Modeling Contexts in Collaborative Environment: A New Approach

Guiling Wang, Jinlei Jiang, and Meilin Shi

Dept. of Computer Science and Technology, Tsinghua University,
Beijing, 100084, P.R. China
{wgling, jjlei, shi}@csnet4.cs.tsinghua.edu.cn

Abstract. Context awareness, context sharing and context processing are key requirements for the future CSCW, HCI and Ubiquitous computing systems. However, research issues of collaborative context have not been completely addressed till now. While arguing that a generic context model is very important for building context-aware collaborative applications, this paper proposes a new semantic rich context modeling approach, Ontology for Contextual Collaborative Applications (OCCA), for collaborative environments. Based on OCCA, mechanisms for context query, context matching and collaboration awareness control are devised using semantic query and reasoning technology to support the three perspectives of a context model, i.e., information space, interaction space and collaboration control. We present an evaluation study on the features and performance of OCCA and context query services.

1 Introduction

Context and context awareness have been hot topics in recent years, especially in three communities: CSCW, HCI and Ubiquitous Computing. As a result, several context models have been proposed and several context-aware applications have been applied into ubiquitous computing environment, collaborative environment and human-computer interaction environment [1-9]. The new trend of context and context awareness research is to develop context services middleware as an infrastructure for distributed, heterogeneous and autonomous environments.

To build such an infrastructure, in our opinion, two levels are involved: at the conceptual level, a conceptual context model should be given to categorize the concepts and the relationship between them and to define the functions of application, and at the architectural level, a common and consistent architectural model should be given to define the hierarchy, the modules and their interactions. Services provided for context-aware applications include Provider & Consumer Services, Directory Services, Context Query/Event Services, Aggregation/Composition Services, Information Memory Services, Reasoning Services and Control Services. In this view, some of the most important projects on context-aware applications in distribution environment are summarized in Table 1.

Context awareness in CSCW is different from that in ubiquitous computing. Specific context factors are often taken into account by ubiquitous computing like location, time and people while the collaborative context like group, role and process, which is very important to CSCW applications, is not considered in ubiquitous computing environment. In collaborative environment, there is lack of a generic mechanism to model the context functions and concepts and to implement common services and architecture. Among the emerging models (Table 1), ENI (Event and Notification Infrastructure) [10] is the only CSCW-specific model which is different from other ubiquitous computing-specific models, but it is an incomplete work because most of the common services are not implemented. In our view, the characteristics of collaborative context should be taken into account from an integrated view. It is our aim to develop a CSCW-specific, complete and generic context model. We propose a generic model of context that focuses on modeling and designing context within collaborative environment.

This paper initially analyzes the functions of a context model in a collaborative environment. Next, a conceptual context model is proposed. And then the implementation of context services is explained. In the last section the paper is summarized and the future work is outlined.

Table 1. A comparison of context modeling

	Conceptual model	Architecture						
		Sensor/ Consumer	Directory Service	Context Query/ Event Service	Aggregation /Composition Service	Information Memory Service	Reasoning Service	Control Service
Context Toolkit [1]	Key-Value Model	Context widgets/ Actuator services	Discovers	No	Aggregators	No	Interpreters	No
Context Fabric [8]	Markup Scheme Model	Sensor/ application, context spec language	Sensor management service	Context event & query service	Automatic Path Creation service	No	No	Privacy
Gaia [9]	Ontology based Model	Context provider/ Consumer	context provider lookup service	Implemented by context consumer	context synthesizer	Context History	Ontology server	No
CoBrA [2]	Ontology based model	Context Acquisition Component/ Agent	Broker directory service	Implemented by Broker Behavior	Broker Behavior	Knowledge Base	Inference Engine	Privacy Policy
ENI [10]	Markup scheme model	Sensor/ Indicator	No	ENI	Situation module	Context module	No	No

2 The Functions of Context Model

In our view, context can be defined as any information that can be used to characterize the situation of entities in the collaborative space. On the one hand, the traditional groupware systems didn't model context explicitly. Context information is embedded in function modules during system development. In this way, the context services are fixed and hard to be changed. The system lacks a special context module, which results in difficulty with context services reuse. On the other hand, an integrated collaborative space can promote linking, navigation and querying of resources before, after, and while a collaborative action occurs. So it is very important to model the context for a collaborative environment.

