shared space. Shared, persistent, associative, transactional secure and synchronous/asynchronous communication are main properties of Tuple Space.

However, "*persistent publish and read*" has the drawback that does provide flow decoupling from the client side. The interaction model provides time, space decoupling but not flow decoupling [12]. A user who is interested on an update version of a concrete web page has to check periodically for new contents although those contents are not already published. This restriction is a consequence of the REST (Representational State Transfer) [36] architecture style that characterizes the Web. To overcome this limitation, the "persistent publish and read" metaphor has been extended into "*persistent publish, read and subscribe*". The popularization of "weblogs[15]" (blogs or bloggings) together with the development of RSS (Rich Site Summary or Really Simple Syndication, http://www.rss-specifications.com/) bring a new form of interaction for web-users based on content subscription[16].

TupleSpace has the same limitation from the reader-side. An application which wants to read a concrete tuple has to run a process that periodically checks if the data is available. JavaSpaces[17] and TSpaces[18], concrete java implementations of TupleSpace, provide a simple notification mechanism to mitigate the problem. Thus Event-based technology can complement Tuple-Space with a sophisticated notification and subscription mechanism that allow a proper asynchronous interaction from the consumers/reader side. Additionally, in Tuple Space, it is not possible to know before hand which is the format of the data that producers will use to publish information in the space, and therefore, there is no way to know which data format the consumers expect. An implicit agreement is expected, but in the Semantic

[15] A website which stores miscellaneous notes updated daily and published in chronological order [18]

[16] RSS is still based on polling, but it is invisible for end-user. Users subscribe, but their RSS client is just polling the webserver every x minutes.

[17] http://java.sun.com/developer/products/jini/index.jsp

[18] http://www.research.ibm.com/journal/sj/373/wyckoff.html

Web where millions of users and applications will interact, these implicit agreements are not feasible.

On the other hand, a main disadvantage of most TupleSpace and Event-based implementations systems is the limited ability to define matching templates. Since CSpaces will store schema and data semantics, the coordination model based on the integration of TupleSpace and Event-based technologies have to be extended to support machine processable semantics[19]. For instance, in Triple Space Computing [3] the data published and accessed is represented by RDF triples. [48] proposes an equivalent approach for event-based systems using DAML+OIL to express a more accurately subscriptions and to improve event filtering mechanisms. CSpaces integrate tuplespace, event-based operations and semantic data specification in a new coordination model. Table 2 shows a reduced version of the first specification of the coordination model API for CSpaces. This coordination API is inspired by the combination of TSpace API and SIENA API[20].

Table 2. Coordination Model API for CSpaces

API call and description		
Void	write	(set tuples, IdCSpace id)
	Write one or more tuples in a concrete CSpace identified by a unique id. This method is shared by tuple-space and Event-based implementation	
Tuple	take	(Template t, IdCSpace id)
	Return a tuple (or nothing) that match with the template (that can be expressed using a formal query language) and delete the matched tuple from a concrete CSpace	
Tuple	waitToTake	(Template t, IdCSpace id)
	Like take but the process is blocked until the a tuple is retrieved	
Tuple	read	(Template t, IdCSpace id)
	Like take but the tuple is not removed	
Tuple	waitToRead	(Template t, IdCSpace id)
	Like read but the process is blocked until the a tuple is retrieved	
Set	scan	(Template t, IdCSpace id)
	Like read but returns all tuples that match with t	
Long	countN	(Template t, IdCSpace id)
	Return the number of tuples that match template t	
Void	subscribe	((IdSubscriber s, Template t), IdCSpace id)
	A subscriber expresses its interested on tuples that match with template t in a concrete CSpace. Any time that there is an update in the CSpace, the subscriber receives a notification that there are tuples available that match the template	
Void	unsubscribe	((IdSubscriber s, Template t), IdCSpace id)
	A subscriber deletes its subscription, and no more related notifications are received	
Void	advertise	((IdPublisher p, Template t), IdCSpace id)
	A publisher shows its intention to provide tuples that match t	
Void	unadvertise	((IdPublisher p, Template t), IdCSpace id)

[19] CSpace coordination model = "persistent publish, read and subscribe" + "semantics"

[20] http://serl.cs.colorado.edu/~carzanig/siena/

API call and description
A publisher shows will not provide more tuples that match t

In a Semantic Web mostly composed by machine processable semantics, "persistent publish, read and subscribe" metaphor can be the common interaction model for machines and humans. I think that this metaphor is flexible enough (as Weblogs, Tuple Space and Event-based demonstrated) to progressively substitute other communication means like fax, e-mail, instant messages, and present Web pages. By unifying those communication channels, CSpaces coordination model can contribute to mitigate one of the most relevant sources of information overload.

