RC23327 (W0409-030) September 3, 2004
Computer Science

IBM Research Report

Management of SLA Management Data Relationships Using the DIRECT Metadata Management Framework

Christopher Ward, Nianhua Li*, Melissa J. Buco, Rong N. Chang, Laura Z. Luan
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

*Department of Computer Engineering
Purdue University
Indianapolis, IN 46202

Management of SLA Management Data Relationships using the DIRECT Metadata Management Framework

Christopher Ward[†], Nianhua Li[‡], Melissa J. Buco[†], Rong N. Chang[†], Laura Z. Luan[†]

[†] IBM T. J. Watson Research Center, Hawthorne, NY 10532, United States
{cw1,mbuco,rong,luan}@us.ibm.com

[‡] Department of Computer Engineering, Purdue University, Indianapolis, IN 46202, United States
niali@iupui.edu

ABSTRACT

A key challenge in the domain of service level agreement (SLA) management is the effective representation and exploitation of SLA management data which includes the SLA contract data and related internal service management data (e.g., interim service level attainment results). To the best of our knowledge, there is still no commercially available satisfactory means of capturing and managing the relationships between these SLA management data in support of SLA contract performance management. In more general terms, this challenge is best characterized as the representation of a domain specific set of relationships (or "knowledge base") in support of domain specific application oriented queries against that knowledge base. This paper describes the essential relationships in the SLA management domain and presents an extensible relationship management framework which enables effective integration and exploitation of SLA management data (using XML data store technology) in support of proactive management of SLA contract performance. The design philosophy behind the relationship management framework is believed to have broader applicability than only this management domain.

Categories and Subject Descriptors

E.2 [**Data Storage Representations**]: Data Storage Representations – *linked representations.* H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *selection process.*

General Terms

Management, Documentation, Performance, Design.

Keywords

service level agreement, contract data management, service level agreement management, business impact analytics, metadata integration, ontology, semi-structured data store.

1. INTRODUCTION

The importance of proactively managing service qualities adhering to service level agreement (SLA) management has been increasing at a rapid pace over the past few years as its value to successful e-business outsourcing and hosting service engagements is increasingly recognized [1,2]. Yet, despite the importance of this capability, the market intelligence firm Gartner estimates that over 80 percent of the SLAs established by 2003 would be breached in 2003/2004 [3]. One reason for this disparity is the difficulty for SLA management systems to effectively manage and exploit the relationships between SLA contract data and other service management data in support of typical SLA management functions (e.g. complete SLA attainment reporting, SLA related business impact analytics, etc.) as already appreciated in the research community [4,5]. In fact, to date, SLA contract data and its relationships to the internal service level management (SLM) data are still typically managed as an ad hoc collection of paper documents, text files, spreadsheets, database records, and manual processes (such as dispute resolution and SLA compliance reporting).

In an effort to improve both the effectiveness and efficiency of managing these relationships in support of business analytics over the selected domain of discourse (SLA management) we have developed and implemented DIRECT (Data Integration Repository for Executing Contract Terms) an actionable, domain specific, data directed, knowledge management framework that facilitates relationship management by the use of ontology and canonical representations of both data and algorithms [6]. The framework provides:

- Access to SLA management data based on a domain specific ontology structured using a hierarchy of multidimensional characteristics so as to represent and manage a select set of relationships for both template and instance data.
- A common data representation to store and retrieve data from a number of semantically equivalent (but) disparate data sources using ontology.
- A dataflow relationship repository that encompasses two distinguished features: each dataflow step is annotated with ontology data to provide a context for how this relates to other dataflow steps; and each dataflow exposes the data format of the result sets in the common data representation.
- A template based approach that links the creation of new (contract) instances to their offerings based on ontology.

To support these capabilities DIRECT includes a generic data model for capturing SLM-related SLA contract data, a metadata model for integrating SLA data and internal SLM data, and an SLM application enablement component that facilitates exploiting and exploring the managed SLM data through Web Service [7,8] application programmer interfaces.

The remainder of this paper is organized as follows. Section 2 elaborates the nature of the SLM data and business logic that are required to provide contractual service level reports. Section 3 provides an overview of the SLM data integration model in DIRECT, briefly describes how the ontology and XML technologies are used to characterize the SLM data and the business logic for selected business impact analytics (BIA), and exemplifies DIRECT usage with screenshots of from an exploitative GUI. Section 4 summarizes areas of related work. The paper is concluded in Section 5.

