2015

Collaborative knowledge as a service applied to the disaster management domain

Katarina Grolinger
Western University, kgroling@uwo.ca

Emna Mezghani

Miriam AM Capretz
Western University

Ernesto Exposito

Collaborative knowledge as a service applied to the disaster management domain

Katarina Grolinger*

Department of Electrical and Computer Engineering
Faculty of Engineering
Western University,
London, ON, Canada N6A 5B9
E-mail: kgroling@uwo.ca
*Corresponding author

Emna Mezghani

CNRS; LAAS
7 av. du Colonel Roche
F-31400 Toulouse, FRANCE
Université de Toulouse;
UPS, INSA, INP, ISAE; LAAS
F-31400 Toulouse, France
E-mail: emezghan@laas.fr

Miriam A.M. Capretz

Department of Electrical and Computer Engineering
Faculty of Engineering
Western University,
London, ON, Canada N6A 5B9
E-mail: mcapretz@uwo.ca

Ernesto Exposito

CNRS; LAAS
7 av. du Colonel Roche
F-31400 Toulouse, FRANCE
Université de Toulouse;
UPS, INSA, INP, ISAE; LAAS
F-31400 Toulouse, France
E-mail: eexposit@laas.fr

Abstract—Cloud computing offers services which promise to meet continuously increasing computing demands by using a large number of networked resources. However, data heterogeneity remains a major hurdle for data interoperability and data integration. In this context, a Knowledge as a Service (KaaS) approach has been proposed with the aim of generating knowledge from heterogeneous data and making it available as a service. In this paper, a Collaborative Knowledge as a Service (CKaaS) architecture is proposed, with the objective of satisfying consumer knowledge needs by integrating disparate cloud knowledge through collaboration among distributed KaaS entities. The NIST cloud computing reference architecture is extended by adding a KaaS layer that integrates diverse sources of data stored in a cloud environment. CKaaS implementation is domain-specific; therefore, this paper presents its application to the disaster management domain. A use case demonstrates collaboration of knowledge providers and shows how CKaaS operates with simulation models.

Keywords: cloud computing; data interoperability; data integration; data heterogeneity; knowledge as a service (KaaS); NoSQL storage; disaster management

Biographical notes: Katarina Grolinger is currently a Postdoctoral Fellow at Western University, Canada. She received her Ph.D. and M. Eng. Degrees in Software Engineering from Western University. Previously, she obtained her M. Sc. and B. Sc. in Mechanical Engineering from the University of Zagreb, Croatia. She is also a Certified Oracle Database Administrator with over ten years of industry experience in database administration and software development. Her research interests include NoSQL data stores, cloud computing, Big Data management, disaster data management, data integration and interoperability.

Emna Mezghani is currently a Ph.D. student in Software Engineering at the University of Toulouse. Previously, she obtained her M. Sc. degree in Computer Science and her B.Sc. degree in Computer Engineering from the National School of Engineering of Sfax, Tunisia. Her research interests include service oriented architecture, collaborative systems, cloud computing, Big data, semantics and knowledge management.

Miriam A. M. Capretz is an Associate Professor in the Department of Electrical and Computer Engineering at Western University, Canada. Before joining Western University, she was an Assistant Professor in the Software Engineering Laboratory at the University of Aizu, Japan. Dr. Miriam Capretz received her B.Sc. and M.E.Sc. degrees from UNICAMP, Brazil and her Ph.D. from the University of Durham, UK. She has been involved with the organization of several workshops and symposia as well as has been serving on program committees in several international conferences. She was a Program Co-Chair of the IEEE Workshop Web2Touch – living experience through web (W2T) in 2008 and 2009 and was the Program Chair of the IEEE Symposium on Human and Socio-Cultural Service Oriented Computing 2009. She has been working in the software engineering area for more than 30 years. Her current research interests include cloud computing, Big Data Analytics, service oriented architecture, ontology and semantic integration, business process management, software security and privacy.

Ernesto Exposito is an Associate Professor at the INSA of Toulouse and a researcher at the LAAS laboratory of the CNRS, France. In 2004, he worked as

researcher in the National ICT Australia Limited (NICTA) research center in Sydney, Australia. In 2003, he earned his Ph.D. in “Informatique et Télécommunications” from the Institut National Polytechnique de Toulouse, France. In 2010, he earned his "Habilitation à diriger des recherches" from the Institut National Polytechnique de Toulouse. He has served as chair and member of program committee of many international conferences. He has contributed in several European and French research projects and currently he is the coordinator at LAAS of the IMAGINE European project related to Dynamic Manufacturing Networks. His research interests include designing, modeling and developing service-oriented, component-based and ontology-driven autonomic transport, middleware and cloud communication services.

1 Introduction

With the development of information and communication technologies (ICT), software and hardware capabilities have been evolving continuously, and therefore large quantities of heterogeneous data are being generated and stored. Due to the size, complexity, and diversity of these data, traditional computing systems are encountering challenges with handling, processing, storing, and analyzing them. Moreover, computing systems are facing ever-increasing and strict availability and scalability requirements.

In the domain of distributed and networked information systems, cloud computing promises to meet this demand by using large numbers of networked resources. The cloud computing approach is well known for providing on-demand computing services (Almorsy *et al.*, 2011). It aims to minimize user-perceived latency by providing advanced mechanisms for data storage, computation, and dynamic resource allocation according to real-time computation needs. Consequently, a large number of IT companies, including Amazon, Google, and Microsoft, are now providing cloud computing services. The National Institute of Standards and Technology (NIST) has defined a generic cloud computing reference architecture (Liu *et al.*, 2011). From the service perspective, this reference model includes three layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The IaaS layer contains the physical resources, such as servers, processors, and networks, on which the platforms will be deployed. The PaaS layer offers platforms for data storage, programming languages, and Web application servers. Finally, the SaaS layer handles software applications that directly access infrastructure resources or refer to the PaaS layer for their computing platform and for data access. Storing data in a cloud environment, more precisely in the PaaS layer, can provide the following benefits (Kossmann and Kraska, 2010):

- High availability. Within the cloud environment, data are automatically replicated, often across large geographic distances. If a local data centre fails, the system remains available because it can switch to another data centre.
- Scalability and elasticity. A cloud solution can adapt storage resources based on real-time needs and priorities. Data can be automatically redistributed to take advantage of heterogeneous servers.
- There is no need for a large initial investment. The system can start small and be expanded by adding heterogeneous nodes as needed.

Recent advances in cloud computing (Erl *et al.*, 2013), including big data (Gupta *et al.*, 2012), (Mohanty *et al.*, 2013), (Wigan and Clarke, 2013) and NoSQL (Sadalage and

Fowler 2013), (Stonebraker *et al.*, 2007), (Grolinger *et al.*, 2013a), (Brewer, 2012), (Hu and Qu, 2013) approaches, have been changing how data are captured, stored, and analyzed. They have been especially popular in Web applications (Sakr *et al.*, 2011), including Facebook, Twitter, and Google. NoSQL approaches (Hecht and Jablonski, 2011) are characterized by flexible data models, horizontal scalability, and excellent performance with simple read/write operations.

However, data heterogeneity remains a major hurdle for data integration, analysis, and decision-making. In this context, the Knowledge as a Service (KaaS) approach (Abdullah *et al.*, 2011) was proposed. KaaS provides a collection of lessons learned, best practices, and case studies that can help systems leverage knowledge from anywhere in a distributed computing environment. The main objective of KaaS is to generate knowledge from heterogeneous data located in a cloud environment and make it available as a knowledge service. In KaaS, knowledge is considered as an understanding of information based on its relevance to a problem area and is perceived as a precious resource essential for decision-making.