3.3 Knowledge Visualization

At this initial stage, Knowledge visualization user interface will be implemented reusing and combining two different approaches. I will provide a graphical navigation interface based on *TouchGraph*[21] (a popular general-purpose hyperbolic tree visualization library). To facilitate the understanding of the information showed by the graphical interface, I decided to integrate ONTOSUM [35], a generator for textual tailored summaries from ontologies. I chose ONTOSUM because is based on a well tested technology [32], it is domain-independent, it is designed for non-NLG experts, and it supports entries in different formal ontology languages like RDF(S), DAML+OIL and OWL.

ONTOSUM is implemented as a pipeline system [30] inside of the GATE infrastructure [49]. Althought the integration with GATE reports a lot of benefits, it would be interesting to disaggregate the NLG components and build an independent tool that can be executed in light-weight devices. The generator, HYLITE+, is implemented in Prolog and can run in diverse platforms. However, other components for preprocessing data semantic inputs are required (including a light-weight ontology called PROTON[22]), and thus, they should be adapted to work independently from the GATE system.

3.4 Architecture model: a decentralized hybrid approach

Given that CSpaces aims to re-elaborate the Semantic Web proposal by minimizing syntactic data representation, many of the design considerations

[21] http://sourceforge.net/projects/touchgraph/
[22] http://proton.semanticweb.org/

for the Semantic Web architecture are still valid for CSpaces. Scalable, distributed and decentralized are three requirements that CSpace and Semantic Web architectures have in common. However, the CSpace coordination model built over "persistent publish, read and subscribe" metaphor requires an architecture model that can deal with asynchronous communication. Furthermore, the organization of metadata around Individual and Shared CSpaces is another different in both infrastructures.

Like Semantic Web, my first idea was to build CSpaces upon existing Web infrastructure that has been described using an abstract model called REST (Representational State Transfer) [36]. The fundamental principle of REST is that resources are stateless and identified by URIs. HTTP is the protocol used to access to the resources and provides a minimal set of operations enough to model any applications domain [36]. Those operations (GET, DELETE, POST and PUT) can be easily mapped to Tuple-Space operations (READ, TAKE and WRITE in TSpaces). Tuples can be identified by URIs and/or can be modeled using RDF triples (as [3] suggests). Since every representation transfer must be initiated by the client, and every response must be generated as soon as possible (the statelessness requirement) there is no way for a server to transmit any information to a client asynchronously in REST. Furthermore, there is no direct way to model a peer-to-peer relationship [37]. Several extensions of REST, like ARRESTED [37], have been proposed to provide a proper support of decentralized and distributed asynchronous event-based web systems.

The limitations of REST to model asynchronous interaction motivated that I pay attention to Peer-to Peer [12] systems. They are decentralized, distributed, self-organized and capable of adapting to changes such as failure [38]. Although there are several open issues regarding scalability, shared resources management, security and trust [39], current efforts in the field [40,41] are progressively overcoming these problems.

My preliminary proposal for CSpaces architecture is strongly influenced by the work done in OceanStore[23], Edutella[24] and SWAP[25]. I distinguish between three kinds of nodes: CSpace-servers, CSpace-heavy-clients and CSpace-light-clients.

CSpace-servers store primary and secondary replicas of the data published in individual and shared CSpaces; support versioning services; provide an access point for CSpace clients to the peer network; include reasoning services for evaluating complex queries; implement subscription mechanisms related with the contents stored; balance workload and monitor

[23] http://oceanstore.cs.berkeley.edu/
[24] http://edutella.jxta.org/
[25] http://swap.semanticweb.org/public/index.htm

requests from other nodes and subscriptions and advertisements from publishers and consumers.

CSpace-heavy-clients provide a storage infrastructure and reasoning support to let users to work off-line with their own individual and shared spaces. Replication mechanisms are in charge to keep replicas in clients and servers up-to date. In addition, these clients also include a presentation service (based on NLG and Knowledge visualization techniques) to facilitate the visualization and edition of knowledge contents.