2. MOTIVATION – SLM RELATIONSHIPS FOR VIRTUALIZED STORAGE

The relationship between SLA contract data and the internal SLM data can be illustrated via a managed on demand storage service contract that offers the customer a virtual disk service with a service level on the incremental storage provisioning response time (for details see [9]). Such a service requires the provider to manage the mapping between the virtual disk space and the corresponding real storage resources. The SLA contract data in this case comprises all contractually described data attributes associated with the managed virtual disk, including pricing, required capacity, availability service level targets, etc. The corresponding set of internal SLM data includes the data attributes associated with this mapping (e.g. physical storages server names, allocated capacity). While the SLM data and the SLA-SLM data relationships must be managed well by the provider, such non-contractual implementation details need not be exposed to the customer [10,11].

Figure 1 highlights the business logic (which is a subset of the SLA management relationships) for SLA compliance reporting. The gathered raw quality measures must be adjudicated prior to being used as qualified quality measures. The service level evaluation step can be triggered to generate the quality attainment reporting data for a past (completed) service-level evaluation period or for the current evaluation period (intermediate reporting). After making changes to the input (e.g. after resolution of a quality measure dispute) that step as well as the following steps must be re-executed to update the effected service level reports. The figure additionally shows sample SLA and internal SLM data. Each data element is characterized based upon the data category it belongs to (e.g. SLA Contract or Internal Management) and the role it plays in SLA management (e.g. Actual Measurement Data, Measurement Source, etc.). These characterizations provide one way in which the data may be referenced by ontology-based SLM applications.

Figure 2 illustrates our staged approach to capturing and managing SLA and internal SLM data based upon contract life-cycle management activities [12,13]. The SLA contract is established during the Contract Authoring phase and may be represented in a variety of formats (e.g., printed text, tables, arithmetic formulas, electronic documents or formal language specifications). Once signed, the SLA contract data is extracted via a document processing technology (e.g., manual data extraction, text keyword parsing, XSLT extraction [14] or rule based element generation) and stored in the system in a Common Intermediate Representation, independent of how the contracted services are to be fulfilled. The SLA data is then associated with a Fulfillment Solution, which is a multilevel implementation-independent relationship graph of the SLA-related SLM metadata. For example, for a managed storage service contract, the SLA contract data refers to the storage service components and associated service levels, whereas the internal SLM data in the Fulfillment Solution refers to (at level 1) the set of attributes required to compute the contractual or internal service levels for the contract and links them (at level 2) to the set of fulfillment linkage specifications that locate the relevant data and algorithms. Exact details of the Fulfillment Solution are related to the nature of the contracts (e.g., related service offering details) and could incorporate many linkage levels (Level 1 … n in the figure).