Consequently, this paper proposes Collaborative Knowledge as a Service (CKaaS), a generic architecture that integrates disparate cloud knowledge through collaboration among distributed KaaS entities with the goal of satisfying consumer knowledge needs. CKaaS has the following objectives:

1) storing large amounts of data from diverse sources,

2) supporting interoperability and integration, and

3) offering a scalable reconfigurable cloud solution for efficient resource consumption.

Based on the NIST reference architecture (Liu *et al.*, 2011), CKaaS is a distributed cloud architecture that offers collaboration capabilities and dynamic provisioning of resources. CKaaS delivers knowledge as a service while relying on semantic integration to facilitate search and interoperability. Storage of large amounts of heterogeneous data is achieved by using relational databases and NoSQL data stores in the cloud environment.

In this paper, the proposed CKaaS is applied to the disaster management domain. Disaster-related data are massive, heterogeneous and complex; they include response and mitigation plans, sensory information, simulation models exploring critical infrastructure behaviour, and disaster-related social media information. However, current data storage systems in disaster management are disparate and provide few if any integration capabilities. CKaaS represents a powerful collaborative system that enriches disaster knowledge management solutions by adding scalable storage and integration capabilities. Consequently, in the disaster management domain, CKaaS facilitates better decision-making by integrating distributed disaster-related information and providing knowledge as a service.

The remainder of the paper is organized as follows. In Section 2, the proposed CKaaS is portrayed, and its application in the disaster management domain is detailed in Section 3. A case study is described in Section 4, while Section 5 reviews work related to Knowledge as a Service and presents approaches to data and information management in the disaster management domain. Section 6 identifies CKaaS limitations, challenges, and opportunities, and conclusions and future work are presented in Section 7.

2 CKaaS architecture

This section proposes CKaaS, a generic architecture based on the NIST[1] reference model for knowledge integration through collaboration among distributed KaaS entities.

Cloud computing is well known for its ability to deliver highly scalable distributed computing platforms in which computational resources are offered as a service (Almorsy *et al.*, 2011). The CKaaS architecture has been designed on the basis of the NIST cloud computing reference architecture, which represents a generic cloud computing model (Liu *et al.*, 2011). Other reference architectures, such as those proposed by IBM[2] and CISCO[3], have enhanced the NIST model by proposing specializations in specific areas such as business and communication networks.

The proposed CKaaS architecture is illustrated in Figure 1. It is based on the NIST reference model with the intention of providing a collaborative cloud-based knowledge management solution. CKaaS incorporates KaaS as a new sub-layer on top of the standard PaaS layer of each *Cloud Provider* and enriches the basic *Cloud Broker* by adding the *KaaS Broker*. It was decided to locate KaaS on top of the PaaS layer because it offers a knowledge integration service by taking full advantage of the available PaaS-level resources and services. Likewise, by means of the PaaS, the KaaS layer is able to use virtual and physical network resources available in the IaaS layer to integrate distributed and collaborative knowledge sources.

The main purpose of this distributed cloud architecture is to facilitate collaboration among various KaaS *Cloud Providers* and to address the knowledge-incompleteness limitation that can be encountered with isolated *Cloud Providers*. The architecture is domain-independent, although its implementation is intended to be domain-dependent because it relies on ontologies in a specific knowledge domain. A domain-specific implementation of CKaaS is an instantiation of this distributed cloud architecture in which several *Cloud Providers* contain complementary domain-dependent knowledge. *Cloud Consumers* will be able to retrieve this knowledge by accessing one specific *Cloud Provider* directly or several cloud providers indirectly through the *Cloud Broker*.

The main advantages of this distributed cloud architecture in the CKaaS solution are:

- First, by interconnecting several KaaS entities, the platform can facilitate collaboration and orchestrate disparate knowledge sources (from various KaaS providers) through the intermediation service deployed by the broker.
- Second, KaaS services can be efficiently managed by the distributed cloud architecture to provide adequate QoS levels from two perspectives: intra-cloud and inter-cloud. Management strategies include continuous monitoring and dynamic configuration (provisioning/re-provisioning) of the resources and services offered by one or several *Cloud Providers*.

The CKaaS architecture, as shown in Figure 1, defines three actors: *Cloud Consumer*, *Cloud Broker*, and *Cloud Provider*.

---

[1] NIST: National Institute of Standards and Technology

[2] IBM Cloud Computing Reference Architecture
https://www.opengroup.org/cloudcomputing/uploads/40/23840/CCRA.IBMSubmission.02282011.doc

[3] CISCO Cloud Reference Architecture Framework
http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns836/ns976/white_paper_c11-617239.html

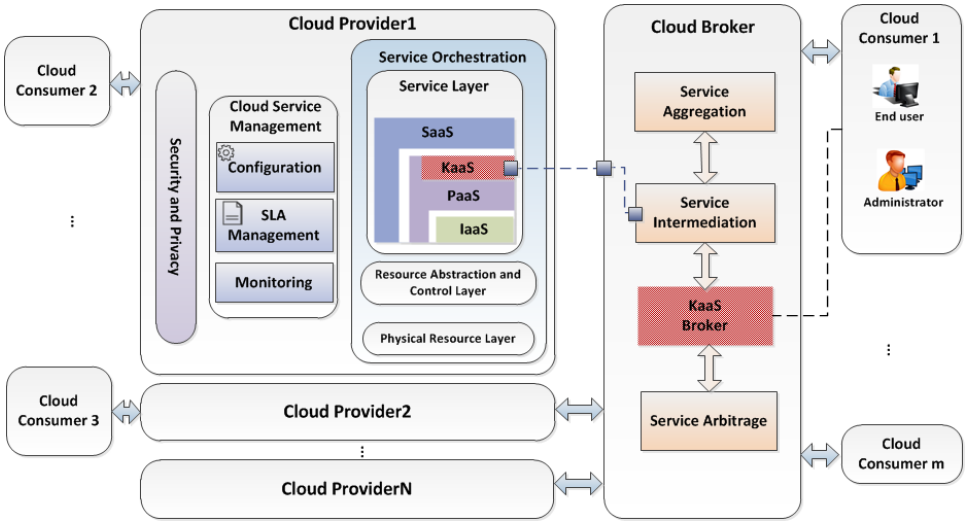

Figure 1. CKaaS architecture.

2.1 Cloud Consumer

In CKaaS, a *Cloud Consumer* is any entity that establishes and maintains a knowledge-based business process and consumes services offered by the KaaS *Cloud Provider* under specific service level agreements (SLA). These entities may represent human actors, such as end users or administrators, or logical resources such as software applications. As presented in Figure 1, *Cloud Consumers* can consume knowledge services directly from one *Cloud Provider* or indirectly through the *Cloud Broker*.

2.2 Cloud Broker

In the CKaaS solution, the *Cloud Broker* represents a mediator between the *Cloud Consumers* and the KaaS services of various *Cloud Providers*. This mediation is achieved by the *Cloud Broker* based on its interpretation and adaptation of the *Cloud Consumer's* requests to provide efficient access to the appropriate KaaS *Cloud Providers*. The *KaaS Broker*, which is part of the *Cloud Broker*, can also provide a knowledge-based cache that is built from previous requests/responses to improve the *Cloud Consumer's* quality of experience. At the time of the first request, this cache is empty and cannot respond to users' requests, but it dynamically learns from subsequent requests. Consequently, for new requests, the *KaaS Broker* refers to this cache to verify whether it can directly respond to these requests without forwarding the requests to the KaaS *Cloud Providers*. The *KaaS Broker* cache implements adequate maintenance strategies (i.e., well-known cache-design pattern strategies) to guarantee the temporal validity of the cached knowledge.

