Institute of Architecture of Application Systems

Compliant Cloud Computing (C3): Architecture and Language Support for User-driven Compliance Management in Clouds

Ivona Brandic, Schahram Dustdar, Tobias Anstett, David Schumm, Frank Leymann, Ralf Konrad

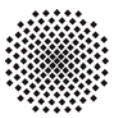
Distributed Systems Group, Vienna University of Technology
{ivona, dustdar}@infosys.tuwien.ac.at

Institute of Architecture of Application Systems, University of Stuttgart
{anstett, schumm, leymann}@iaas.uni-stuttgart.de

T-Systems International GmbH, Frankfurt, Germany
ralf.konrad@t-systems.com

BibTeX:

```
@inproceedings{BrandicDAS10,
  author    = {Ivona Brandic and Schahram Dustdar and Tobias Anstett and
               David Schumm and Frank Leymann and Ralf Konrad},
  title     = {Compliant Cloud Computing (C3): Architecture and Language
               Support for User-driven Compliance Management in Clouds},
  booktitle = {Proceedings of the 3rd International Conference on
               Cloud Computing (IEEE Cloud 2010), 5-10 July, Miami, USA},
  year      = {2010},
  pages     = {244--251},
  doi       = {10.1109/CLOUD.2010.42},
  publisher = {IEEE Computer Society}
}
```

Universität Stuttgart
Germany

Compliant Cloud Computing (C3): Architecture and Language Support for User-driven Compliance Management in Clouds

Ivona Brandic, Schahram Dustdar
Distributed Systems Group
Vienna University of Technology
Vienna, Austria
{ivona,dustdar}@infosys.tuwien.ac.at

Tobias Anstett, David Schumm, Frank Leymann
Institute of Architecture of Application Systems
University of Stuttgart
Stuttgart, Germany
{anstett,schumm,leymann}@iaas.uni-stuttgart.de

Ralf Konrad
T-Systems International GmbH
Frankfurt, Germany
Ralf.Konrad@t-systems.com

Abstract—Cloud computing represents a promising computing paradigm, where computational power is provided similar to utilities like water, electricity or gas. While most of the Cloud providers can guarantee some measurable non-functional performance metrics e.g., service availability or throughput, there is lack of adequate mechanisms for guaranteeing certifiable and auditable security, trust, and privacy of the applications and the data they process. This lack represents an obstacle for moving most business relevant applications into the Cloud. In this paper we devise a novel approach for compliance management in Clouds, which we termed Compliant Cloud Computing (C3). On one hand, we propose novel languages for specifying compliance requirements concerning security, privacy, and trust by leveraging domain specific languages and compliance level agreements. On the other hand, we propose the C3 middleware responsible for the deployment of certifiable and auditable applications, for provider selection in compliance with the user requirements, and for enactment and enforcement of compliance level agreements. We underpin our approach with a use case discussing various techniques necessary for achieving security, privacy, and trust in Clouds as for example data fragmentation among different protection domains or among different geographical regions.

Keywords-Compliance managmenet; SLAs; DSLs;

I. INTRODUCTION

Cloud Computing represents a promising approach for implementing highly scalable software systems for individual-, community-, and business-use [6] [11] [19] [23][22]. In order to achieve that, computing resources have to be allocated to software that has to be executed. The resources are selected based on functional requirements on one hand, and on non-functional requirements on the other, termed Service Level Agreements (SLAs) [13][9][20][5]. The benefits of moving data and applications to the Clouds are manifold. For instance, Cloud computing allows companies to decrease expensive in-house computer systems configured to cope with peak performance by migrating to custom-tailored pay-per-use solutions for computing cycles requested on-demand.

However, Cloud users, e.g., application providers, especially those dealing with sensitive data like customer or patient data, are concerned about a vast number of issues regarding their data being stored and processed in the Cloud [17]. Most of the well-known Cloud products like Amazon's EC2 [1] or Google App Engine [12] provide some basic security, privacy, and trust mechanism, which however often cannot be customized. Requirements like limiting the usage of the submitted data only to those intended by the submitting user or storage of data in certain geographical regions cannot be guaranteed at all. Considerable body of work has been done for the development of user-driven compliance management frameworks considering various aspects as for example compliance to requirements coming from laws, regulations, and internal policies [2][8][21]. However, there is lack of appropriate mechanisms for the compliance management in Clouds. This gives cause for serious concerns related to security, privacy, and trust, which prevents various potential users to move their data and applications into Clouds.