First of all, every entity has its context in a collaborative space. For example, a person at work wants to know who are his (or her) collaborators as well as their profiles and present statuses. This information promotes cooperating among people. These contexts form the information space dimension of the context model for a collaborative environment, with which users can query any information, including historical and real-time information. Secondly, in an interaction space, various collaborative tools co-exist, users need different interaction patterns in different contexts. Based on the context model, system can help users switch to the proper groupware, fetching the collaborative documents on demand. Lastly, the collaboration control mechanism based on this context model can be more flexible and intelligent. For example, a policy of access control based on the context of a person is much richer in semantics than that based on role.

3 The Conceptual Model and OCCA Ontology

The Denver Model for Groupware Design [12] is a useful model describing the generic elements of any groupware application. Based on the Denver model, Rosa et al proposed a conceptual model for context-aware groupware [4]. However the Denver model doesn't take into account the element of collaboration tool. Various tools play different roles in a collaborative environment. The Denver model lacks the capability of modeling the tool element because its aim is for independent groupware rather than an integrated collaborative environment.

In order to make up for this lack of tool element of the Denver model, we propose a conceptual model that classifies contextual information into 8 categories: Person Context; Task Context; Process Context; Artifact Context; Tool Context; Environment Context; Collaboration Control Policy Context; and Historical Context. We define an Ontology for Contextual Collaborative Applications (OCCA) to present this conceptual model.

$$OCCA = \{Per, Tsk, Process, Art, Tool, Env, Pol, His\}$$

OCCA is used for description of human, task, process, artifact, tools, environment, policies, and the history of these entities. OCCA is written in OWL [13] and maintained by Protégé 3.0[14]. *Per, Tsk, Process, Art, Tool, Env,* and *Pol* are the names in XML namespace, and *His* is the record composed of other context information's history. Figure 1 shows OCCA in upper layer. The classes in upper layer model the generic concepts in collaborative environment such as *Person, Group, Role, Task, Process, Artifact, WorkSpace, Policy* and so on. Most of the classes in lower layer are sub-classes of the classes in Figure 1. They model the domain relevant concepts for applications in diverse fields. For instance, concepts like *Professor, Lecturer*, etc., in cooperative learning and *Driver, Passenger,* etc., in cooperative design are described. Due to space limit, the classes and properties in lower layer are not presented in this figure and are not discussed in this paper.

For modeling person context, OCCA defines such personal information as name, email, homepage, interests, identification in some chat tools, the related project, belonging group and so on. Elements like per: membership, per: Project, per: hasRole, per: Role and per: Group are the basic concepts or properties in collaborative applications. Persons in a common group share their contexts in the workspace.

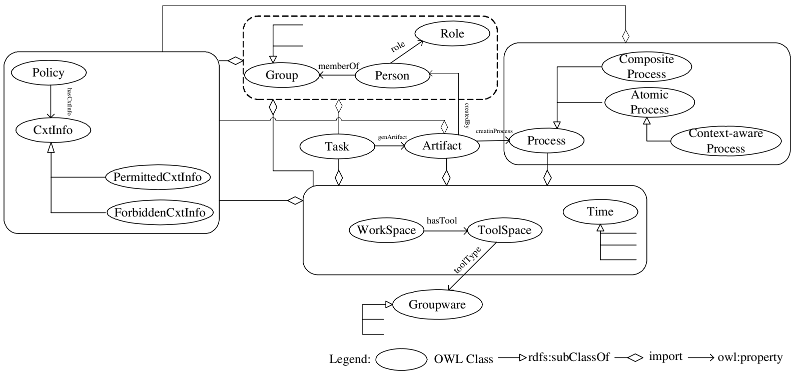

Fig. 1. OCCA in upper layer

Task is one of the core concepts in OCCA. OCCA defines vocabulary for describing task properties, restrictions, and plan. This description is used for task contextual information querying and reasoning. The basic properties of a task in OCCA include the name of the task (tsk: name), the description of the task (tsk: desp), the deadline of the task (tsk: deadline), the group which is in charge of the task (tsk: groupInCharge), the artifacts generated by the task (tsk: genArtifact), the attendee of the task (tsk: attendee), and the workspace where users do the task (tsk: inWorkspace). Such information is useful for users to query task information during their collaboration.