Basically, the broker is implemented based on three fundamental services: *service intermediation*, *service aggregation*, and *service arbitrage*. These services implement advanced policies to integrate distributed knowledge by enabling collaboration between disparate KaaS *Cloud Providers*. *Service intermediation* facilitates communication between *Cloud Consumers* and KaaS *Cloud Providers*. It enhances the *Cloud Broker* with an advanced API that adapts requests according to the appropriate provider characteristics (communication protocol, message format, data model, etc.) by

dynamically creating equivalent requests (e.g., SPARQL-based requests). If the requirement is not satisfied in the *KaaS Broker*, the service intermediation facility can replicate the adapted requests in parallel to all *Cloud Providers* to locate the appropriate KaaS source(s) which can provide the response. Two policies can be followed when implementing the service intermediation facility:

- Once the service intermediation facility receives a response, it cancels the incoming responses from the other KaaS *Cloud Providers*.
- The service intermediation facility waits for all responses and delegates them to the *service aggregation* facility, which combines the multiple responses into a final and integrated response.

Within these two policies, the *KaaS Broker* is dynamically learning about *Cloud Provider* performance and knowledge services based on the returned responses. The learned knowledge is stored in the cache.

The broker also implements the *service arbitrage* capability, which is the most advanced and intelligent service because it is based on learning strategies which enable the *KaaS Broker* to construct an overall view of the KaaS *Cloud Providers* and the performance of each. Based on this view, the *KaaS Broker* will be able to select the best KaaS *Cloud Provider* for future requests based on the *Cloud Consumer's* requests and QoS requirements.

2.3 Cloud Provider

Each *Cloud Provider* offers adequate facilities to guarantee service provisioning in a secure way. As previously described, in CKaaS, the service layer of the *Cloud Provider* has been extended by adding a KaaS sub-layer on top of the PaaS layer. The KaaS layer will use the services provided by the PaaS to access the required virtual or physical resources such as servers, processors, or networks that are offered as services and allocated accordingly by the IaaS layer. The PaaS layer contains platform services related to data storage, programming language, and integration solutions. Based on this general platform service, the KaaS layer will be able to transform the data stored within the PaaS into knowledge and offer them as services.

One important non-functional aspect that has been considered in the KaaS design is performance management when delivering knowledge services. This functionality is provided by the *Cloud Service Management* component as defined in the NIST architecture, which includes well-adapted components and strategies. In the proposed distributed cloud architecture, these components guarantee efficient delivery of knowledge services. *Cloud Service Management* provides three basic functionalities:

- Monitoring. It supervises and observes execution of cloud services and their impacts on resource consumption (e.g., CPU and memory usage).
- SLA management. This includes SLA and key performance indicator (KPI) monitoring and evaluation based on SLA contracts.
- Configuration. Based on monitoring and SLA evaluation, adequate reconfiguration strategies will be enforced to improve service performance by adjusting resource allocation or dynamically deploying new resources to satisfy increasing demands.

Security and privacy are outside the scope of this work; however, they are included in the CKaaS architecture presented in Figure 1 to underline their importance.

Based on the generic CKaaS architecture presented in this section, the next section will describe how this solution has been adapted to the disaster management domain. More precisely, the KaaS *Cloud Provider* and *KaaS Broker* architecture will be described.

3 CKaaS applied to the disaster management domain

The key element of the proposed Collaborative KaaS architecture is the KaaS added to the NIST cloud provider's service layer. Therefore, the *Cloud Provider* service layer as introduced in Figure 1 is further described in this section. Moreover, KaaS is detailed because it is the basis for providing knowledge to consumers and the core entity responsible for offering a collaborative cloud-based knowledge management solution.

KaaS internals are application-dependent; in this section, a Collaborative KaaS application in the disaster management domain is presented. The choice of the disaster management domain is motivated by a number of reasons, including the importance of knowledge in disaster decision-making, the potential to reduce the impact of disasters on human lives and property, the distributed nature of disaster-related knowledge, and the authors' previous experience with disaster data management. The heterogeneity of the data involved in disaster-related activities and its distributed nature are the main challenges in providing a comprehensive knowledge-management solution that could be used by various stakeholders in diverse disaster situations.

Figure 2 illustrates a *Cloud Provider* service layer in the disaster management domain. The four layers, IaaS, PaaS, SaaS, and KaaS, are included, with the elements belonging to each layer. IaaS is unchanged from the NIST layer and includes physical computing resources such as computers, storage, and networks. The PaaS layer consists of the NIST PaaS layer and the added KaaS. The traditional NIST layer provides a platform and includes entities such as operating systems, programming languages, Web frameworks, and databases. In Figure 2, the database element is further detailed to show that in the disaster management domain, the cloud storage can include both relational databases and non-relational, so-called NoSQL data stores (Sakr, 2013), including document, column, and graph stores. The second part of the PaaS layer is the KaaS layer which was added within PaaS, but on top of the traditional PaaS components. The KaaS layer uses the disaster cloud data management (Disaster-CDM) approach proposed by Grolinger *et al.* (2013b). Finally, the SaaS layer provides access to software and databases, and in the disaster management domain, contains services such as simple data access, administration, analytics, and model checking. The SaaS layer can access both the PaaS and KaaS layers according to application requirements.

The remainder of this section provides details of the KaaS service layer. The services provided by the KaaS can be classified as knowledge acquisition and knowledge delivery services, as shown in Figure 2. Knowledge acquisition is responsible for acquiring knowledge from diverse sources, processing it to add structure to unstructured or semi-structured data, and storing it in databases. The second part, knowledge delivery services, is responsible for integrating information from different data stores and delivering knowledge to consumers. It was decided to extract and store the knowledge because the time required for queries and information integration would be incompatible with the requirements of emergency response. This would enable shorter response time to queries than performing processing "on the fly".

The following two subsections describe the two main parts of the KaaS *Cloud Provider*: knowledge acquisition and knowledge delivery.

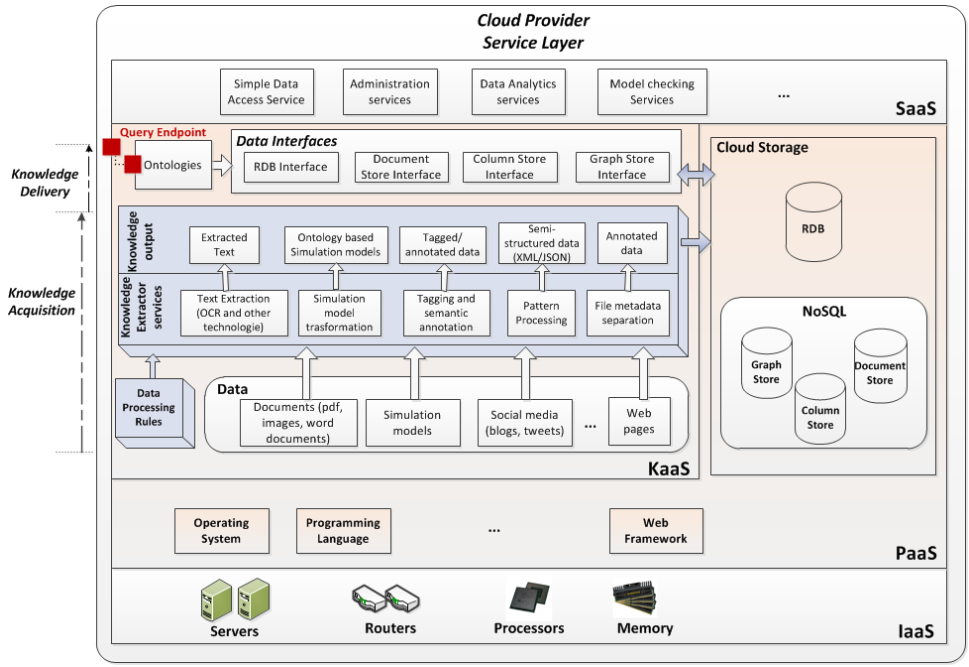

Figure 2: *Cloud Provider* Service Layer

3.1 Knowledge acquisition

The knowledge acquisition function obtains data from heterogeneous data sources, processes them to extract knowledge, and stores them in the cloud environment.