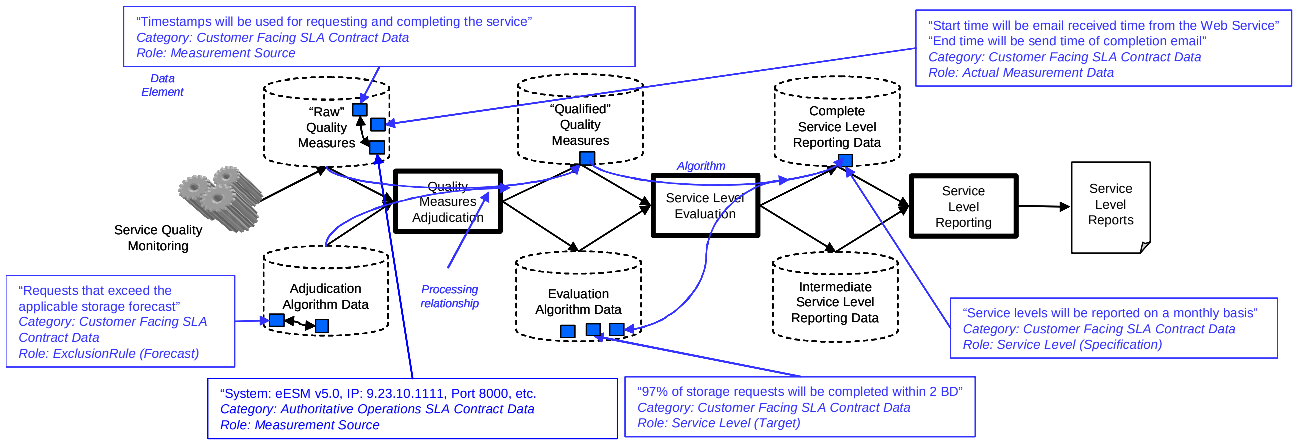

"Timestamps will be used for requesting and completing the service"
Category: Customer Facing SLA Contract Data
Role: Measurement Source
Data Element
"Raw" Quality Measures
"Qualified" Quality Measures
Complete Service Level Reporting Data
"Start time will be email received time from the Web Service"
"End time will be send time of completion email"
Category: Customer Facing SLA Contract Data
Role: Actual Measurement Data
Algorithm
Service Quality Monitoring
Quality Measures Adjudication
Service Level Evaluation
Service Level Reporting
Service Level Reports
"Requests that exceed the applicable storage forecast"
Category: Customer Facing SLA Contract Data
Role: ExclusionRule (Forecast)
Adjudication Algorithm Data
Processing relationship
Evaluation Algorithm Data
Intermediate Service Level Reporting Data
"Service levels will be reported on a monthly basis"
Category: Customer Facing SLA Contract Data
Role: Service Level (Specification)
"System: eESM v5.0, IP: 9.23.10.1111, Port 8000, etc.
Category: Authoritative Operations SLA Contract Data
Role: Measurement Source
"97% of storage requests will be completed within 2 BD"
Category: Customer Facing SLA Contract Data
Role: Service Level (Target)

Figure 1. SLA contract and SLM data and its role in service level reporting.

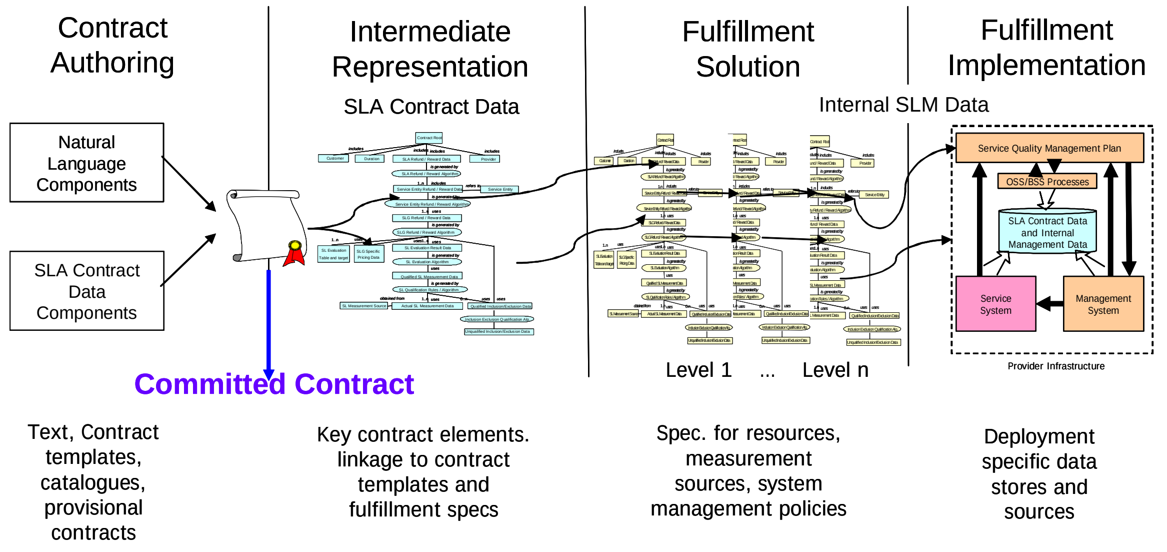

Contract Authoring
Intermediate Representation
SLA Contract Data
Fulfillment Solution
Fulfillment Implementation
Internal SLM Data
Natural Language Components
SLA Contract Data Components
Service Quality Management Plan
OSS/BSS Processes
SLA Contract Data and Internal Management Data
Service System
Management System
Provider Infrastructure
Level 1 ... Level n
Committed Contract
Text, Contract templates, catalogues, provisional contracts
Key contract elements. linkage to contract templates and fulfillment specs
Spec. for resources, measurement sources, system management policies
Deployment specific data stores and sources

Figure 2. Linkage between SLA contract data and other internal SLM data.