Process is another core concept in OCCA. Process:AtomicProcess is subclass of process:Process, which represents an action taking place during task run. Process: CompositeProcess is a subclass of process:Process, which represents a composite process composed of several activities. Process: ContextawareProcess is also a subclass of process: Process, and it has some special properties to generate the query request for context information in runtime. The details will be discussed in another paper.

The class OCCA: Artifact represents a set of artifacts. Artifacts are those objects produced or consumed during an interaction. Individuals of this class can have a set of properties like art: name, art: fileSize, art: fileType, art: createdBy, art: createdOn and art: modifiedOn to characterize its identification, size, type, creator, create time and modified time. It also has properties like art: createdIn and art: lastmodifiedBy with their values in the process: Process and per: Person respectively.

When building a contextual collaborative environment, it is very useful to describe kinds of collaboration tools, their characteristics, and their status. The class tool: Groupware is used to describe all the meta-information about collaboration tools in the environment. It has properties like tool: workTimeType, tool: workLocationType, tool:groupSize and tool:participant. Their domain is tool: Groupware and their range is tool: TimeOrLocationType, which is an enumerated class composed of value "common", "predictable" and "unpredictable". For example, the tool: workTimeType property and tool: workLocationType property of a teleconference system are both "common" because such a system only supports users who work at the same time (synchronous) and at the same place. Tool:groupSize has a data type of integer. The range of tool:participant is per:Person, which describes the constraints on person using this tool. Thus the properties are used to select a proper collaboration tool and transfer the document into the tool automatically.

OCCA ontology defines a vocabulary for representing policy of collaboration awareness. In a collaborative environment, the policy of collaboration awareness is defined by users and is used to allow or forbid the awareness information access or presentation to other users. Classes of pol: Policy and pol: CxtInfo are defined to represent the policy and the information. Properties like pol: fromGroup/pol: toGroup, pol: fromPerson/pol:toPerson, pol: fromRole/pol: toRole, pol: fromWorkSpace/to: toWorkSpace, pol: fromToolSpace/pol: toToolSpace, pol: fromProcess/pol: toProcess, pol: Artifact and pol: fromTime/pol: toTime are defined to represent the condition of who, where, and when the publishers or receivers are.

Environment contexts include information which can influence the users' actions like the location, workspace, time and so on. This information can be described based on OCCA. The agent can reason about the environment context to trigger some services.

When a task or activity is completed, personal profiles are changed or an artifact is modified, Information about them is stored for reference and tracking purpose. This is called historical context which is very important for a collaborative environment. Sometimes it is called the collaborative memory of a system.

4 Context Services

Previously we have built up an infrastructure for distributed, cross-organization and collaborative environment, called LAGrid [11]. Although we have also developed several groupware like the whiteboard, video-conference system and workflow management system, they haven't been integrated flexibly in a grid infrastructure. To support such a flexible integrated collaborative environment, we added context-aware support to current infrastructure by exploiting Contextual Collaborative Space (CCS). CCS is a new extension of LAGrid service-oriented middleware implemented for OCCA.

In the current version of CCS, the collaborative applications at the same time are context sensors/providers. The applications generate the ontology instances and store them in the knowledge base in real time. The ontology introduced above is also stored in the knowledge base. On top of the knowledge base, we use a reasoner which can derive new knowledge from the knowledge base. The reasoner supports OWL-DL inference and $\mathcal{AL}$-log inference which is a hybrid integration of ontology reasoning and rule based reasoning. In order to implement information space, interaction space and collaboration control mechanisms for the contextual collaborative space (CCS), we design new services like Context Query & Memory Service, Inference Service and Control Service based on the knowledge base and the reasoner.

4.1 Context Query and Memory

Information space is divided into 3 parts: group information space, private information space and historical information space. During the collaborative work, individuals of every class in OCCA are declared in collaborative information space by context sensor services. Context is visualized as a graph so that users can navigate all the elements in the environment together with their associate attributes and the relationships between them.

Every entity in collaborative environment has its context. The context is dynamic with some properties having different values at each collaborative situation. For storing, retrieving and aggregating dynamic context information, RDF dataset and named graph model defined in [15] is used.