In this paper we propose a novel approach which we term *Compliant Cloud Computing (C3).* We envision that Cloud providers are selected ensuring customizable compliance with the user requirements, such as, security restrictions. To achieve compliance in Clouds we propose the *C3* infrastructure consisting of two major parts: (i) language concepts to express user requirements and Compliance Level Agreements (CLAs) and (ii) a middleware for the deployment of C3-aware applications, for the management of CLAs, and discovery and brokerage of the appropriate Cloud providers. Once the CLAs are agreed between a Cloud provider and a consumer, the C3

middleware manages the enactment of CLAs considering available monitoring information and complying with predefined security, privacy, and trust issues. This includes for example information flow restriction considering geographic and infrastructure affinity or automatic data fragmentation and aggregation among different Cloud providers. Data fragmentation considers dispersal and isolation of data in order to protect sensitive data. An example for data fragmentation would be distribution of personal and medical data of a patient among two different protection domains. Thus, while dispersed personal and medical data has medium security/protection demand, aggregated (i.e., recombined) patient data is highly sensitive.

The major contributions of the paper are (i) presentation and the discussion of a use case and derivation of according requirements for the compliance management in Clouds; (ii) a conceptual design of the languages for the compliance specification and management including Domain Specific Languages (DSLs) and Compliance Level Agreements (CLAs) and (iii) design of the architecture for the automatic discovery of appropriate Cloud providers and compliance management and enforcement.

This paper is structured as follows. In Section 2 we present a motivating use case for our approach and derive requirements for the C3 architecture and languages. Based on the requirements in Section 3 we present the C3's application deployment scenario and role models, followed by the scenario for the sample application execution. Section 4 presents the language design issues considering DSLs and CLAs. In Section 5 we elaborate the implementation issues for the C3 middleware. In Section 6 we discuss related work. Finally, in Section 7 we conclude the paper and point to future work.

II. MOTIVATING USE CASE

In this section, we present the motivating example for the development of the C3 infrastructure. T-Systems Deutsche Telekom [1] operates information and communication technology systems for multinational corporations and public sector institutions delivering various solutions based on global infrastructure of data centers and networks.

A typical example for the T-Systems ICT solutions is the Process Service Platform (PSP) depicted in Figure 1, which is offered as a Software as a Service (*SaaS*) platform. The PSP enables provision of individual services to consumers, facilitates look up mechanisms for common services, classification of services, and orchestration of services to business processes. The PSP is built as a layered architecture including hardware infrastructure, infrastructure services, as for example billing and contracting services, and enterprise service bus containing various business logic services and supporting services. Finally, the service repository and service portal are located on top of the infrastructure. These are customized for the end user to model (process modeling), deploy, and search custom services. In order to efficiently operate global infrastructures and networks and for optimal use of the available resources novel technologies like Cloud computing are gaining more and more on importance. As indicated in Figure 1, PSP infrastructure could be outsourced at different layers e.g., using *IaaS* approach for the hardware infrastructure or using *PaaS* for the infrastructure services.

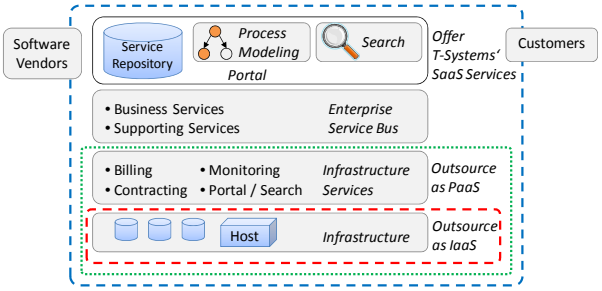


Figure 1: T-Systems' Process Service Platform (PSP)

***KiGa Use Case*:** One of the custom applications offered within the PSP portal is the Kindergarten Portal (*KiGa)* supporting business processes necessary for the management and customer support of the German Kindergarten Association [14]. The business processes include services for the enrolment of children to a particular kindergarten, absence recording of children, and monitoring/controlling of kindergartens. The main objective of the *KiGa Portal* is to support three stakeholders: parents, kindergartens, and communes. Hosting such a portal results in conflicting priorities of efficiently storing data in Clouds and protecting sensitive children's data. There is demand for flexible and trustworthy management of sensitive data in compliance with privacy and security requirements. In this paper we address in particular the following compliance management scenarios: (1) *flexible compliance management of sensitive children's data like information about special diseases* (e.g., diabetes or food allergies), which is necessary for the assignment of trained kindergarten teachers; (2) *guarantees that data is only stored in certain geographical regions* e.g., European Union due to various legal guidelines, that data is managed properly to ensure its safety and compliance with local laws.

Thus, C3 should facilitate the following compliance management issues:

(1) Dispersal of data among different protection domains as for example business and application data, which are dispersed among different Cloud providers.
(2) Storage of data only in certain geographic areas, and guarantees about its adequate retention policies, privacy, integrity, and safety.

In the following sections we discuss architectural and language design decisions based on the two major compliance management issues.

[1] http://www.t-systems.de

III. C3 ARCHITECTURE

To address the problems described above, we present an architecture respectively a framework enabling security, privacy, and trust based compliance management in Clouds. The abstract architecture is divided into Figure 2, Figure 3, and Figure 4. In each figure we highlight and elaborate a particular aspect of the architecture.