3.1.1 *Heterogeneous data sources*

A few examples of information related to disasters are disaster plans, incident reports, situation reports, social media, and simulation models, including infrastructure and health-care simulation. As for representation formats, examples include MS Word, PDF, XML, a variety of image formats (jpeg, png, tiff), and simulation package-specific model formats. Data representation is important because it determines the methods that can be used to add structure to unstructured or semi-structured data.

From the authors' experience working with local disaster-management agencies, the majority of information is stored in unformatted documents, primarily PDF and MS Word files. This agrees with the work of Hristidis *et al.* (2010), who reported that most information is in PDF and MS Word files.

3.1.2 *Knowledge extractor*

Because the input data are so diverse, the knowledge they contain cannot be extracted using a single approach. Therefore, processing is driven by the input data and by *data processing rules*, as illustrated in Figure 2. *Data processing rules* specify what processes are to be applied to which input data and in which order. For example, an incident report

stored in a PDF file format must go through *file metadata separation*, *text extraction*, and *pattern processing.*

The main processes with their associated outputs are included in Figure 2:

- **Text Extraction from Images** recognizes and separates the text in an image (Sumathi *et al.*, 2012). This step prepares images and PDF files for other processing steps such as tagging. Text extraction is especially important in the case of diagrams such as flowcharts or event-driven process chains because these documents contain large amounts of text that can be used for tagging.
- **File Metadata Separation** makes use of file and directory attributes, including file name, creation date, last modified date, owner, and access permissions. For example, the creation date and last modified date can assist in distinguishing newer and potentially more relevant information from older and possibly outdated information.
- **Pattern Processing** makes use of existing patterns within documents to extract the desired structure. Hristidis *et al.* (2010) observed that most of the available information is stored in unstructured documents, but that "typically the same organization follows a similar format for all its reports" (Hristidis *et al.*, 2010). Therefore, it is feasible to use patterns for information extraction.
- **Simulation Model Transformation** is the process of converting simulation models into a representation which enables model queries and integration with other disaster-related data. Simulation is considered especially important for this study because it involves various domains which are crucial to disaster management. To extract as much knowledge as possible from simulation model files, an ontology-based representation of simulation models has been developed (Grolinger *et al.*, 2012). Unlike text-processing approaches, an ontology-based representation makes it possible to 1) address simulator-specific terminology, 2) remain schema-independent because ontologies do not have predefined schemata, and 3) focus on entities and their relations.
- **Tagging and semantic annotation.** Tagging is the process of attaching keywords or terms to a piece of information with the objective of assisting in classification, identification, or search (Wang *et al.*, 2012). Semantic annotations additionally specify how entities are related. In disaster-management data tagging, both manual and automated tagging are needed. Automated tagging applies various natural language processing (NLP) and soft computing techniques to add tags automatically to pieces of information.

The processes presented above are common processes for addressing file-style data; nevertheless, CKaaS can be easily expanded to include new data processes.

3.1.3 *Storage in the cloud environment*

Relational databases (RDBs) are traditional data storage systems designed for structured data. They have been used for decades due to their reliability, consistency, and query capabilities through SQL. However, they do not gracefully meet mass data needs. In other words, RDBs exhibit horizontal scalability challenges, big data inefficiencies, and limited availability (Han *et al.*, 2011).

In this context, the next generation of databases, namely NoSQL data stores, have been designed for a distributed environment (Kossmann and Kraska, 2010). They are

mainly dedicated to projects that are distributed, that involve large amounts of data, or that must scale. In the case of simple operations, NoSQL data stores improve performance relative to traditional RDBs. Consequently, as illustrated in Figure 2, the CKaaS storage solution incorporates NoSQL data stores. The work of Grolinger *et al.* (2013b) provides details of database use in the context of a Disaster-CDM approach, which is used in this study.

3.2 Knowledge delivery

The consumer and the *KaaS Broker* can access the KaaS *Cloud Provider's* service directly through query endpoints or by means of the SaaS layer which offers the services to reply to knowledge requests. As presented in Figure 2, knowledge consumption is mainly achieved through:

- *Ontologies*: These provide an overall view of the local ontologies representing each database independently of its category. Ontologies represent a mapping between heterogeneous sources, which is needed to unify query capabilities. A query endpoint is provided to access the KaaS *Cloud Provider* and enable direct querying of underlying data. This makes KaaS consumers unaware of the storage architecture and provides a unified view of the data. Specifically, the SPARQL endpoint is used here, but other kind of endpoints can be integrated as well.
- *Data interfaces:* After querying the ontology, it is necessary to access the data. Data interfaces enable translation of the generic query into a specific language that corresponds to the underlying database system. Thus, data stored in heterogeneous sources can be accessed, analyzed, and administered.

4 Case study

This section presents the ability of CKaaS to manage heterogeneity and semantics in both *inter-cloud* and *intra-cloud* environments. Section 4.1 demonstrates the behaviour of the CKaaS in an inter-cloud environment and shows collaboration of the distributed KaaS *Cloud Providers*. Section 4.2 illustrates CKaaS in an intra-cloud environment; it shows how a KaaS *Cloud Provider* operates with domain-specific simulation models.

4.1 CKaaS in inter-cloud management: CKaaS collaborative behaviour

Collaboration of knowledge providers in the disaster management domain is essential for successful decision-making; without this collaboration, knowledge provided to consumers might be incomplete. In a classic cloud architecture, the *Cloud Consumer* is connected to a single *Cloud Provider* to benefit from the services offered. If that *Cloud Provider* cannot satisfy the request, no answer is delivered. As a result, the *Cloud Consumer* cannot make informed decisions.

Collaboration between different KaaS *Cloud Providers* (*inter-cloud* management) remains necessary to satisfy *Cloud Consumer* requirements by integrating service providers' knowledge. CKaaS enables this through the *Cloud Broker*. As presented in Figure 3, the *Cloud Consumer* sends its request to the *KaaS Broker*. The *KaaS Broker* refers first to the knowledge-based cache; if it can find the answer to the request there, it sends the response back to the consumer. However, if the knowledge is not found, or in

other words, if the cache cannot satisfy the request, advanced services are activated which enforce collaboration between several KaaS *Cloud Providers*.

The first service activated is *service intermediation.* In this example, it is assumed that the request is complex. Therefore, service intermediation interprets this request and decomposes it into a sequence of requests. For the requests that cannot be satisfied by the *KaaS Broker*, service intermediation adapts them accordingly to the *Cloud Providers'* API characteristics and replicates them to the KaaS *Cloud Providers* to obtain all their responses. To provide a complete response, it waits for all *Cloud Provider* responses. After gathering all responses, service aggregation combines them and provides the *KaaS Broker* with the new response (knowledge service learning). Finally, the *KaaS Broker* provides the *Cloud Consumer* with the right response.

CKaaS collaboration is illustrated with a simple scenario in the Emergency Response and Crisis Management System that contains two KaaS *Cloud Providers* and a *Cloud Consumer*. The first KaaS provider knows about critical infrastructures, including gas distribution; specifically, its knowledge is in simulation models which describe different infrastructures and have been used to explore the behaviours of real-life infrastructures. The second KaaS provider offers best practices and recommendations about gas-related disaster management decisions. The *Cloud Consumer* represents the supervisor who manages the gas infrastructure and makes decisions about the operation of the gas infrastructure.

If a fire is threatening the lines supplying gas to a particular area, the supervisor (*Cloud Consumer*) sends a request to the *Cloud Broker* to make the right decision about turning off the gas. This decision will be based on collaboration between the two KaaS *Cloud Providers* as follows: the first KaaS provider will find, using the simulation models, the elements that are supplied with gas, while the second KaaS will recommend to the supervisor how to deal with this situation and what actions are required (e.g., turning off the gas in the region where the supplied elements are detected).