Finally, the Fulfillment Implementation (e.g., service delivery environment configurations and other sources of relevant data) stores the actual SLM data in various databases and locales as referenced by the Fulfillment Solution. The semantic models for the Intermediate Representation and Fulfillment Solution identify the essential components of a Contract as described in [12,13]. The text below the icons summarizes example metadata for the stage in question.

Based on observations of the nature of SLA contracts and their associated business logic, the following relationship management requirements have been identified:

- Ontological structure for SLM data including clear delineation and association of SLA contract data.
- SLM data associations within SLA contracts (e.g. data attributes that collectively describe a particular service level).
- SLA relevant computation associations (e.g. the computation steps needed to support SLA compliance reporting and rebate).
- Associations between contract instances and the offering templates from which they were created.
- Associations between the set of algorithms available in the domain and their generalized input/output data type signatures (independent of their use).

Details of the resulting relationship characterization scheme are elaborated further in Section 3.

3. SLM DATA INTEGRATION MODEL

Figure 3 illustrates a data integration model (or a metadata model) for exploiting the relationships between SLA data and internal SLM data. The data integration model enables the representation of a generic ontology for the SLA management domain and can serve as the core data model of a data-driven SLA management system. It enables SLA management domain knowledge sharing and reuse by representing the knowledge in an organized way. Ontology augmented concepts are defined as SLM elements in the model using XML [15,16,17]. Presently DIRECT uses nonstrandards based syntax for ontology representation rather than the evolving web semantics technology [18]. The reason being the maturity of the tools to query and store ontologies balanced with DIRECTs need to query and store “non ontological aspects” of the data using XQUERY [19,20,21]. Since DIRECT aims to manage a large number of contracts it is necessary to have a contract repository that supports: contract querying/reasoning, infrequent updates (presently beyond the scope of XQuery), validation/integrity control, and recovery. Given these requirements our ontology syntax is sufficient to express concepts and relations in the SLM domain.

SLM elements vary in descriptive scope from domain-wide generalized specifications to those applicable within only individual contracts. Although there are no limitations on the number of data scope levels, the current data integration model is organized into three specification levels: SLM domain specification, service offering specification, and contract instance specification, as listed in descending order of abstraction level (Figure 3). Within each SLM element type specification level, individual data/algorithm specifications are defined separately from relationships to improve flexibility and scalability (left side Figure 3).

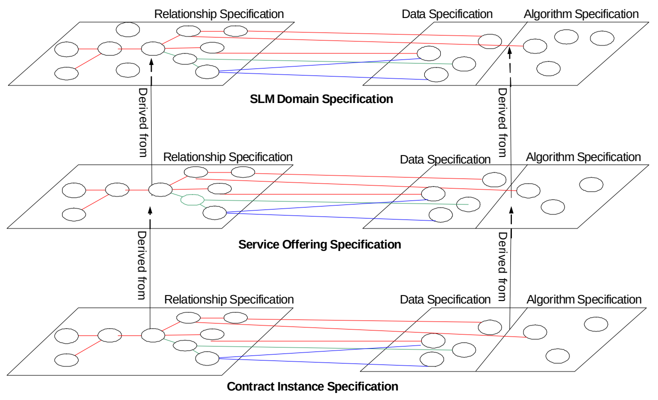

Relationship Specification
Data Specification
Algorithm Specification
SLM Domain Specification
Derived from
Service Offering Specification
Contract Instance Specification

Figure 3. A multi-level data integration model for SLA management data.

The top-level, i.e. the SLM domain specification level, provides a hierarchical classification of SLM domain data. Common roles in SLM tasks are organized into an ontology using the subclass-of relation. For example, the class *Algorithm* is subclassed by *ServiceLevelEvaluation, MeasurementQualification* etc. algorithms, with *ServiceLevelEvaluation* in turn subclassed by *PercentageAvailability, PercentCompletedInTime* etc. Algorithms are referenced by the SLMAlgorithm element which defines the basic features of an SLM algorithm, e.g., an SLM algorithm has input, output and algorithm expression. So, the SLM algorithm for *ServiceLevelEvaluation* inherits all the basic features of the base SLMAlgorithm, and further extends it with features specific to service level evaluation, e.g., the input of an evaluation algorithm is qualified measurement data, whereas the output is service level evaluation data. Likewise *PercentCompletedInTime* inherits all features from *ServiceLevelEvaluation* and further extends it with specific features such as the particular algorithm expression.