A. Application Deployment

C3 infrastructure supports semi-automatic deployment of applications to a *C3-aware Cloud provider*. A C3-aware Cloud provider can execute deployed applications in compliance with predefined security, privacy, and trust requirements as defined by the C3 certification process. Definition of the C3 certification process is part of our ongoing work. Figure 2 depicts the C3's deployment process.

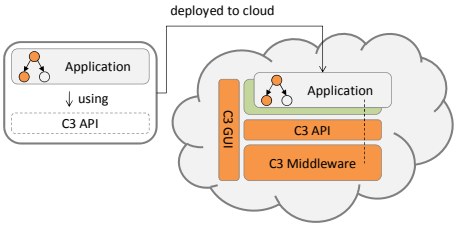


Figure 2: C3's application deployment

C3's API facilitates utilization of the applications for the deployment in the C3 Cloud, e.g. to assign which applications' data can be fragmented. *C3's GUI* is used for the configuration of the deployed application, as for example to tailor the DSL (see Section 4.2) for the particular user group. Once the application is deployed, *C3 Middleware* manages application execution and is responsible for the establishment and generation of CLAs between Cloud providers and consumer. C3 middleware also manages the CLA enactment process. For example, in case of failures C3 middleware can start a renegotiation process in order to select new Cloud providers, which can complete the execution of the particular tasks in compliance with predefined restrictions.

B. Roles

Each C3 role is exemplified using the *KiGa use case*. Before an application can be deployed to a Cloud provider, it has to be created (i.e. programmed) by an *application developer*. Referring to the roles illustrated in Figure 3, an application developer is similar to a (Cloud) adopter or (software/service) vendor, who basically enhances own services and capabilities by exploiting the provided Cloud platform. In case of *KiGa use case* the application administrator would be T-Systems as well. In general, *Cloud providers* offer PaaS to the *application developers* and SaaS to the *application consumers*. In the C3 approach they also act as (Cloud) resellers or aggregators by aggregating Cloud platforms, while maintaining compliance to the specified requirements. That means a C3 Cloud provider has not necessarily to offer resources in terms of typical *SaaS*, *PaaS* or *IaaS* services. A C3 provider could offer supporting services for the compliance management necessary to select an appropriate Cloud provider, who can grant certain security level. Furthermore, we envision a dynamic binding of services, if a provider cannot fulfill the requirements on his own.

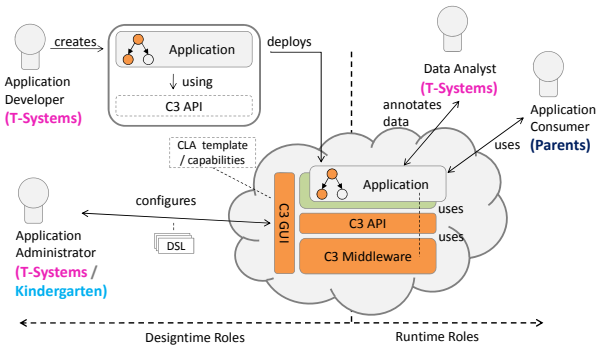


Figure 3: C3's role models

After the *application developer* has deployed the application to a *C3 Cloud provider*, an *application administrator* is responsible for configuring the required (compliance-specific) quality of service by using the predefined *DSLs*. *DSLs* are small languages that can be tailored to a specific problem domain [16]. As indicated implicitly the configuration can be done using either a graphical user interface or a policy-based approach using domain specific languages (DSLs). In case of *KiGa portal*, a *systems administrator* tailors the DLS for the specification of data fragmentation among different Cloud providers. Please note that the role of the application administrator has not necessarily to be implemented by the same company, which developed the Cloud application and could also allow manual or policy-based configuration by multiple tenants. *Application consumers* use the provided service directly (e.g., via a Web client) or integrate it in their tools. In some cases a *Data Analyst* annotates data e.g., which data has to be fragmented. Thus, the parents enrolling their children to the German Kindergarten Association use the *KiGa application* and enter the children's data, which are automatically fragmented among different Cloud providers as specified with the DSL. *Cloud providers* are selected on demand for data storage or application processing in compliance with predefined requirements configured by an application administrator. The details of the provider selection process and the necessary CLA-CLA matching are explained in Section 5. In case of *KiGa portal* one of the Cloud providers could for example be Amazon's EC2 used for the processing of medical data. Another provider could be responsible for storing or processing personal data.

C. Application Execution in C3

In this section we analyze the execution of the deployed application by a C3-aware Cloud provider.