Context Memories can be seen as RDF datasets composed of Person Context Memory, Task Context Memory, Artifact Memory, etc. Each context memory (CM_i) is a RDF dataset as follows:

$CM_i = \{Cxt, (<u_1>, Cxt_1), (<u_2>, Cxt_2), \ldots (<u_n>, Cxt_n)\}$, where *Cxt* is the aggregate graph, and each $<u_i>$ is a distinct URI. $(<u_i>, Cxt_i)$ is a named graph. Cxt_i, which is described in *Cxt*, is a set of facts and the situation within which those facts are believed to be true. When a new collaborative entity is created or changed, a graph *Cxt* is created and stored into *CM* together with the situation.

Track of document or task can be implemented easily based on this model. An example of a track is as the following: "query the word documents modified during Oct.2005 by Alice, return the documents' name, creator and the task name within which the document was modified". Table 2 shows the query expression in SPARQL that supports RDF dataset. For context query and context memory, Jena [16], which supports SPARQL language, is used for RDF document parsing and query.

Table 2. A context query expression

```
SELECT  ?name ?creatorname ?taskname
WHERE{
      ?g  inTask  ?task .
      ?task hasname ?taskname .
      GRAPH ?g
       {?doc  rdf:type art: WordDocument;
             art:modifiedOn ?date;
             art:modifiedBy ?mPerson;
            art:name ?name;
            art:creatBy ?creator .
        ?creator per:name ?creatorname .
        ?mPerson per:name "Alice".
        FILTER
        (?date > "2005-09-30"^^xsd:date
          && ?date < "2005-11-01"^^xsd:date)
       }
     }
```

4.2 Context Service Matching

In a contextual collaborative space, the metadata of the context services are published in a central server for further discovery. We advertise the service descriptions by sending the message to a so-called "information service". The information service assigns it a unique ID, stores it in the repository and sends it to the inference service to be added to the subsumption hierarchy. When a collaborative action is planned, a request is submitted to the information service. The inference service computes the match degree between the request and each advertisement in the repository. The most matchable one is returned with its unique ID. Then the requester can query more detailed characteristics of the tool in the information service using this unique ID.

For example, in the interaction space, the characteristics of the collaboration tools like whiteboard, video-conferencing, threaded discussions and so on are encapsulated as web services. We want to publish a whiteboard to the information service with some restrictions on the participants' size and location like this: i). Group's size must be less than 5. ii) Participants must be from some city. So the advertisement can be written as presented in Table 3 in description logic (DL) notation. It is submitted to information service and WBAdvert is matched and returned with the subsumption relationship of $\mathit{Query} \sqsubseteq \mathit{WBAdvert}$. The implementation is based on Pellet used in conjunction with Jena [17].

Table 3. An advertisement and query of the whiteboard

WBAdvert = ServiceProfile ⊓ ∀item (∀ type.WB ⊓ ∀workTimeType.common ⊓ ∀workLocationType.common ⊓ < 5 groupSize ⊓ ∀partici- pant.(Person⊓∀location.City)))	Query = ServiceProfile ⊓ ∀item(∀type.WB ⊓ ∀workTimeType.common ⊓ ∀workLocationType.common ⊓ = 3 groupSize ⊓ ∀partici- pant.(Person⊓∀location.Beijing))

4.3 Collaboration Awareness Control Policies

Policies are increasingly used for behavior control of complex systems, allowing administrators to modify system behavior without changing source code. Semantic-rich policy representations based on a common ontology can facilitate interoperation, and the policy representation based on description logic can simplify policy analysis and conflict detection.

Based on OCCA, the publisher and receiver of the shared collaboration awareness information can define control policy in a declarative way. With whom the awareness information is shared is determined both by the profile, location, task, related artifact and time period of the publisher and by those of the receiver.

The cascading characteristic of context is indicated as "the collaboration spaces associated with broader contexts are also visible within an inner context" [18]. Contexts are nested following the structure of task, activities, role, workspace and organization. For example, G1 represents an organization and G1.1 represents a department of this organization in Figure 3. Thus the collaborative awareness information generated from G1.1 can be seen as generated from G1 and the information which will be sent to G1 will also be sent to G1.1. The group information view is the aggregation of the nested context and others. It can save the storage cost and processing cost by inheriting the outer context.