CSpace-light-clients only include the presentation infrastructure to query, edit and visualize knowledge contents stored on CSpace-servers.

When clients are online and connected with the rest of the nodes of the system through an access point (server node) they have the obligation to share computational resources (CPU time, memory and persistent storage services). Thus CSpace-servers can divert to client's resources demanding requests, and consequently, alleviate temporary the workload of servers. If the client is a heavy-client, requests that can be performed locally will not be sent to CSpace-servers. Periodically, replicas will be updated to keep heavy-clients and servers up-to-date.

I call these types of systems hybrid because elements of both pure P2P and client/server systems coexist. CSpace-servers are formally peers, but it is not the case of CSpace-clients that promote a client-server relation with CSpace-servers. Like OceanStore [40], this configuration drives into two-tiered system. The upper-tier is composed by well-connected and powerful servers, and the lower-tier, in contrast, consists in clients with limited computational resources temporary available.

It is expected that CSpaces infrastructure will be self-organized like in other peer-to-peer systems and will include monitoring mechanisms that will analyze the distribution of the data in the different nodes and the data flows between these nodes. Servers and clients will be re-distributed in appropriate configurations that minimize the network traffic and maximize semantic similarity of the data stored in closer peer. Subscriptions and advertisements from publishers and consumers will provide useful information to determine optimal configurations where consumers and publishers with common interests will be connected to closer servers. In addition, the definition of Shared CSpaces will be other information source to determine semantic similarity between nodes.

Communication metaphor will differ from most of the P2P implementation that use message passing. Like OceanStore is built on top of an event-based architecture[26], CSpace promotes the coordination model "publish, read and subscribe" for the communication of its nodes. In

[26] More precisely is Pond [22], the OceanStore prototype, which is built on top of an event-based system

addition, the use of topologies that simulate spanning trees (i.e. HyperCup in Edutella) will reduce unnecessary data flows and will facilitate the implementation of replication mechanisms.

Given the decentralized nature of CSpace infrastructure, solutions for security and trust mechanisms have to rely on the same principles. Currently, there are several proposals that aim to provide in a decentralized manner solutions for building public key infrastructures [42], restricting access to concrete contents [40], avoiding manipulation from malicious peers of data flowing in the network [43], and defining trust values for each peer without centralized globally-trust servers [44]

OceanStore incorporates several interesting solutions that will be studied in detail to verify their suitability for CSpaces. The authors of OceanStore propose a distributed system where read-only versions of the data will be kept it in the system. Support for fault tolerance is achieved by including a configuration of primary and secondary replicas, and the use of a technique that splits each data object into several fragments that can be dispersed in different servers. Those fragments can be recovered to re-construct the original data object even if some of the fragments are not accessible. Cryptography features are also incorporated to increase security access to the system and mechanisms to guarantee the validity, or trustworthiness of the information accessed.

4. SPACES IN THE REAL WORLD

In my opinion, the combination of an intensive use of semantic data descriptions, a flexible coordination model and the implementation of visualization mechanisms based on graphical and natural language technologies can be the basement for a new set of applications for industry, research and education areas. In particular, I am investigating the applicability of CSpaces in personal and distributed knowledge management, enterprise application integration, Semantic Web Services, software components coordination and ubiquitous computing. The former is closely related with the problem of information overload. In the following subsections, it will be briefly introduced these potential application scenarios for CSpaces.

4.1 Personal and distributed knowledge management

"***Personal Knowledge Management (PKM)*** *is a collection of processes that an individual needs to carryout in order to gather, classify, store, search and retrieve knowledge in his/her daily activities. Activities are not confined*

to business/work-related tasks but also include personal interests, hobbies, home, family and leisure activities" [61]. As today's knowledge workers often have to deal with data, information and knowledge specified in various formats (e.g. hard copy, video, picture, texts, voice message etc.), distributed using different information channels (e.g. emails, fax, instant messages, file systems, etc) and stored using multiple electronic devices for communications, planning and recording purposes [61].

A potential way to alleviate this situation is to use semantic data representation of the information that it is received or distributed, and to reduce the number of information channels. As it was mentioned earlier in the paper, I envision "persistent publish, read and subscribe" metaphor for machine processable semantics as the common interaction model for machines and humans that progressively substitute other communication means. By unifying those communication channels, CSpaces coordination model can contribute to mitigate one of the most relevant sources of information overload and to improve personal knowledge management. Furthermore, the use of knowledge visualization and natural language processing techniques will facilitate the manipulation of data semantics by humans and consequently will reduce significantly the amount of syntactic data representation and textual information. The benefit of this reduction will minimize duplicities (syntactic and semantic) of data representation.