The next section provides a detailed description of the KaaS *Cloud Provider* and illustrates how the KaaS *Cloud Provider* operates in an *intra-cloud* environment. Specifically, the example uses simulation models to determine which entities would be impacted by a gas shortage.

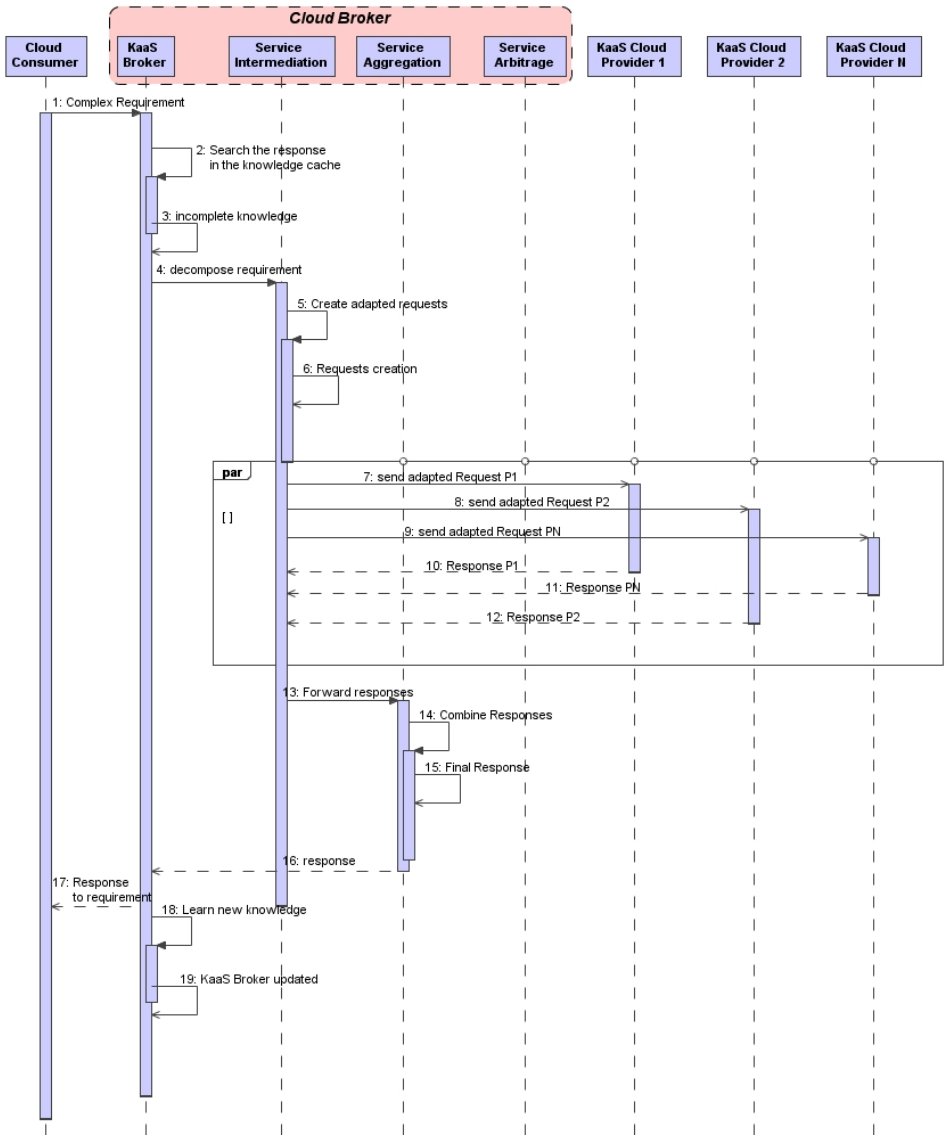

Figure 3: CKaaS behaviour in an inter-cloud environment.

4.2 CKaaS in intra-cloud management: KaaS disaster cloud provider

Intra-cloud KaaS operations are illustrated here using a simulation model example. Because simulation models represent abstractions of real-world systems, they contain information about interconnections and dependencies among entities. In the described crisis scenario, the consumer needs to know which entities will be affected if the gas supply is turned off. The simulation models contain connections among entities and therefore can be used to provide the needed information.

Specifically, the I2Sim (Rahman *et al.*, 2008) model, which was developed for the investigation of infrastructure interdependencies, is used here. I2Sim is an interdependency simulator built upon MATLAB's Simulink engine. Simulink provides block libraries which can be customized to conform to a specific simulation domain. Complex models are managed by dividing models into hierarchies of sub-models. Accordingly, I2Sim builds upon Simulink by customizing Simulink blocks and providing entities specific to infrastructure interdependency simulation.

The I2Sim simulator model used in this case study was developed to investigate infrastructure interdependencies in an incident on the Western University campus. The model involves a number of infrastructures, including electricity, water, gas, and steam distribution. It is complex and consists of several levels of hierarchy. These hierarchy levels hide complexity and aid in model creation and management; however, they pose a challenge for model querying. Storing ontology-based representations of simulation models in a database provides querying abilities. The simulation models are first processed to convert them into ontology-based models; then they are saved in a database.

4.2.1 *Simulation knowledge extractor*

As described in Section 3.1.2, simulation models are processed by transforming each simulator-specific proprietary model to its corresponding ontology-based model. In the case of I2Sim, the simulator model is stored in a Simulink-style .mdl file. The transformation of this .mdl file to an ontology-based model has been described in the work of Grolinger *et al.* (2012). In this case study, OWL (W3C OWL, 2009) was used as ontology language because it is the W3C-recommended ontology language. The simulation model used in this case study was transformed to an ontology-based model with 679 instances and 6575 property assertions.

4.2.2 *Storing simulation knowledge*

The KaaS disaster *Cloud Provider* is designed to enable the choice of a storage solution that corresponds to data requirements in terms of data structure as well as access patterns.

After the simulation models have been transformed into ontology-based models, they are represented in OWL, which is characterized by a formal semantic and an abstract ontology structure that can be perceived as a graph. Graph databases use graph structures with nodes, edges, and properties to represent and store data. They are optimized for efficient management and storage of graph-like data. Consequently, because ontologies can be perceived as graphs, it is apparent that graph databases are a good choice for storing ontologies as well as ontology-based simulation models. Another characteristic that makes a graph database a good choice is its query capabilities. Graph database implementations typically offer query capabilities using specialized graph query languages. Specifically, this case study uses the Neo4j graph database (Neo4j, 2013). Neo4j can be queried using Cypher, a proprietary graph query language developed by Neo4j; using Gremlin, a graph traversal language; or even using the RDF query language, SPARQL.

The processing stage creates ontology-based representations of simulation models in the OWL language. Next, these ontologies are loaded into Neo4j. Because Neo4j is a graph database and OWL ontologies are forms of graphs, loading ontologies into the database proved to be straightforward. Loading the present use case model into a database resulted in a graph with 2533 vertices and 9724 edges.