The middle-level (i.e. the service offering specification level[1]) exposes “offerings” (i.e. connected sets of elements from the top level) that the provider wishes to make available. The offerings facilitate service level management on the provider side in that it summarizes customer-neutral service offering information, facilitates cross-contract data management tasks on service provider side, and improves knowledge sharing among contracts. Service providers usually offer a set of customer-neutral services based on a common service delivery infrastructure which includes servers, monitoring system, problem management system, account management system, reporting system, etc. Customer-specific customizations are developed based on this common infrastructure to accommodate the customer’s unique data or business process management needs.

The SLM elements in the bottom level, i.e. the contract instance specification, represent individual contract specifications and providers fulfillment implementations by way of instantiation of an offering template from the middle layer. For example, “*The service level attained is the monthly percentage of qualified requests completed within 2 Business Days: ((Total_Qualified OnTime_Requests) / (Total_Qualified_Requests))*100”* is the evaluation algorithm specification which appears in one on demand storage service contract. This can be expressed as part of an offering and is an instance of the top level SLM element *PercentageCompletedInTime*. The instance inherits the generic algorithm expression from *PercentageCompletedInTime*, and then instantiates the algorithm input and output data. It instantiates the customer specific time threshold with value “2 business days”, and instantiates the input qualified measurement data with an appropriate source reference. That SLM element defines the qualified measurement data as “the provider’s response time to customer’s on demand storage provisioning request”.

A second feature of the model are its various relationship specifications. Three of the previously identified types of relationships for SLA management are presently supported: 1) the SLM processing-related relation (element #215) indicates that two fragments of the fulfillment solution need to be connected in order to compute certain SLM results. Based on Section 2 for example, the first step of the “service level evaluation process” is to adjudicate the measurement data. In this step, “actual measurement data” (element #151) has to be analyzed by using an “adjudication algorithm” (element #152). Therefore, their fulfillment solutions should be connected. 2) the schema relation (element #201) indiciates that a fragment of a contract specification is a constituent of a service management specification. For example, “storage provisioning request response time” (element #101) is defined to facilitate the specification of “storage provisioning service level target” (element #201). Finally, 3) the SLA-SLM mapping relation (element #209) indicates one fragment of the fulfillment solution is to be associated with one SLA contract specification fragment. With the above features, the data model can support various SLM tasks, such as complete SLA compliance report data generation, customer centric business impact assessement, service delivery center business impact assessment for autonomic operations management as illustrated in Section 3.2.

Deploying contracts and fulfillment solutions into the data model is technically challenging due to the complex interactions between the SLA, the fullfillment solution, and the fullfillment implementation. The top two specification levels are provided by the model (Figure 3) to address this. Recall, the mid-level (service offering specification) consists of a set of templates for each service offering proposal provided by service provider. They will be used as templates to generate SLM elements in the bottom-level (contract instance specification) when deploying individual contracts into the data model. The templates in the mid-level are, in turn, generated from templates in the top-level (SLM domain specification) which summarize the common data structures of SLA/SLM data and common roles in SLM tasks in an offering independent manner.