CSpaces can contribute to organize and share knowledge using a bottom-up approach. Instead of centralized systems that forces users to agree in a set of rules, schemas and data, CSpaces offer a distributed infrastructure where users can publish personal knowledge that can be shared with other users with common information/interests. This approach is inspired in an earlier proposal called Distributed Knowledge Management [6, 7] where its authors confirmed during the realization of several tests in real scenarios that users were more favorable to this kind of approach because it takes into account the different perspectives and understandings that users have about the world and more concretely about the information, processes and interactions of their organizations or working groups. The combination of Individual CSpaces can generate a new space shared by all these users. Shared CSpaces is built on top of a semantic data representation agreement of a group of users. Moreover, Shared CSpaces can be combined to generate bigger Shared CSpaces, or in other words, bigger knowledge repositories.

4.2 Enterprise Application Integration

Given that one of the main goals of CSpaces is to provide a collection of homogeneous semantic spaces in which heterogeneity of data sources are

reconciled, CSpaces can be an excellent approach to handled smoothly Enterprise Application Integration (EAI)[27].

Integration and heterogeneity are two concepts that come together. Heterogeneity is one of the hardest problems that humans and machines have to overcome in order to ensure interoperability, and in a distributed and open system like Internet, heterogeneity cannot be avoided [50]. Several attempts to classify source/levels/categories of heterogeneity can be found in the literature (refer [50] for a survey).

The process of reconciling differences between heterogeneous information sources is called mediation [51], and in the particular case of Ontologies is called Ontology mediation [45]. In general, the identification and alignment of heterogeneity between several ontologies or data sources in a mediation process is not fully automatic and in case of complex data sources very time consuming.

Many frameworks assumed wrongly that mediation can be done "on the fly". In my opinion, the initial identification and alignment of heterogeneity in a mediation process should be done before any possible interaction or collaboration, and because usually there are several ways to reconciliate two or more heterogeneous data sources, it is expected an interactive mechanism that facilitate consensual decisions between involved actors. Shared CSpace is this consensual mediated space where applications can publish and share their data. During the registration process to a Shared Space, applications publish relevant ontologies and rules using a consensual specification. Through the publish-subscription mechanism, applications indicate which kind data will publish in the space and which kind of data would like to consume. Thanks to the CSpace coordination model applications interact through the space by simple means of publishing and reading data semantically described.

4.3 Towards Semantic Event Oriented Architecture for Web Services (SEOA-WS)

The case of enterprise application integration can be easily extrapolated to Web Services and software components coordination. Web Services are built over three main building blocks: service oriented architecture, redesign of middleware protocols and standardization [10]. *Service Oriented Architecture* (SOA) is based on the idea that companies will publish interfaces of their applications as services that can be invoked by clients. The second block, the *redesign of middleware protocols* to work in decentralized

[27] EAI is the unrestricted sharing of data and business processes throughout the networked applications or data sources in an organization (http://www.webopedia.com/TERM/E/EAI.html).

environment in order to overcome the limitations of centralized middleware architectures in terms of trust and confidentiality. Finally, the last key block is a set of *standard languages* and protocols that eliminates the necessity of many different middleware infrastructures. As a relevant part of the Service Oriented Architecture (SOA), notification is expected to play an essential role in the development of asynchronous, loosely-coupled and dynamic systems, where entities receive messages based on their registered interest in certain occurrences or situations. WS-Notification [52] and WS-Eventing [53], bring again the event-based communication paradigm to the fore.

Following the main principles that the Semantic Web introduced to extend the current Web, Semantic Web Services proposes to add machine processable semantics to Web Services in order to reduce manual efforts during the deployment of integration of distributed applications by improving automation in the location, combination and use of Web Services. For software architects, Semantic Web Services are the building blocks to evolve Service Oriented Architecture into **Semantic Service Oriented Architecture** (**SSOA**). Unfortunately "*adding semantics*" is not enough. Heterogeneity has become in an insurmountable obstacle that current proposals for web services and semantic web services[28] are not able yet to tackle. The vision of a global distributing computing through web services only could become true if all the participants involved provide mechanisms to achieve a common explicit formal understanding of their semantic specifications.