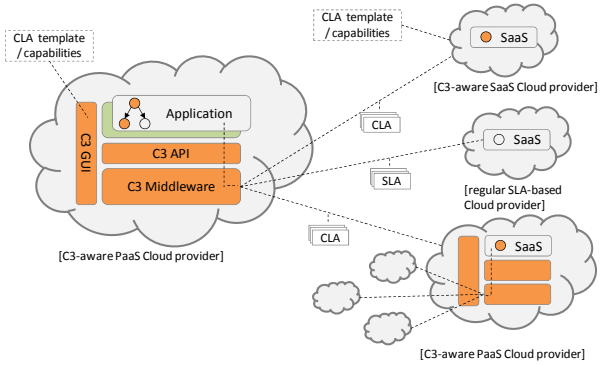


Figure 4: C3's application execution

As shown in Figure 3 we distinguish between three binding scenarios for the Cloud application execution:
(1) Regular SLA-based Cloud provider: represents a regular Cloud provider using SLAs for granting contracts. This type of Cloud provider can be used to process data, which require only basic security, privacy, and trust-related issues.
(2) C3-aware SaaS Cloud provider: represents a *C3-aware SaaS* provider. This type of provider can execute applications in compliance with predefined security, privacy and trust restrictions.
(3) C3-aware PaaS Cloud provider: represents a Cloud provider hosting a platform, which recursively composes other services to fulfill the requested requirements of a *C3-aware PaaS* provider. In such a case the provider could just act as an intermediate without providing a real service other than routing.

In our approach developers can deploy applications represented as a *Composite as a Service (CaaS)*, as explained and discussed in [11]. Business processes usually contain tasks with different level of required security, privacy, and trust management. In the following we explain the C3-based application execution using the *KiGa use case*:

In case of *KiGa portal* some business task can be assigned to *regular SLA-based Cloud provider* using standard SLAs, as for example some infrastructure monitoring services, which do not contain any sensitive children's data. Some other tasks could be assigned to *C3-aware SaaS provider as* for example to providers offering some bookkeeping services. Such providers represent *C3-aware SaaS* providers, who can guarantee that certain personal data necessary for the bookkeeping process are stored within certain geographies (e.g., European Union). Finally, the *C3-aware PaaS Cloud provider* depicted on the left hand side of Figure 4 provides PaaS solution for the deployment and hosting of the *KiGa portal*. Business process tasks are assigned to the different Cloud providers i.e., *regular SLA-based Cloud provider, C3-aware SaaS provider* and *C3-aware PaaS Cloud provider* depending on the required security level of the particular task. Thus, *C3-aware PaaS provider* depicted on the left-hand side of Figure 4 acts as an intermediary providing some value added services i.e., C3-aware compliance management.

IV. C3 Language Support

In this section we present the language support necessary for the proper implementation of the proposed C3 architecture. Firstly, we present how domain specific languages can be designed and tailored in order to support the requirement specification done by C3 API users. Secondly, we discuss Compliance Level Agreements (CLAs), used for the specification of compliance based security, trust, and privacy issues.

A. DSL-CLA

In this section, we present the relation between the DSLs and CLAs. *DSLs* describe the domain knowledge either via a graphical or via a textual syntax.

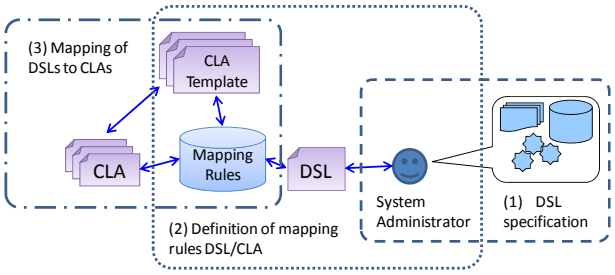


Figure 5: Hierarchical CLA composition

As shown in Figure 5, DSLs are small languages, which are tailored by the application administrators to domain specific modeling elements (step 1). Applied to the *KiGa* use case, the application administrator tailors the DSL for the needs of the German Kindergarten Association (cf. Figure 2). Thus, the system administrator defines mappings from the DLS to CLAs (see Figure 5, step 2). The mapping specification process is described in detail in the next section. Thereafter, the domain knowledge is eventually transformed into other languages (e.g., programming languages, Service Level Agreements, etc.) using predefined mappings. As depicted in Figure 5, in step 3 we translate DSLs into *Compliance Level Agreements (CLAs)*. CLAs are extended Service Level Agreements with the elements for the specification of certifiable and auditable guarantees. Valid *CLA documents* are generated using predefined *CLA templates*, which are CLA documents with all parties, elements, and attributes but without concrete Quality of Service (QoS) values.

B. Model Driven DSL Development

In this section we present an approach to model domain specific languages, which is used for the generation of contracts and agreements using Model Driven Software Development (MDSD) approach [27]. As shown in Figure 6, the Model Driven Development of DSLs is divided into

two parts: the first part is the definition of a DSL e.g., necessary to specify different data protection scenarios (upper part of Figure 6). The second part is the transformation of the DSL into the CLA (down part of Figure 6).