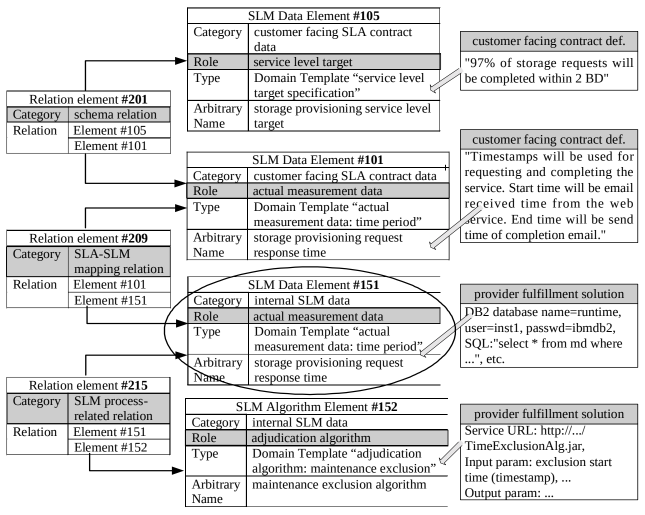

SLM Data Element #105
Category
customer facing SLA contract data
Role
service level target
Type
Domain Template “service level target specification”
Arbitrary Name
storage provisioning service level target
customer facing contract def.
"97% of storage requests will be completed within 2 BD"
Relation element #201
Category
schema relation
Relation
Element #105
Element #101
SLM Data Element #101
Category
customer facing SLA contract data
Role
actual measurement data
Type
Domain Template “actual measurement data: time period”
Arbitrary Name
storage provisioning request response time
customer facing contract def.
"Timestamps will be used for requesting and completing the service. Start time will be email received time from the web service. End time will be send time of completion email."
Relation element #209
Category
SLA-SLM mapping relation
Relation
Element #101
Element #151
SLM Data Element #151
Category
internal SLM data
Role
actual measurement data
Type
Domain Template “actual measurement data: time period”
Arbitrary Name
storage provisioning request response time
provider fulfillment solution
DB2 database name=runtime, user=inst1, passwd=ibmdb2, SQL:"select * from md where ...", etc.
Relation element #215
Category
SLM process-related relation
Relation
Element #151
Element #152
SLM Algorithm Element #152
Category
internal SLM data
Role
adjudication algorithm
Type
Domain Template “adjudication algorithm: maintenance exclusion”
Arbitrary Name
maintenance exclusion algorithm
provider fulfillment solution
Service URL: http://.../ TimeExclusionAlg.jar, Input param: exclusion start time (timestamp), ... Output param: ...

Figure 4. SLM elements in Contract Instance Specification level.

Figure 5 illustrates the hierarchical structure of all three levels of the data model (SLM Domain or SD, Service Offering or SO and Contract Instance or CI) for related sample of SML elements. Brief descriptions are also provided for some of these (SD-1..3,

[1] The term “level” is used with recognition that in the same way that the top “level” included hierarchies, so too do the middle and lower levels. The middle level contains hierarchies of offering (or offering fragment) templates. The lower level contains instances of these templates (as depicted in Figure 2).

SO-1, and CI-1). Each SLM element (e.g. CI-1) is derived from its parent element and classifed based on its category, role and type and (optionally) name. Notice that SLM elements in the real model contains many more elements than are illustrated in Figure 5. For example, the xml encoding of the SLM element #151 in both Figure 4 and 5 is provided in greater detail in Appendix.A.

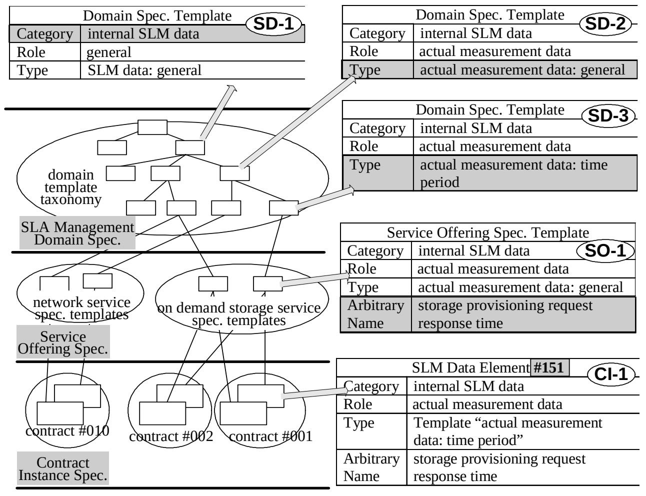

Domain Spec. Template
SD-1
Category | internal SLM data
Role | general
Type | SLM data: general
Domain Spec. Template
SD-2
Category | internal SLM data
Role | actual measurement data
Type | actual measurement data: general
Domain Spec. Template
SD-3
Category | internal SLM data
Role | actual measurement data
Type | actual measurement data: time period
domain template taxonomy
SLA Management Domain Spec.
Service Offering Spec. Template
SO-1
Category | internal SLM data
Role | actual measurement data
Type | actual measurement data: general
Arbitrary Name | storage provisioning request response time
network service spec. templates
on demand storage service spec. templates
Service Offering Spec.
SLM Data Element #151
CI-1
Category | internal SLM data
Role | actual measurement data
Type | Template "actual measurement data: time period"
Arbitrary Name | storage provisioning request response time
contract #010
contract #002
contract #001
Contract Instance Spec.

Figure 5. Hierarchical structure of the data integration model.

Figure 5 should be compared against the illustrative models depicted in Figure 2 (showing the notion of linkage between contractual data and internal SLM data) and the multi-level data integration model of Figure 3.