4.2.3 *Knowledge delivery*

The KaaS disaster *Cloud Provider* makes knowledge available as a service and provides services responsible for querying the KaaS within the *Cloud Provider*. One of these services is responsible for finding specific named entities in simulation models together with any entities to which they are connected or related. Within CKaaS, this is achieved by database querying, which is possible because simulation models have been transformed to their corresponding ontology-based representations and stored in a database. In this use case, knowledge delivery is achieved through a SPARQL endpoint as recommended by the W3C. The implemented SPARQL query is:

```
SELECT ?instanceName ?class ?subModelName ?connectedTo
WHERE {?instance simmodel:Name "Gas".
        ?instance a ?class.
        ?instance simmodel:Name ?instanceName.
        ?channel simupper:hasStartNode ?instance.
        ?channel simupper:hasEndNode ?connectedToNode.
        ?connectedToNode simmodel:Name ?connectedTo.
        OPTIONAL {?instance i2sim:parentSystem ?subModel.
                  ?subModel simmodel:Name ?subModelName.}.
      }
ORDER BY ?class ?connectedTo
```

This query looks for all ontology instances, or in graph terminology, nodes named "Gas" together with all entities to which they are connected. Entities in the I2Sim simulation model are connected by *channels*, which are also responsible for transporting items among other entities. Channels have *hasStartNode* and *hasEndNode* properties, which indicate which nodes/entities each channel connects. In the presented query, the two properties are used to identify all entities to which "Gas" connects.

A few rows of the results of this query are displayed in Table I. The first column is the name of the entity that was searched for, the second column is the type of I2Sim element used to model the "Gas" entity, the third column is the sub-model to which the "Gas" entity belongs, and the last column is the entity to which it connects. The first row identifies an entity of type *I2Sim_source*. In I2Sim, *source* elements represent the origin of resources and are typically used to model resources external to the simulation model. Therefore, the first row of the table indicates that the simulation model under study is using an external gas supplier which is connected to *Steam house*. The next four rows indicate that within the *Steam house* sub-model, the "Gas" entity connects to the four boilers, and the final four rows indicate that within the four boilers' sub-models, "Gas" connects to *Combustion chamber*.

TABLE I. QUERY OUTPUT

instanceName	**class**	**subModelName**	**connectedTo**
Gas	isSim:i2sim_source		Steam house
Gas	i2Sim:inport	Steam house	Boiler 1
Gas	i2Sim:inport	Steam house	Boiler 2
Gas	i2Sim:inport	Steam house	Boiler 3
Gas	i2Sim:inport	Steam house	Boiler 4
Gas	i2Sim:inport	Boiler 1	Combustion chamber
Gas	i2Sim:inport	Boiler 2	Combustion chamber
Gas	i2Sim:inport	Boiler 3	Combustion chamber
Gas	i2Sim:inport	Boiler 4	Combustion chamber

Ultimately, this query identifies which entities within a specific simulation model are using gas as a supply. Consequently, these entities will be affected if the gas supply is

turned off. To obtain the same information directly from the simulation model without querying, the user needs to open the simulation model, find all gas elements, and check to which entities each connects. The hierarchy of sub-models makes this task especially challenging because each sub-model needs to be checked as well.

The same query could have been executed against an OWL ontology without storing the ontology in the database. However, disaster management deals with a large number of simulation models, making use of a database preferable to storing ontologies as OWL files.

This case study has demonstrated how CKaaS behaves in an intra-cloud environment on an example involving a simulation model. Specifically, it has demonstrated *Cloud Provider* services on an example of simulation model querying.

5 Related Work

Knowledge management is crucial in a number of fields, including health science, environmental science, computer science, and a number of engineering disciplines. This paper focuses on the disaster management domain. Consequently, this section first reviews works related to KaaS and then disaster data management studies.

5.1 Knowledge as a Service (KaaS)

Nowadays, cloud computing offers computing capabilities as services and represents an environment suitable for collaboration and capable of delivering horizontal scalability and high availability (Almorsy *et al.*, 2011). A number of research studies (Lai *et al.*, 2012), (Langenberg *et al.*, 2011), (Ju and Shen, 2011) have pointed out the importance of providing a cloud knowledge system in different domains to facilitate sharing and accessing knowledge from different sources. A new concept, "Knowledge as a Service", was defined by Xu *et al.* (2005) as the process by which a knowledge service provider answers queries presented by knowledge consumers through a knowledge server. Knowledge is typically extracted from large volumes of data coming from heterogeneous data owners according to knowledge models such as ontologies and is then delivered as a cloud computing service. Based on these knowledge models, the knowledge server is able to deliver the right answer to the right consumer at the right time (Abdullah *et al.*, 2011).

Several researchers have used the KaaS approach to build cloud-based knowledge solutions (Qirui, 2012), (Kannimuthu, 2012), (Lai *et al.*, 2012), including disaster management solutions (Lino *et al.*, 2012). Qirui (2012) brought new thinking to agricultural information-system development by using the KaaS approach. In this approach, KaaS provides services that offer recommendations about planting on the farm according to user specifications and environmental factors. The knowledge representation in this KaaS is based on ontologies, while the data are stored exclusively in a relational database (MySQL). Similarly to the approach proposed by Qirui (2012), CKaaS also relies on ontologies for data integration; however, in contrast to Qirui's method which stores data exclusively in a relational database, the CKaaS approach takes advantage of NoSQL data stores.

Kannimuthu *et al.* (2012) applied KaaS in the e-commerce domain, where they focussed mainly on how to extract knowledge from data using data mining techniques. After a user selects items, the utility mining service uses information about the selected items to extract knowledge from large quantities of data and subsequently to attract the user to other products of the same enterprise. Ultimately, this leads to financial benefit for

the enterprise. In their approach, data are formatted according to XML and stored in an XML database. CKaaS, similarly to the work of Kannimuthu *et al.* (2012), provides a means to extract knowledge from data; however, while Kannimuthu *et al.* focussed specifically on mining for product recommendation, the scope of knowledge extraction in CKaaS is wider because it includes various kinds of disaster-related information. Moreover, in CKaaS, knowledge extraction capabilities are made available as services, which facilitates their reuse.

In contrast to the work of Qirui (2012) and Kannimuthu *et al.* (2012), the CKaaS solution proposed in this work is not limited to structured data. Rather, it relies on ontologies to integrate both structured and unstructured data stored in NoSQL data stores.

Another interesting approach was proposed by Lino *et al.* (2012), who used KaaS to facilitate emergency response in natural disasters like tsunamis and earthquakes using interactive digital TV. To support smart applications and to share knowledge and planning information, their solution integrates a semantic layer based on interactive digital TV (IDTV) middleware. Specifically, knowledge is shared by means of ontological descriptions. The work of Lino *et al.* focussed on implementing a planning algorithm for emergency response in the KaaS layer to support evacuation of unsafe areas. In contrast, the CKaaS approach proposed in this work targets a wider context by integrating heterogeneous knowledge sources for more generic decision-making. Moreover, despite its use of a KaaS approach, the solution provided by Lino *et al.* seems to be restricted to a specific client/server architecture as opposed to an accepted cloud computing architecture.

Qirui (2012), Kannimuthu *et al.* (2012), and Lino *et al.* (2012) all proposed a KaaS based on a cloud architecture; however, they did not follow a well-accepted cloud computing reference model such as those proposed by NIST, CISCO, or IBM. In contrast, CKaaS benefits from the use of a standard cloud computing architecture, specifically the NIST architecture, to provide flexible and scalable KaaS solutions. It was decided to follow the NIST reference architecture in this research because it is a generic cloud computing model, while the others, including the CISCO and IBM solutions, are more specialized in specific areas such as business and communication networks.

Similarly to the proposed CKaaS, Ju and Shen (2011) introduced a KaaS system as an extension of the NIST cloud computing model. They considered KaaS as a fourth layer on top of SaaS. Likewise, Abdullah *et al.* (2011) placed KaaS onto each layer of the standard cloud computing architecture, yielding four layers: Knowledge-Infrastructure as a Service (K-IaaS), Knowledge-Platform as a Service (K-PaaS), Knowledge-Data as a Service (K-DaaS), and Knowledge-Software as a Service (K-SaaS). In contrast to the works of Ju and Shen (2011) and Abdullah *et al.* (2011), the KaaS layer in CKaaS is considered as a sub-layer within the PaaS layer, but on top of other PaaS components. Therefore, KaaS can take full advantage of platform-level and infrastructure-level services to deliver adequate knowledge services built from distributed and collaborative sources.