In my opinion, Shared CSpaces can facilitate a more effective discovery, invocation and interoperation of Semantic Web Services that are register to the same space. Heterogeneity problems are reduced thanks to the mediated data semantics published in Shared CSpaces and the coordination model of CSpaces provides a simple interaction mechanism for Semantic Web Services. The description of the data that the semantic web services registered in a Shared CSpace plan to publish is stored in the publisher register, and the data that those services plan to consume is stored in the subscriber register. These descriptions of data publication and consumption can be viewed as a very simplified version of services capabilities and goals [54]. Moreover, I am currently studying the practicability of evolving WSMX architecture[29] into Semantic Event Oriented Architecture for Web Services (SEOA-WS), SSOA architecture based on event mechanisms. I am

[28] The inclusion of Mediators as a part of the WSMO (http://www.wsmo.org) architecture is a promising initiative to confront the problem of heterogeneity.

[29] WSMX (http://www.wsmo.org/wsmx/) is a reference implementation of an execution environment for the dynamic discovery, selection, mediation, invocation and interoperation for Semantic Web Services based on WSMO specification. WSMX follows SOA principles.

working in the integration of a simplify version of CSpace as a data semantic repository and coordination model for the components of the system [15].

4.4 Ubiquitous Computing

Pervasive or *Ubiquitous computing* was the vision of Mark Weiser[30] for a World saturated with computing and wireless communication gracefully integrated with human users. One of the typical examples of ubiquitous computing applications is *active environments* ([55, 56, 57] and [58]). Active environments are sensor-rich environments with computational and communication facilities that analyze users behave to anticipate potential new requirements or facilitate the developing of certain tasks in a natural way. The large number of sensors required in this kind of environment potentially produces a vast amount of data that it is necessary to filter to potential consumers to allow more efficient performances, and avoid their saturation. Thus, a middleware infrastructure is required to cope with high-volume data and be able to aggregate and transform it before dissemination.

Started at Stanford University in mid-1999, the Interactive Workspaces[31] project is a concrete example of application of the idea of active environment in laboratories and collaborative e-learning spaces[32]. One of the versions of the prototype for interactive workspace, called the iRoom [58], included iROS (Interactive Room Operating System), a middleware infrastructure designed to be embedded in all devices that belong to the iRoom. One of the components of this software is Event Heap [59], a coordination mechanism derived from a tuple space model, and implemented on top of the TSpaces (Tuple Spaces) system from IBM Research [60]. Unlike TSpaces, the Event Heap treats the fields as an unordered collection and allows references to tuple fields only by name, not by index. In addition, applications can specify incomplete query templates with only some combination of name, type and value for fields.

CSpaces can improve the Event Heap mechanism with a publish-subscription mechanism embedded in the Tuple Space system that provide a asynchronous behave from the client side and can facilitate the identification of data requirements from producers and consumers by analyzing the subscriptions stored in the system. Moreover, the use of data semantics would allow a richer specification of the information that flows through the system and would improve searching capabilities for retrieving operations.

[30] http://www2.parc.com/csl/members/weiser/
[31] http://iwork.stanford.edu/main.shtml
[32] http://www.stanford.edu/dept/SUL/acomp/teamspace/

5. CONCLUSIONS

CSpaces aim to create an infrastructure that reduces the problem of information overload and facilitates collaboration between machines, humans, and humans-and-machines, based on an intensive use of machine processable semantics. "*Publish, read and subscribe*" of machine processable semantics, a re-elaboration of Tuple Space and Publish-Subscribe systems, is the communication mean that pretend to reduce heterogeneity produced by a diverse set of heterogeneous information channels (emails, fax, instant messages, web pages, etc). Through natural language and knowledge visualization techniques, humans will be able to interact with this "purest semantic" web.

Machine processable semantics will be published and shared in CSpaces, a finite set of instances, ontologies, and mapping-and-transformation rules (alignment specification) that are represented using a common formal language, and exhibit some degree of semantic autonomy. To improve reasoning performances and do not difficult the update of the data stored, CSpaces maintain a reasoning sub-space and a raw sub-space, respectively.

Individual and Shared CSpaces will provide a logical organization, inspired in a tree model, of machine processable semantics in the Semantic Web. Individual CSpace is a formal representation of the perception that each individual (human or not) has about the Semantic Web (or a limited part of it). On the other hand, Shared CSpace represents the agreement of a group of humans and/or machines of how formally represent concepts of their Individual CSpaces. Through "publish, read and subscribe", humans and/or machines will be able to interoperate using a common Shared CSpace. Access rights associated to Individual and Shared CSpaces will assure different levels of privacy of the information published.