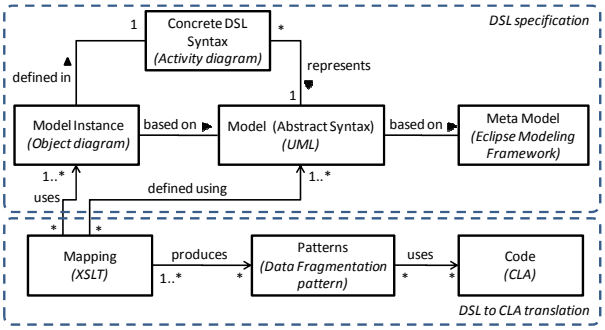


Figure 6 : DSLs based on Model Driven Software Development

As shown in Figure 6, in case of C3 a DSL consists of an *abstract* and a *concrete* syntax. The *abstract syntax* which represents the language model, defines the elements of the domain and their relationships without any particular notation. The italic names represent concrete examples of the *KiGa Use Case* presented in Section 2. Abstract model is based on the meta model. In case of *KiGa*, *UML* represents the abstract model used to model different data protection scenarios e.g., *data fragmentation*. *UML* is based on *Eclipse Modeling Framework (EMF)*. Application administrators can specify different data protection scenarios using UML and the Eclipse Modeling Framework. Concrete DSL syntax describes the representation of the domain elements and their relationship in a form suitable for the DSL users. In case of *KiGa portal* we define *activity diagrams* as concrete DSL syntax for the definition of *data protection patterns*. Model instances are based on the UML and defined in the concrete DSL. For *KiGa* we use *Object diagrams* as model instances, which are defined using *UML's activity diagrams*. Once the *object diagrams* are defined, they can be mapped to CLA code according to the CLA template. To do that we need mappings which translate the object diagrams into the CLAs. Therefore, we use *eXtensible Style Sheets (XSLT)*[2] to translate between the object diagrams and CLAs. Mappings can be defined by application administrators on a high level using *UML*. Thereafter, they are automatically translated to *XSLT*. We distinguish between the high and the low level representation of mappings, e.g. using UML versus using XSLT. The C3 API offers methods for the translation between the high level into the low level mapping representation as described next.

[2] http://www.w3.org/TR/xslt


```
1. ...
2. <xsl:template ...>
3.  <xsl:element name="Function" ...>
4.   <xsl:attribute name="type">
5.    <xsl:text>Times</xsl:text>
6.   </xsl:attribute>
7.   <xsl:attribute name="resultType">
8.    <xsl:text>double</xsl:text>
9.   </xsl:attribute>
10. <xsl:element name="Operand" ...>
11.   <xsl:copy>
12.    <xsl:copy-of select="@*|node()"/>
13.   </xsl:copy>
14.  </xsl:element>
15.  <xsl:element name="Operand" ...>
16.   <xsl:element name="FloatScalar" ...>
17.    <xsl:text>1.27559</xsl:text>
18.   </xsl:element>
19.  </xsl:element>
20. </xsl:element>
21.</xsl:template>
22. ...
```

(a) **(b)**

Figure 7: High level UML mapping (a) and corresponding low level XSLT transformation (b)

Figure 7 (a) represents the high level UML mapping defined by the application administrator. For case of brevity we use a very simple example where we map between dollars and euro.

Figure 7 (b) depicts the corresponding simple XSLT transformation (to Figure 7 (a)). As shown in that figure, the euro metric is mapped to the dollar metric. In this example, we define the mapping rule returning dollars by using the Times function of the WSLA Specification (see line 5) [25]. The Times function multiplies two operands: the first operand is the dollar amount as selected in line 12, the second operand is the dollar/euro quote *(1.27559)* as specified in line 17. The dollar/euro quote can be retrieved by a Web service and is usually not hard coded.

Figure 7 presents a *1:1* mapping rule where one attribute is translated into another one. However, using XSLT and corresponding UML modeling tools even more complex rules can be defined e.g., *1:n*, *n:m*. However, explanation of such rules is out of scope of this paper.

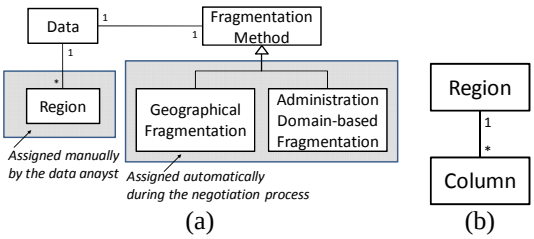


Figure 8: High level Fragmentation Specification (Domain Expert) (a) Low level Fragmentation Specification (Data Analyst) (b)

DSLs can provide multiple levels of abstraction to help multiple stakeholders with different backgrounds and knowledge to express relations and behaviors of a domain with notations they are familiar with. The goal is that each stakeholder can easily understand, validate, and even develop parts of needed solution. For instance, domain experts do not have to deal with technological aspects, such as programming APIs or service interface descriptions.