In contrast to the reviewed solutions which use XML or relational databases or do not address the storage aspect, CKaaS enables a choice of storage solution that best corresponds to data requirements in terms of data structure and access patterns. In the context of CKaaS, NoSQL solutions provide schema flexibility, horizontal scalability and high availability. As for the collaboration aspect, the reviewed solutions do not support collaboration of knowledge providers, while CKaaS enables such collaboration by means of the *KaaS Broker*.

Finally, the reviewed solutions do not deal with integrating knowledge from different knowledge providers, whereas CKaaS aims to solve this problem through collaboration

among multiple KaaS providers. Moreover, the reviewed solutions dealt only with inter-cloud interaction, while CKaaS includes intra-cloud interaction of knowledge providers, thus enabling collaboration.

5.2 Disaster management

Crisis informatics (Palen *et al.*, 2010), (Schram and Anderson, 2012), the area of research concerned with the role of information and technology in disaster management, has been attracting increased research attention recently. Data are the main factor in disaster management because they represent a description of the environment, disaster plans, and resources and consequently are the basis for analysis and decision-making.

Hristidis *et al.* (2010) surveyed data management and analysis in the disaster management domain. The main focus of their survey was on data analysis techniques without the storage aspect. In contrast, in CKaaS, storage and analysis are considered as integral parts of the solution. Moreover, CKaaS provides advanced techniques that transform data into knowledge and deliver it as a service to provide a high quality of experience to users. Hristidis *et al.* (2010) identified the following data analysis technologies as relevant to disaster data management: information extraction, information retrieval, information filtering, data mining, and decision support. Similarly, CKaaS uses a number of information extraction and retrieval technologies to provide knowledge. Their survey revealed that most research has focussed on a very narrow area of disaster management, for example, a specific disaster event such as an earthquake or a flood, or on specific disaster-related activities such as communication among actors, estimating disaster damage, and use of mobile devices. Hristidis *et al.* (2010) recognized the need for flexible and customizable disaster management solutions that could be used in different disaster situations. CKaaS aims to provide such a solution using cloud computing extended by the KaaS approach, ontologies, and NoSQL approaches.

Silva *et al.* (2011) aimed to integrate diverse, distributed information sources by bringing them into a standardized and exchangeable common data format. Their approach focussed on data available on public Web sites. Data were first extracted from various source Web sites and stored in a relational database. Next, the data were transformed into Linked Open Data (LOD) form and published. In contrast to their work, which addressed data available on public Web sites, the proposed CKaaS can accommodate various information sources.

Palen *et al.* (2010) presented a vision of technology-supported public participation during disaster events. They focussed on the role of the public in disasters and how information and communication technology can transform that role. Similarly to Hristidis *et al.* (2010), they recognized information integration as a core concern in crisis informatics. While Palen *et al.* (2010) presented a vision, our work focuses on providing an architecture for cloud data management.

Anderson and Schram (2011), like Palen *et al.* (2010), studied the role of public and social media in disaster events. They proposed a crisis-informatics data-analysis infrastructure for collection, analysis, and storage of information from Twitter. The main objective of their work was support of other crisis information research by extracting disaster-related tweets from Twitter and storing them in a database. In their initial study (Anderson and Schram, 2011), data were stored in a relational database, specifically MySQL. Later, after encountering scalability challenges, they transitioned to a hybrid architecture that incorporates a relational database and a NoSQL data store (Schram and Anderson, 2012). Similarly, CKaaS allows for use of relational databases and NoSQL data stores for data storage. However, in the CKaaS approach different NoSQL data

stores can be used to address the storage requirements of diverse data. Specifically, CKaaS enables a choice of storage solutions to suit data structures and access patterns.

The listed studies have focussed on data analysis for disaster management; however, to obtain the right decision/response in critical situations, data management must be enriched with knowledge management. Therefore, CKaaS addresses the need for knowledge integration and knowledge-sharing solutions through transforming and formalizing structured and unstructured data into knowledge. To overcome the problem of semantic heterogeneity when integrating various knowledge sources, the CKaaS approach uses ontologies. Moreover, publishing knowledge as a service (KaaS) provides scalable management and facilitates use of knowledge in practice.

6 Limitations, challenges, and opportunities

The CKaaS proposed in this work extends the NIST cloud computing reference architecture by adding a KaaS layer which is responsible for integrating diverse data sources. Because the CKaaS approach is based on cloud computing, it is exposed to a number of limitations and challenges similar to those encountered by cloud computing. The main limitations, challenges, and opportunities faced by the CKaaS approach include the following:

- The CKaaS architecture, similarly to the NIST reference architecture, relies on a *Cloud Broker*, which acts as a mediator between the *Cloud Consumers* and the KaaS services of various *Cloud Providers*. If the system relies on a single *Cloud Broker*, a single point of failure is introduced. The use of multiple *Cloud Brokers* in the proposed architecture and their coordination and communication require further research.

- The *Cloud Broker* is responsible for gathering and integrating knowledge from various service providers. Even though knowledge integration is not the focus of this work, it should be addressed to ensure successful provision of the comprehensive knowledge service. Because different *KaaS Providers* collaborate to answer consumers' requests, there is a possibility that knowledge conflicts will occur. The *Cloud Broker* must first detect those conflicts and then resolve or manage them so that non-contradictory knowledge can be provided to consumers. Moreover, since *Cloud Providers'* knowledge may evolve differently and at different pace, the *Cloud Broker* needs to coordinate knowledge across different providers.

- The CKaaS architecture, similarly to the NIST reference architecture, includes security and privacy components as part of the *Cloud Provider*. However, security and privacy span all components of the proposed architecture and involve both service consumers and providers. In a public cloud, data are stored and processed on third-party premises and in a shared multi-tenant environment; therefore, security and privacy vulnerabilities are increased. Providing an adequate solution is difficult because it needs to be addressed in the context of the proposed architecture and it needs to include both the service provider and the service consumer.

- Quality of service (QoS) is outside the scope of this work; nevertheless, QoS represents a major challenge in the CKaaS context because of the large number of components and actors involved in providing knowledge as a service. A vital component with respect to QoS is the *Cloud Broker*, which is responsible for

integrating information from various providers. Moreover, the *Cloud Broker* is in charge of deciding the waiting response time from KaaS knowledge providers.

- Customer lock-in. Due to lack of standardization within the cloud computing industry, it is challenging to move from one *Cloud Provider* to another. Moreover, customer lock-in makes *Cloud Consumers* vulnerable to price increases.

7 Conclusions

This paper has proposed a CKaaS architecture based on the NIST cloud architecture integrating a domain-independent KaaS layer. CKaaS stores large amounts of data while maintaining high availability using NoSQL and cloud solutions. Data search, interoperability, and integration are facilitated through knowledge acquisition and knowledge delivery. Knowledge acquisition uses language processing, information extraction, and retrieval techniques to add structure and metadata to largely unstructured disaster data. Knowledge is delivered as a service using the KaaS approach so that service performance can be managed by the cloud management services. CKaaS overcomes the limitation of knowledge integration by implementing the *Cloud Broker* with the aim of implementing collaborative distributed cloud knowledge system through enriched services. In this work, CKaaS has been applied to the disaster management domain because the quantity and heterogeneity of disaster-related data are large and managing them effectively remains crucial for minimizing the impact of disasters on society.

The case study presented in this work provides evaluation of the proposed CKaaS architecture; nevertheless, further evaluation will be performed including complex and heterogeneous data sources. Critical aspects that need to be addressed are the integration of diverse knowledge provided by various service providers and the criteria for optimal data storage selection.

References

Abdullah, R., Eri, Z.D., Talib, A.M. (2011) 'A Model of Knowledge Management System for Facilitating Knowledge as a Service (KaaS) in a Cloud Computing Environment'. *Proceedings of the International Conference on Research and Innovation in Information Systems (ICRIIS)*, pp. 1–4.