Describing the cascading characteristics of the collaborative awareness information goes beyond the expressive capabilities of OWL DL. So we take advantage of the expressive power of rules to depict it. In this paper, we use $\mathcal{AL}$-log [19], a hybrid approach combining ontology language and rule language. The example above can be described in $\mathcal{AL}$-log as follows:

```
toGroup(C, G11) :- toGroup(C, G1) , partOf(G11, G1) &
                     C:CxtInfo , G1:Group , G11:Group
fromGroup(C, G1):- fromGroup(C, G11) , partOf(G11, G1) &
                     C:CxtInfo , G1:Group , G11:Group
```

More powerful cascading characteristics can be described based on the relation between different properties:

```
toPerson(C, P):- toGroup(C, G) , memberOf(P, G)&
                     C:CxtInfo , P:Person , G:Group
```

In the following rule, pol represents a control policy on collaborative awareness information sharing. It is specified by John saying that the context information could be shared with a group at meeting room if and only if the publisher is doing some action on some word documents of CSCW topic.

```
permits(pol,cxt):-
policyOf(pol,p) , name(p,John) , toGroup(cxt,mg) , fromPerson(cxt,sender)
& pol:Policy , p:Person , cxt:CxtInfo , mg:MeetingRoomGroup , sender:Person
⊓ ∃AttendIn.CSCWRelatedTask
```

The concepts in this rule which have not been introduced in Section 3 are defined as the following:

```
CSCWRelatedTask = Task ⊓ ∃GenArtifacts.CSCWWordArtifact;

CSCWWordArtifact = WordArtifact ⊓ ∃Topic.{CSCW}; WordArtifact ⊑ Artifact
```

5 Evaluation

Compared with other most important context-aware models in Table 1, OCCA is a context model especially for collaboration applications. OCCA is designed towards a distributed, heterogeneous context-aware collaboration environment. It is ontology based and adopts the standard OWL as its description language. The architecture model based on OCCA has been introduced in section 4. CCS adopts the service-oriented architecture and has developed some key services. Also Table 4 presents another feature of OCCA in our collaborative environment. Which context information is used in implementing the three kinds of functions is showed in this table. Observed that the most frequently used context information are description of human, task, artifact and tool, we can say that OCCA has the feature of focusing more on internal context instead of outer context.

We evaluated the context query performance against the artificial RDF named graph dataset which is generated according to the real world collaboration scenario. The test ontology defines 48 classes and 35 properties for description of a collaborative learning group, the task and the artifact. We created 8 datasets with different sizes of class instances and properties. The size of the datasets is from 600KB to 9MB. The test was done on a windows 2003 server with the configuration of AMD Athlon(tm) XP 1600+ at 1.40GHz and 768MB RAM. The java environment is J2SE 1.5.0_04 and the max java heap size is 512MB. The test module is based on Jena 2.4 ontology toolkit with an Oracle 9i database as the back-end. The Oracle9i database server runs on a VMware workstation with windows 2003 server OS, AMD Sempton™ Processor 2800+ at 1.61GHz and 740MB RAM. All of the ontology data is loaded in database in advance so that they needn't be re-loaded while performing query. We performed a SPARQL query on Named Graphs both with RDFS reasoner and OWL reasoner 5 times on each dataset. We measured the average query time as shown in Figure 2. The results show that the query time is within 25 seconds for the knowledge base of about 1000 class instances (about 9MB) in our prototype when only RDFS inference is needed. But for the complex query that need OWL inference, the query time can be 8 minutes long. The observation suggests us cache the query results for non-real-time tasks. Also we can split the instances and classes or load the data in common use in the main memory in order to reduce the query response time.

Table 4. Ontology used in different scenario

	Per	Tsk	Proc	Art	Tool	Env	Pol	His
Context Query	√	√	√	√	√	√	√	√
Collaborative Tools Switching	√	√	×	×	√	×	×	×
Collaboration Awareness Control	√	×	×	√	×	×	×	×

√used; ×not used.

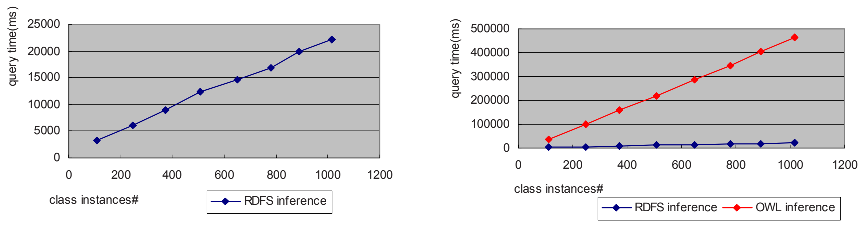

Fig. 2. Context query time

6 Conclusions and Future Work

The context model described in this paper is the first attempt to give a generic semantic rich model for integrated context-aware collaborative environments. This model focuses on context information sharing and collaborative work rather than an individual's context awareness and reasoning.