3.1 Maintaining BIA Relationships

A simple scenario illustrating business impact evaluation using SLM relationships is monthly SLA refund/reward reporting, i.e. the computation of SLA refund/reward for a particular month. The process includes three steps: actual measurement data adjudication, service level evaluation, and SLA refund/reward computing from service level evaluation results.

Regarding measurement data adjudication, the first sub-step is to retrieve raw measurement data and exclusion data (e.g. maintenance log) from external or runtime data stores. Data retrieval methods are specified in the data model by SLM data elements. For example, SLM element #151 illustrated in Figure 4, 5, and A.1 specifies the SQL command of retrieving data from a relational database (the field RetrieveSQL in StructureBinding part). Other access mechanisms include Web Services references (in a manner similar to XVM [22]) and file system references. The data structures of retrieved data are also defined by using XML schema (in StructureDescription part). The second sub-step is to use retrieved data against an executable that implements the adjudication algorithm. The SLM algorithm element describes the implementation aspects of an adjudication algorithm. For example, it specifies the location of the algorithm executable. It also defines the name and data structure of algorithm parameters (i.e. input and output data). A contract instance may involve multiple measurement data with the same type (e.g. both "server down time" and "request response time" are time durations), each is adjudicated by different exclusion data. The association of raw measurement data with exclusion data and adjudication algorithm is neither defined in the adjudication algorithm specification nor in the measurement data specification, but rather specified by the SLM-related process relation described in the last section. All the data/algorithm specifications used in this measurement data adjudication step are referred by a relation element in the data model. There are important advantages to separating data relationships from data specifications. First, it improves sharing and reuse of data/algorithm specifications. Second, it makes the data model flexible. For example, if a customer wants to evaluate business impact based on a customer-defined algorithm (top right in Figure 6) in addition to the authoritative algorithm defined in contract, the customer-defined algorithm will be specified by a new SLM algorithm element. The relation element will then provide references to both authoritative algorithm and customer-defined algorithm. The third sub-step is to save the output of algorithm executable into a data store. The mechanism for saving data is specified in the qualified measurement data specification.

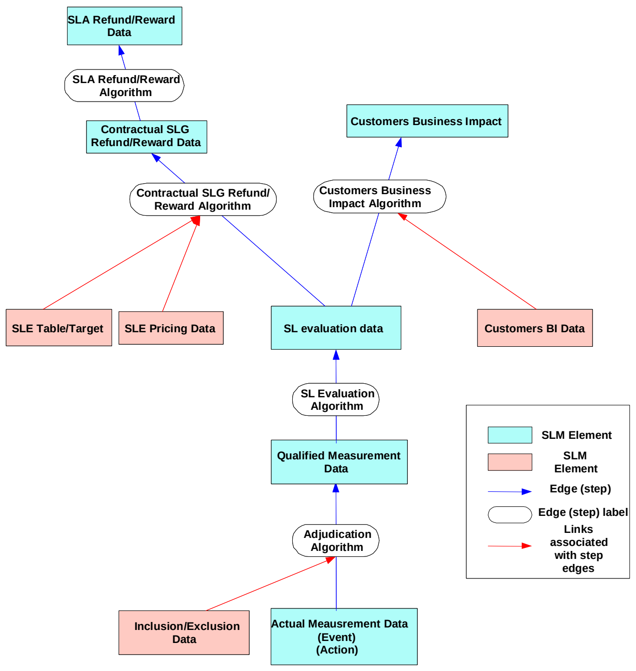

SLA Refund/Reward Data
SLA Refund/Reward Algorithm
Contractual SLG Refund/Reward Data
Customers Business Impact
Contractual SLG Refund/ Reward Algorithm
Customers Business Impact Algorithm
SLE Table/Target
SLE Pricing Data
SL evaluation data
Customers BI Data
SL Evaluation Algorithm
Qualified Measurement Data
Adjudication Algorithm
Inclusion/Exclusion Data
Actual Meausrement Data (Event) (Action)
SLM Element
SLM Element
Edge (step)
Edge (step) label
Links associated with step edges

Figure 6. Evaluation of exposed business impact and the necessary step of service level attainment computation are represented by a ProcessingRelationship element.