Similar to the separation of concerns for the mapping specification and as shown in Figure 8, we separate DLS specification into *high level* and *low level.* We exemplify the separation based on *KiGa's data fragmentation problem.*

Figure 8 (a) shows the high level fragmentation specification used by domain experts. The *KiGa administrator* (domain expert) decides which modules of the provided DSL are relevant. As shown in Figure 8 (a) he/she decided that the data stored in the database has to be stored in multiple regions. Furthermore, it has to be decided whether the data has to be fragmented using the geographical fragmentation (e.g. children's private data in Europe, business data elsewhere) or whether the data has to be fragmented among different administration domains i.e., different Cloud providers. As shown in Figure 8 (a) the domain expert decides about the fragmentation method and the concrete resources necessary to fragment data i.e., concrete Cloud providers. Those resources are either geographically or technologically fragmented during the CLA-CLA negotiation (as described in Section 5). The domain expert decides in which parts the data has to be split up and how the fragments have to be distributed to different geographical regions. An example could be splitting into personal and medical data. As for example in case of *KiGa's use case*, Figure 8 (b) shows the low level fragmentation for an application that uses a relational database specified by data analysts. Thus, a data analyst has to assign which columns of the data model have to be assigned to which region.

C. CLA Development

In this section we discuss Compliance Level Agreements and their relation to common Service Level Agreements (SLAs). Service Level Agreements represent negotiated agreements between two parties, namely the service consumer and the service provider. It serves as a legally binding formal or informal contract. There are several (de-facto) standards for the specification of SLAs, e.g. WSLA [25]. The SLA defines a common understanding between the parties about different contract terms including responsibilities, guarantees, warranties, and penalties. Usually, SLAs in computing resource markets specify measurable metrics and the way those metrics are measured, guaranteed, and billed. Measurable metrics include for example availability, response time, and serviceability. We define core CLA components as those elements, which are part of both SLA and CLA specification. Such elements are for example parties, which are involved in the contract. Furthermore, each SLA contains performance relevant objectives as specified in major SLA languages (WSLA [25], WS-Agreement). Thus, as depicted in Figure 9, each SLA is a valid CLA. CLAs extend SLAs with the parameters for the specification of certifiable and auditable objectives in order to provide some kind of measurement for compliance agreements, too. Auditable objectives could include the specific audit method or the necessary audit intervals. In case of KiGa use case certifiable objectives could be an agency, which can certify that a specific Cloud middleware can fragment data among different Cloud providers in compliance with predefined requirements i.e., geographical fragmentation as specified in Figure 8. Herewith we address the current trends in software offerings, where more and more providers certify their services considering data protection and other security concerns. A good example for the certified software offerings are certified online shops [26].

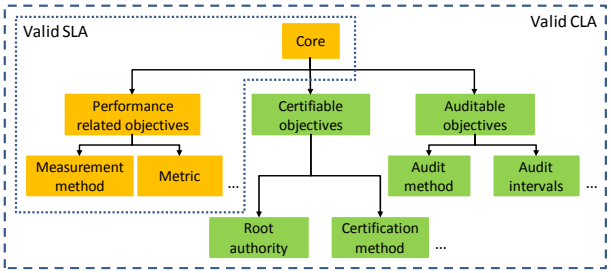


Figure 9: Hierarchical CLA composition

V. C3 Technical Architecture and Middleware

In this section we elaborate the technical architecture for the C3 middleware. As depicted in Figure 10, the C3 middleware consists of the two major parts (1) a *Decision making / deployment component (DCDM)* acting as the connector and mediator between Cloud providers and consumers and the *Runtime Component* being responsible for the enactment of the CLAs. The main tasks accomplished by DCDM are: process of *publishing* of CLA templates (consumer and provider); C3-aware *deployment* of applications (described in Section 3.1); *mapping of DSLs to CLAs* (as described in Section 4.1); *brokering* between consumer and provider by matching requested and provided *CLAs, i.e., CLA-CLA matching*; Business Process Management (BPM); and data fragmentation and execution (as described in Section 4.2).

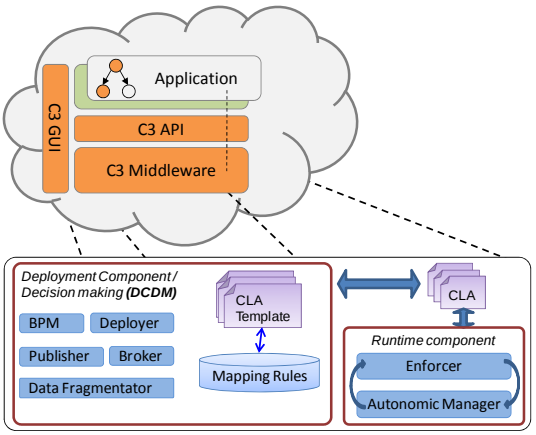


Figure 10: C3 Middleware

For publishing of CLAs we consider advanced approaches beyond typical SLA repositories, as for example such approach presented in [18], in order to facilitate liquid Cloud markets. CLA-CLA matching considers matching of providers and consumers in terms of requested and provided level of security, trust, and privacy. Thus, during the CLA-CLA matching only those providers are preselected, which can deliver specific data protection techniques. In case of *KiGa* this matching process would include pre-selection of Cloud providers, who support data fragmentation technique.