To complete the presentation of CSpaces, I sketched in section 3 the architecture that can take into account the requirements imposed by the Semantic Web and the "publish, read and subscribe" coordination paradigm. Since REST cannot model this coordination paradigm, and hybrid architecture based on P2P is my choice for CSpace infrastructure. Three kind of nodes (CSpace-servers, CSpace-heavy-clients, and CSpace-light-clients) are defined, and several proposals for security, trust and P2P topologies and communication mechanisms are discussed.

The paper concludes with a briefly description of the applicability of CSpaces in four different scenarios: personal and distributed knowledge management, Enterprise Application Integration (EAI), Semantic Event oriented Architecture for Web Services (SEOA-WS), and Ubiquotous computing.

CSpaces is in an early stage, so as a future work, I will continue the process of refining the ideas presenting in this paper, testing new tools that can be used in this framework, and working in the implementation of a first prototype.

ACKNOWLEDGEMENTS

I would like to thank Ying Ding and Uwe Keller (Leopold-Franzens Universität Innsbruck, Austria), Stefan Decker, Mathew Moran and Eyal Oren (National University Ireland Galway) and Manfred Hauswirth (Ecole Polytechnique Federal de Lausanne, EPFL) for fruitful discussion and feedback.

This work is funded by the European Commission under the projects Knowledge Web; by the Vienna city government under the CoOperate programme and by the FIT-IT (Forschung, Innovation, Technologie - Informationstechnologie) under the project TSC.

REFERENCES

1. School of Information Management and Systems (SIMS). How much Info? (Technical Report). University of Berkeley. 2003.
2. T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web. Scientific American, May 2001.
3. D. Fensel: Triple-based Computing. Digital Enterprise Research Institute (DERI) Technical Report DERI-TR-2004-05-31. 2004
4. J. de Bruijn, F. Martin-Recuerda, D. Manov, M. Ehrig, 4.2.1, State-of-the-art survey on Ontology Merging and Aligning V1, SEKT, 2004
5. K. Kotis, G.A. Vouros, J.P. Alonso, HCONE: tool-supported methodology for collaboratively devising living ontologies. In SWSDB 2004: 2nd int. Workshop on Semantic Web Databases , 2004
6. M. Bonifacio, P. Bouquet and R. Cuel. Knowledge Nodes: the Building Blocks of a Distributed Approach to Knowledge Management. Journal of Universal Computer Science, 8(6), 652-661. 2002
7. M. Bonifacio, P. Bouquet, G. Mameli and M. Nori, Peer-Mediated Distributed Knowledge Management, 2003, http://eprints.biblio.unitn.it/archive/00000426/01/032.pdf
8. J. Euzenat, Building consensual knowledge bases: context and architecture, in N. Mars (ed.), Towards very large knowledge bases, IOS press, Amsterdam (NL), pp143-155, 1995
9. A.D. Birrell and B.J. Nelson, Implementing remote procedure calls. ACM Transactions on Computer Science, 2(1):39-59, Feb. 1984
10. G. Alonso, F. Casati, H. Kuno, and V. Machiraju (2004). Web Services. Springer, 2004.
11. D. Gelernter, N. Carriero, S. Chang: Parallel Programming in Linda, Proceedings of the International Conference on Parallel Processing. ICPP. 255-263. 1985