Almorsy, M., Grundy, J., Ibrahim, A.S. (2011) 'Collaboration-Based Cloud Computing Security Management Framework', *Proceeding of the IEEE International Conference on Cloud Computing (CLOUD)*, pp. 364–371.

Anderson, K.M., Schram, A. (2011) 'Design and Implementation of a Data Analytics Infrastructure in Support of Crisis Informatics Research: NIER Track'. *Proceedings of the 33rd International Conference on Software Engineering*, pp. 844–847.

Brewer, E. (2012) 'CAP Twelve Years Later: How the "Rules" have Changed', *Computer*, Vol. 45, No. 2, pp. 23–29.

Erl, T., Mahmood, Z., and Puttini, R. (2013) *Cloud Computing: Concepts, Technology, & Architecture*, Prentice Hall, Upper Saddle River, NJ.

Grolinger, K., Higashino, W.A., Tiwari, A., Capretz, M.A.M. (2013a) Data Management in Cloud Environments: NoSQL and NewSQL Data Stores, *Journal of Cloud Computing: Advances, Systems and Application*, Springer Open, Vol. 2, doi:10.1186/2192-113X-2-22.

Grolinger, K., Capretz, M.A.M., Mezghani, E., Exposito, E. (2013b) 'Knowledge as a Service Framework for Disaster Data Management', *Proceedings of the 22nd IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pp. 313–318.

Grolinger, K., Capretz, M.A.M., Marti, J.R., Srivastava, K.D. (2012) 'Ontology-Based Representation of Simulation Models'. *Proceedings of the 24th International Conference on Software Engineering and Knowledge Engineering*, pp. 432-437.

Gupta, R., Gupta, H., Mohania, M. (2012) 'Cloud Computing and Big Data Analytics: What Is New from Databases Perspective?', Big Data Analytics Lecture Notes, in *Computer Science*, Vol. 7678, pp. 42–61.

Han, J., Song, M., Song, J. (2011) 'A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing', *Proceedings of the 10th IEEE/ACIS International Conference on Computer and Information Science*, pp. 351–355.

Hecht, R., Jablonski, S. (2011) 'NoSQL Evaluation: A Use Case Oriented Survey', *Proceedings of the Conference on Cloud and Service Computing*, pp. 336–341.

Hristidis, V., Chen, S., Li, T., Luis, S., Deng, Y. (2010) 'Survey of Data Management and Analysis in Disaster Situations', *Journal of Systems and Software*, Vol. 83, No. 10, pp. 1701–1714.

Hu, Y., Qu, W. (2013) 'Efficiently Extracting Change Data from Column Oriented NoSQL Databases', *Advances in Intelligent Systems and Applications* – Volume 2, *Smart Innovation, Systems and Technologies*, Vol. 21, pp. 587–598.

Ju, D., Shen, B. (2011) 'On Building the Knowledge Cloud'. *Proceedings of the International Conference on Computer Science and Service Systems (CSSS)*, pp. 2351–2353.

Kannimuthu, S., Premalatha, K., Shankar, S. (2012) 'Investigation of High-Utility Itemset Mining in Service-Oriented Computing: Deployment of Knowledge as a Service in E-Commerce', *Proceedings of the Fourth International Conference on Advanced Computing (ICoAC)*, pp. 1–8.

Kossmann, D., Kraska, T. (2010) 'Data Management in the Cloud: Promises, State-of-the-Art, and Open Questions', *Datenbank-Spektrum*, Vol. 10, No. 3, pp. 121–129.

Lai, I., Tam, S., Chan, M. (2012) 'Knowledge Cloud System for Network Collaboration: A Case Study in Medical Service Industry in China', *Expert Systems with Applications*, Vol. 39, No. 15, pp. 12205–12212.

Langenberg, D., Kind, C., Dames, M. (2011) 'Knowledge Management in Cloud Environments'. *I-KNOW*, Lindstaedt, S.N., Granitzer, M. (eds.), ACM, Vol. 36.

Lino, N.C.Q., Siebra, C.A., Amaro Tate, M.A. (2012) 'EmergencyGrid – Planning in Convergence Environments', *22nd Int. Conf. on Automated Planning and Scheduling, SPARK Workshop*.

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D. (2011) 'NIST Cloud Computing Reference Architecture', NIST Special Publication 500-292, http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505. (Accessed 22 November 2013)

Mohanty, S., Jagadeesh, M., Srivatsa, H. (2013) *Big Data Imperatives: Enterprise Big Data Warehouse, BI Implementations and Analytics*, Apress, Berkeley, CA.

Neo4j, http://www.neo4j.org/, 2013. (Accessed 22 November 2013)

Palen, L., Anderson, M.G., Martin, J., Sicker, D., Palmer, M., Grunwald, D. (2010) 'A Vision for Technology-Mediated Support for Public Participation and Assistance in Mass Emergencies and Disasters', *Proceedings of the Conference on Visions of Computer Science*, pp. 1–12.

Qirui, Y. (2012) 'KaaS-Based Intelligent Service Model in Agricultural Expert System', *Proceedings of the 2nd International Conference on Consumer Electronics, Communications, and Networks*, pp. 2678–2680.

Rahman, H.A., Armstrong, M., Mao, D., Marti, J.R. (2008) 'I2Sim: A Matrix-Partition Based Framework for Critical Infrastructure Interdependencies Simulation', *Proceedings of the Electric Power Conference*, pp. 1–8.

Sakr, S., Liu, A., Batista, D.M., Alomari, M. (2011) 'A Survey of Large-Scale Data Management Approaches in Cloud Environments', *IEEE Communication Surveys & Tutorials*, Vol. 13, No. 3, pp. 311–336.

Sakr, S. (2013) 'Cloud-Hosted Databases: Technologies, Challenges and Opportunities', *Cluster Computing*, Springer, pp. 1–16.

Sadalage, P.J., Fowler, M. (2013) *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*, Addison-Wesley, Upper Saddle River, NJ.

Schram, A., Anderson, K.M. (2012) 'MySQL to NoSQL: Data Modeling Challenges in Supporting Scalability'. *Proceedings of the Third Conference on Systems, Programming, and Applications: Software for Humanity*, pp. 191–202.

Silva, T., Wuwongse, V., Sharma, H.N. (2011) 'Linked Data in Disaster Mitigation and Preparedness', *Proceedings of the Third International Conference on Intelligent Networking and Collaborative Systems*, pp. 746–751.

Sumathi, C.P., Gayathri Devi, G., Santhanam, T. (2012) 'A Survey on Various Approaches to Text Extraction in Images', *International Journal of Computer Science and Engineering Survey*, Vol. 3, No. 4, pp. 27–42.

Stonebraker, M., Madden, S., Badi, D.J., Harizopoulos, S., Hachem, N., and Helland, P. (2007) 'The End of an Architectural Era: (it's Time for a Complete Rewrite),' *Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 1150–1160.

W3C OWL Working Group (2009) ‘OWL 2 Web Ontology Language’. http://www.w3.org/TR/owl2-overview/. (Accessed 22 November 2013)

Wang, M., Ni, B., Hua, X., Chua, T. (2012) ‘Assistive Tagging: A Survey of Multimedia Tagging with Human-Computer Joint Exploration’, *ACM Computing Surveys*, Vol. 44, No. 4, pp. 1–24.

Wigan, M.R., Clarke, R. (2013) ‘Big Data’s Big Unintended Consequences,’ *Computer*, Vol. 46, No. 6, pp. 46–53.

Xu, S., Zhang, W. (2005) ‘Knowledge as a Service and Knowledge Breaching’, *Proceedings of the IEEE International Conference on Services Computing*, pp. 87–94.