Once the applications are deployed and CLAs agreements between providers and consumers are established, *Runtime Component* deals with the enactment of CLAs and handles events that might lead to CLA violations. The autonomic manager finds reactive actions to a given event, where the event might lead to CLA violations. Our work presented in [3] on self-manageable Cloud services will be extended for the autonomic management of C3-aware Cloud services. As identified in [3] we provide *the self-management interfac*e, which has to be implemented by each application deployed in the C3-aware Cloud. The proper implementation of the self-management interface is ensured during the deployment process though usage of the C3's APIs.

VI. Related Work

Since there is only little work on compliance management in Clouds we look particularly into related areas like (i) usage of SLAs in Clouds, (ii) compliance management in general e.g., in software development cycle and (iii) different data protection techniques like data fragmentation, which however has not yet been applied to Clouds.

Frutos et al. discuss the main approach of the European project BREIN [13]: to develop a framework, which will extend the Grid possibilities by driving their usage inside new target areas in the business domain. BREIN deals with the provision of the basic infrastructure which these new business models need: enterprise system interoperability, flexible relationships, dynamicity in business processes, security mechanisms and enhanced SLA and contract management [5]. However, BREIN applies SLA management to Grids, whereas C3 targets CLA management in Clouds. Brandic et al. presents an approach for adaptive generation of SLA templates [4]. Thereby, SLA users can define mappings from their local SLA templates to the publicly available remote templates in order to facilitate communication with numerous Cloud service providers. This work represents an initial attempt to facilitate on-demand communication between Cloud consumer and provider and will be further investigated for the applicability on top of C3 architecture. Thielman et al. discuss an approach for multi-level SLA management, where SLAs are consistently specified and managed within a service-oriented infrastructure (SOI) [24]. They present the runtime functional view of the conceptual architecture and discuss different case studies including Enterprise Resource Planning (ERP) or financial services. However, they neither consider characteristics of Cloud services nor the auditable and certifiable metrics.

A large body of work has been done on the design and implementation of novel models, languages, and architectural frameworks to ensure dynamic and ongoing compliance of software services to business regulations and design rules. COMPAS (Compliance-driven Models, Languages, and Architectures for Services) is an example European Commission's Framework 7 Specific targeted research project (STREP) dealing with compliance issues [8]. However, very little work has been done on compliance issues in Cloud computing. Anstett et al. propose so-called compliance interfaces that can be used by customers to subscribe to evidence at a provider and to enforce regulations at a provider [2]. They introduced a general architecture that allows compliance to be monitored and enforced at services deployed in different Cloud delivery models. However, the authors do not consider all characteristics of security, privacy, and trust in Clouds, which require novel contract specification languages. Daniel et al. highlights research challenges that need to be addressed in SOA-based compliance governance, spanning design, execution, and evaluation of concerns [10]. They define the compliance management life cycle and the major research goals in compliance governance. For instance, in data outsourcing, privacy constraints on the outsourced data are enforced by combining data fragmentation with encryption and by possibly considering additional assumptions such as different network conditions. Koller et al. [15] discusses autonomous QoS management using a proxy-like approach developed within the SLA@SOI project [20]. The implementation is based on WS-Agreement. Thereby, SLAs can be exploited to define certain QoS parameters that a service has to maintain during its interaction with a specific customer. However, their approach is limited to Web services and does not consider compliance issues in Clouds. Comuzzi et al. defines the process for SLA establishment adopted within the EU project SLA@SOI framework [9].

Work presented in [7] deals with data outsourcing, and privacy constraints on the outsourced data which are enforced by combining data fragmentation with encryption and by possibly considering additional constraints such as the impossibility of external servers to communicate with each other. The approaches presented [7] will be investigated further in context of C3 and possibly applied as a one of the available data fragmentation techniques.

VII. Conclusion and Future Work

In this paper we presented a first attempt to devise concepts for Compliant Cloud Computing (C3). Based on the use case from the telecommunication domain we derived requirements considering compliance management of security, privacy, and trust related issues. We developed

concepts for the languages necessary for the user based requirements specification (domain specific languages) and for the agreement specification (compliance level agreement). We presented the C3 architecture considering application deployment, roles, and the application execution issues. Finally, we presented the technical architecture for the C3 middleware responsible for the application deployment execution, negotiation and enforcement, and compliance level agreements.

In the future we will investigate implementation issues regarding available open source projects. Furthermore, we will develop concepts for the certification processes of the C3-aware Cloud providers.

ACKNOWLEDGMENT

The work described in this paper is supported by the Vienna Science and Technology Fund (WWTF) under grant agreement ICT08-018 Foundation of Self-governing ICT Infrastructures (FoSII), the COMPAS project under the EU 7th Framework Programme ICT Objective (contract no. FP7-215175) and the MASTER project under the EU 7th Framework Programme Information Society Technologies Objective (contract no. FP7-216917).

REFERENCES

[1] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/ 2010.