OCCA is used for description of 8 kinds of generic entities for collaborative applications and the domain relevant concepts for collaborative applications in diverse fields. Its query and inference mechanism is based on OWL and $\mathcal{AL}$-log. This paper presents the context services devised for the architecture. Also we evaluate the features of OCCA and the performance of the context services. There are still several issues to be resolved. The $\mathcal{AL}$-log reasoner that can work with Jena is still in progress and needs further development and performance evaluation for collaboration control.

Acknowledgments. This work is supported by the National Natural Science Foundation of China under Grant No. 90412009.

References

1. Dey, A.K. and Abowd, G.D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. HCI 16(2001) 97-166
2. Chen, H.: An Intelligent Broker Architecture for Context-Aware Systems. PhD.Dissertation proposal. University of Maryland (2003)
3. Brezillon, P., Borges, M., Pino, J. and Pomerol, J.-C.: Context-Awareness in Group Work: Three Case Studies. IFIP International Conference on Decision Support Systems (2004)
4. Rosa, M.G.P., Borges, M.R.S. and Santoro, F.M.: A Conceptual Framework for Analyzing the Use of Context in Groupware. The 9th International Workshop on Groupware: Design, Implementation, and Use. LNCS 2806 (2003) 300-313
5. Steinfield, C., Jang, C.-Y., .Pfaff, B.: Supporting Virtual Team in Collaboration: The TeamSCOPE System. Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (1999) 81-90
6. Bradley, N.A., Dunlop, M.D.: Towards a User-centric and Multidisciplinary Framework for Designing Context-aware Applications. First International Workshop on Advanced Context Modeling, Reasoning And Management (2004)
7. Hong, J.I.: Context Fabric: Infrastructure Support for Context-Aware Systems. Qualifying Exam Proposal (2001). URL: http://www.cs.berkeley.edu/~jasonh/quals/quals-proposal-context-fabric.pdf
8. Ranganathan, A. and Campbell, R.H.: A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. ACM/IFIP/USENIX International Middleware Conference (2003)
9. Gu, T., Pung, H.K., Zhang D.Q.: A Service-Oriented Middleware for Building Context-Aware Services. Journal of Network and Computer Applications 28(1) (2005) 1-18
10. Gross, T., Prinz, W.: Modelling Shared Contexts in Cooperative Environments: Concept, Implementation, and Evaluation. Computer Supported Cooperative Work: The Journal of Collaborative Computing 13(3-4) (2004) 13-34
11. Wang, G., Li, Y., Yang, S., Miao, C., Xu Jun, Shi, M.: Service-oriented grid architecture and middleware technologies for collaborative e-learning. IEEE International Conference on Service Computing (SCC2005) 67-74
12. Salvador, T., Scholtz, J., Larson, J.: The Denver model for groupware design. ACM SIGCHI Bulletin archive 28(1)(1996)
13. McGuinness D.L., van Harmelen, F.: OWL web ontology language overview (2004). URL: http://www.w3.org/TR/2004/REC-owl-features-20040210/
14. The Protege Ontology Editor and Knowledge Acquisition System (2006). URL: http://protege.stanford.edu/
15. SPARQL Query Language for RDF (2006). URL: http://www.w3.org/TR/rdf-sparql-query/
16. Jena – a semantic web framework for Java (2006). URL: http://jena.sourceforge.net/
17. Pellet OWL Reasoner (2006). URL: http://www.mindswap.org/2003/pellet/
18. Lei, H., Chakraborty, D., Chang, H., Dikun, M.J., Heath, T., Li, J.S., et al.: Contextual Collaboration: Platform and Applications. IEEE International Conference on Services Computing (SCC 2004). Shanghai, China (2004)
19. Donini, F.M., Lenzerini, M. et al.: AL-log: Integrating Datalog and Description Logics. Journal of Intelligent Information Systems 10(3) (1998) 227-252