12. P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. Technical Report. 2001.
13. C. Bussler, DERI-TR-2005-04-22 A Minimal Triple Space Computing Architecture, April 2005
14. Robert Tolksdorf, Lyndon Nixon, Franziska Liebsch, Duc Minh Nguyen and Elena Paslaru Bontas. Semantic Web Spaces (Technical Report, B 04-11). 2004. http://www.inf.fu-berlin.de/inst/pubs/tr-b-04-11.abstract.html
15. F. Martin-Recuerda and B. Sapkota, D21.v0.1 WSMX Triple-Space Computing, 2005, http://www.wsmo.org/TR/d25/d25.1/v0.1/
16. Andrea Calì, Giuseppe De Giacomo, Maurizio Lenzerini: Models for information integration: turning local-as-view into global-as-view in Proc. of the Workshop on Foundation of Models for Information Integration (FMII 2001): 10th Workshop in the series "Fondation of Languages, Data, and Object", LNCS, Springer.
17. S.S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J.D. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In Proceedings of the 10th Meeting of the Information Processing Society of Japan, pages 7-18, October 1994.
18. M. Templeton, H.Henley, E.Maros, and D.J. Van Buer. InterViso: Dealing with the complexity of federated database access. The VLDB Journal, 4(2):287-317, 1995.
19. M.T. Roth and P. Schwarz. Don't scrap it, wrap it! A wrapper architecture for data sources. In Proc. VLDB'97, pages 266- 275, Athens, Greece, 1997.
20. A. Levy, A. Rajamaran, and J.Ordille, Querying heterogeneous information sources using source description. In Proc. VLDB'96, pages 252-262, 1996.
21. I. Manolescu, D. Florescu, and D. Kossmann. Answering XML queries on heterogeneous data sources. In Proc. VLDB'01, pages 241-250, 2001.
22. Jeffrey D. Ullman. Information integration using logical views. In Proc. of the 6th Int. Conference on Database Theory (ICDT 97), volume 1186 of Lecture Notes in Computer Science, pages 19-40. Springer Verlag, 1997.
23. S. Abiteboul and O. Duschka, "Complexity of Answering Queries Using Materialized Views" In Proc. 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 98), pp. 254-265, 1998.
24. P.J. McBrien and A. Poulovassilis. Schema Evolution in Heterogeneous Database Architectures, A Schema Transformation Approach. In Proceedings of CAiSE02, Springer Verlag LNCS, Volume 2348, Pages 484-499, 2002
25. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini A framework for ontology integration. In Isabel Cruz, Stefan Decker, Jérome Euzenat, and Deborah McGuinness, editors, The Emerging Semantic Web _ Selected Papers from the First Semantic Web Working Symposium, pages 201-214. IOS Press, 2002.
26. L. Zamboulis and A. Poulovassilis. Information Sharing for the Semantic Web - a Schema Transformation Approach (Tech Report). 2005. http://www.doc.ic.ac.uk/automed/techreports/information_sharing_for_semantic_web.pdf
27. L. Predoiu, F. Martín-Recuerda, C. Feier, A. Polleres, F. Porto, A. Mocan, K. Zimmermann, and J. de Bruijn. Deliverable D1.5, DIP, 2004.
28. M.J. Eppler and R.A. Burkard. Knowledge Visualization. Towards a New Discipline and its Fields of Application, ICA Working Paper #2/2004, University of Lugano, Lugano.2004

29. Lamping, R. Ramana Rao and P. Pirolli, A Focus+context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. Conference on Human Factors in Computing Systems archive. In Proceedings of the SIGCHI conference on Human factors in computing systems. Denver, Colorado, United States. 401 - 408. 1995. ISBN: 0-201-84705-1

30. Ehud Reiter and Robert Dale. Building Natural Language Generation Systems. Cambridge University Press, Cambridge, England, 2000.

31. K. Bontcheva D5.1.1 Natural Language Generation for Knowledge Access. 2004

32. K. Bontcheva and Y. Wilks. Automatic Report Generation from Ontologies: the MIAKT approach. In Nineth International Conference on Applications of Natural Language to Information Systems (NLDB'2004), 2004.

33. G.Wilcock. Talking OWLs: Towards an Ontology Verbalizer. In Human LanguageTechnology for the SemanticWeb andWeb Services, ISWC'03, pages 109–112, Sanibel Island, Florida, 2003.

34. G. Wilcock and K. Jokinen. Generating Responses and Explanations from RDF/XML and DAML+OIL. In Knowledge and Reasoning in Practical Dialogue Systems, IJCAI-2003, pages 58–63, Acapulco, 2003.

35. K. Bontcheva. Generating Tailored Textual Summaries from Ontologies. Second European Semantic Web Conference (ESWC'2005). Crete. 2005.

36. R. T. Fielding. Architectural styles and the design of network-based software architectures. PhD Thesis, University of California, Irvine, 2000.

37. Rohit Khare and Richard N. Taylor. "Extending the Representational State Transfer (REST) Architectural Style for Decentralized Systems." Proceedings of the International Conference on Software Engineering (ICSE), May, 2004, Edinburgh, Scotland.