[2] T. Anstett, D. Karastoyanova, F. Leymann, R. Mietzner, G. Monakova, D. Schleicher, S. Strauch. *MC-Cube: Mastering Customizable Compliance in the Cloud*. In: Springer (Hrsg): Proceedings of the 7th International Joint Conference on Service Oriented Computing, 2009.

[3] I. Brandic. Towards Self-manageable Cloud Services. RTSOAA 2009. In conjunction with the 33rd Annual IEEE International Computer Software and Applications Conference. July 20 - 24, 2009, Seattle, Washington, USA.

[4] I. Brandic, D. Music, Ph. Leitner, S. Dustdar. *VieSLAF Framework: Enabling Adaptive and Versatile SLA-Management*. GECON09. In conjunction with Euro-Par 2009, 25- 28 August 2009, Delft, The Netherlands.

[5] Brein Project (Business objective driven reliable and intelligent Grids for real business), http://www.eu-brein.com/ 2009.

[6] R. Buyya, Ch. Sh. Yeo, S. Venugopal, J. Broberg, I. Brandic. *Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility*, Future Generation Computer Systems, 25(6):599-616, June 2009.

[7] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati. *Fragmentation and Encryption to Enforce Privacy in Data Storage*, in ACM Transactions on Information and System Security, 2009.

[8] Compliance-driven Models, Languages, and Architectures for Services (COMPAS), FP7 http://www.compas-ict.eu/ 2009.

[9] M. Comuzzi, C. Kotsokalis, G. Spanoudkis, R. Yahyapour. *Establishing and Monitoring SLAs in Complex Service Based Systems*, IEEE International Conference on Web Services 2009 Los Angeles, CA.

[10] F. Daniel, F. Casati, V. D'Andrea, S. Strauch, D. Schumm, F. Leymann, E. Mulo U. Zdun, S. Dustdar, S. Sebahi, F. de Marchi, M.S. Hacid. *Business Compliance Governance in Service-Oriented Architectures*. In: Proceedings of the IEEE 23rd International Conference on Advanced Information Networking and Applications (AINA'09), Bradford, United Kingdom, May 26-29, 2009.

[11] F. Leymann. *Cloud Computing*. In: Proc. 52th Photogrammetric Week. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik. S. 1-10,. Online, September 2009.

[12] Google App Engine http://code.google.com/intl/de-AT/appengine

[13] H. M. Frutos, I. Kotsiopoulos. *BREIN: Business Objective Driven Reliable and Intelligent Grids for Real Business*, International Journal of Interoperability in Business Information Systems, Issue 3 (1), 2009.

[14] Kindergatenportal, http://kindergartenportal.o-s.de/elternFN 2009.

[15] B. Koller, L. Schubert. *Towards autonomous SLA management using a proxy-like approach*. Multiagent Grid Syst. 3(3), 2007, IOS Press, Amsterdam, The Netherlands, The Netherlands.

[16] E. Oberortner, U. Zdun, S. Dustdar. *Tailoring a Model-Driven Quality-of-Service DSL for Various Stakeholders*, MiSE'09

[17] Open Cloud manifesto 2009, http://www.opencloudmanifesto.org 2010.

[18] M. Risch, I. Brandic, J. Altmann. Using SLA Mapping to Increase Market Liquidity. NFPSLAM-SOC'09. In conjunction with the 7th International Joint Conference on Service Oriented Computing, November 23-27 2009, Stockholm, Sweden.

[19] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. B.-Y., W. Emmerich, F. Galan. *The RESERVOIR Model and Architecture for Open Federated Cloud Computing*, IBM System Journal Special Edition on Internet Scale Data Centers, to appear.

[20] SLA@SOI Project, http://sla-at-soi.eu/

[21] D. Schleicher, T. Anstett, F. Leymann, R. Mietzner. *Maintaining Compliance in Customizable Process Models*. In: Proceedings of the 17th International Conference on COOPERATIVE INFORMATION SYSTEMS (CoopIS 2009).

[22] L Schubert, K. Jeffery, B. Neidecker-Lutz. The Future of Cloud Computing, Opportunities for European Cloud Computing Beyond 2010, Version 1.0 http://cordis.europa.eu/fp7/ict/ssai/docs/executivesummary-forweb_en.pdf

[23] B. Sotomayor, R. S. Montero, I. M. Llorente, I. Foster. *Capacity Leasing in Cloud Systems using the OpenNebula Engine*. In Proceedings of Cloud Computing and Its Applications (CCA08), 22 – 23 October 2008, Chicago USA.

[24] W. Thielman, R. Yahyapour, J. Butler. *Multi-level SLA Management for Service-Oriented Infrastructures*, Proceedings of the 1st European Conference on Towards a Service-Based Internet, 2008.

[25] Web Service Level Agreement (WSLA), http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[26] VeriSign - Increase Online Sales and Conversions, Proven Results with VeriSign Seals, http://www.verisign.com/trust-seal/increase-online-sales/index.html

[27] M. Völter, Th. Stahl, J. Bettin, A. Haase. Model-Driven Software Development. John Wiley & Sons, 2006.