38. P. R. Pietzuch. "Hermes: A Scalable Event-Based Middleware". Ph.D. Thesis, Computer Laboratory, Queens' College, University of Cambridge, February 2004.

39. M. Bawa, B. F. Cooper, A. Crespo, N. Daswani, P. Ganesan, H. Garcia-Molina, S. Kamvar, S. Marti, M. Schlosser, Q. Sun, P. Vinograd, B. Yang. Peer-to-Peer Research at Stanford, The Peers Group. In SIGMOD Record, September 2003.

40. S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: the OceanStore Prototype. Appears in Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03), March 2003

41. K. Aberer, P. Cudré-Mauroux, A.Datta, Z. Despotovic, M. Hauswirth, Ma. Punceva, R. Schmidt, P-Grid: A Self-organizing Structured P2P System SIGMOD Record, 32(2), September 2003.

42. K. Aberer, A. Datta, M. Hauswirth. A decentralized public key infrastructure for customer-to-customer e-commerce, to be published in International Journal of Business Process Integration and Management, 1(1), 2005.

43. B. Cooper, M. Bawa, N. Daswani, and H. Garcia-Molina. Protecting the PIPE from malicious peers. In FGCS 2004.

44. P. Sepandar D. Kamvar, Mario T. Schlosser and Hector Garcia-Molina.The EigenTrust Algorithm for Reputation Management in P2P Networks, In WWW, 2003.

45. J. de Bruijn and A. Polleres. Towards and ontology mapping language for the semantic web. Technical Report DERI-2004-06-30, DERI, June 2004.

46. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In Proc. of the Twelfth International World Wide Web Conference (WWW 2003), pages 48-57. ACM, 2003.

47. D Gelernter, N Carriero. Generative Communication in Linda. ACM Transactions on Programming Languages and Systems, 1985

48. H. Li and G. Jiang (2004) Semantic message oriented middleware for publish/subscribe networks. Proceedings of the SPIE, Volume 5403, pp. 124-133 (2004).

49. K. Bontcheva, A. Kiryakov, H. Cunningham, B. Popov, and M. Dimitrov. Semantic web enabled, open source language technology. In EACL workshop on Language Technology and the Semantic Web: NLP and XML, Budapest, Hungary, 2003.

50. P. Bouquet, J. Euzenat, E. Franconi, L. Serafini, G. Stamou and S. Tessaris. D2.2.1 Specification of a common framework for characterizing alignment. Knowledge Web Technical Report. 2004.

51. G. Wiederhold. Mediators in the Architecture of Future Information Systems, IEEE Computer, 25(3): 38-49, 1992.

52. S. Graham and P.Niblett (editors). Web Services Notification (WS-Notification). Technical Specification. 2004. http://www-106.ibm.com/developerworks/library/specification/ws-notification/

53. A. Geller (editor). Web Services Eventing (WS-Eventing). Technical Specification. August 2004. http://www-106.ibm.com/developerworks/webservices/library/specification/ws-eventing/

54. D. Roman, H. Lausen, U. Keller, D2v1.2. Web Service Modeling Ontology (WSMO), 2005, http://www.wsmo.org/TR/d2/v1.2/

55. M.C. Mozer. An intelligent environment must be adaptive . IEEE Intelligent Systems and their Applications, No. 2, Vol. 14, 1999, pp 11-3.

56. M.H. Coen. Building Brains for Rooms: Designing Distributed Software Agents. In Proceedings of AAAI/IAAI, 1997, pp 971-7.

57. B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer, EasyLiving: Technologies for Intelligent Environments, Handheld and Ubiquitous Computing, September 2000

58. A. Fox, B. Johanson, P. Hanrahan and T. Winograd. Integrating Information Appliances into an Interactive Workspace. IEEE Computer Graphics & Applications, No. 3, Vol. 20, 2000.

59. Johanson, B. and A. Fox. The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In Proceedings of the 4th IEEE Workshop on Mobile Computer Systems and Applications (WMCSA-2002). 2002. Callicoon, New York, USA.

60. P. Wyckoff, S. W. McLaughry, T. J. Lehman and D. A. Ford. TSpaces. IBM Systems Journal 37(3), 1998. Also available at http://www.almaden.ibm.com/cs/TSpaces

61. E. Tsui. Technologies for Personal and Peer-to-Peer (P2P) Knowledge Management. 2002. http://www.csc.com/aboutus/lef/mds67_off/uploads/P2P_KM.pdf