The second and third steps of the SLA refund/reward reporting dataflow (service level evaluation and SLA refund/reward evaluation) are processed in a similar manner. Therefore, individual steps are represented by relation elements, which can be shared by multiple SLM dataflows. For example, a non-contractual (internal) service level target can be defined in the service delivery center as to how quickly the service delivery personnel process provisioning requests sent to their email in-boxes. The service level is evaluated across all contracts of "on demand storage service" in this center. This process consists of the same "measurement data adjudication" step as the one described above. The corresponding relation element is reused. Adjudication results are then used for service level evaluation. This sharing and reuse of relation elements among SLM tasks further demonstrates the data model's flexibility.

3.2 Relationship Exploitation

The integration data model for DIRECT is stored as an XML repository.[2] Given this, various SLM application consumers (e.g. the SLA contract processor, which inputs contracts into the system, or the SLA Compliance Report Generator, which takes service level attainment results and generates a report) can exploit the relationships using XML queries (e.g. XQuery). For example, the XML query in Figure 7 retrieves all service level specifications associated with contract #1033.

The query includes six steps :

1. Retrieve the ID ($i) of the SLM data element that specifies Contract #1033.
2. Retrieve the SLM relation element ($a) that lists the ID of SLM data elements directly related to $i. SLM data elements include customer, provider, and service entities specifications.
3. Retrieve the ID ($b) of service entities specification SLM data elements from the ID list in $a.
4. For each ID $b, retrieve the SLM relation element ($c) that lists the ID of SLM data elements directly related to a service entity. SLM data elements include specifications of service levels, service entity refund algorithm, etc.
5. Retrieve the ID ($d) of service level specification SLM data elements from the ID list in $c.
6. Retrieve the service level specifications ($e) from the SLM data element whose ID is $d.

```
for $i in ./SLAM/SLMDataElement
where ($i/Characteristics/Role/text()="ContractRoot"
          and $i/StructureBinding/ContractID/text()="1033")
return <Result>
    {for $a in ./SLAM/SchemaRelation
    where ($i/CentralData/text()={String ($a/@label)})
    return <Contract>
        {for $b in $a/LinkData[@role="ServiceEntity"]
        return <ServiceEntity name="{string($b/@name)}">
            {for $c in ./SLAM/SchemaRelation
            where ($c/CentralData/text()=$b/text())
            return <Group>
                    {for $d in $c/LinkData[@label="ServiceLevel"]
                    return <ServiceLevel name="{string($d/@name)}">
                        {for $e in ./SLAM/SLMDataElement
                        where ($e/@label=$d/text())
                        return <Sepecification>{$e/StructureBinding}</Sepecification>}
                    </ServiceLevel>}
             </Group> }
         </ServiceEntity> }
    </Contract>}
 </Result>
```

Figure 7. Sample query to retrieve all service level specifications associated with contract #1033.

The query finds the service level specifications associated with contract #1033 (Step 2-5) by exploring schema relations. Similar queries can be used to obtain all SLM elements for a given contract, or to obtain the customer facing contract parameters for a particular service level.

Figure 8 illustrates a second query, to extract the SLM-processing relation used for monthly SLA refund/reward reporting, as introduced in Section 3.1. All SLM-processing relationship specifications of contract #1033 can be obtained. The query result consists of a list of SLM relation elements. Recall that each SLM relation element represents a SLM processing step. DIRECT selects a connected subset of these SLM relation elements to generate a SLM processing plan.

```
for $i in ./SLAM/SLMDataElement
where ($i/Characteristics/Role/text()="ContractRoot"
          and $i/StructureBinding/ContractID/text()="1033")
return <Result>
    {for $a in ./SLAM/SchemaRelation
    where ($a/CentralData/text()=$a/@label)
    return <Contract>
        {for $b in $a/LinkRelation[@role="SLM-Processing"]
        return <Group>
            {for $c in ./SLAM/SLM-Processing
            where ($c/@label=$b/text())
            return <SLM-Processing>{$c}</SLM-Processing>}
        </Group>}
     </Contract>}
</Result>
```

Figure 8. Sample query to retrieve all SLM-processing relations associated with contract #1033.

During plan execution, each SLM relation element in the plan is used to obtain fulfillment solution details. The query to extract these details is shown in Figure 9. It first finds a particular SLM relation elements ($i), and then lists the details of all SLM data elements ($a) referred by $i.

```
for $i in ./SLAM/SLM-Processing[@label="R1000"]/ProcessingRelationship
where ($i/seq/text()="20")
return <Result>
    {for $a in ./SLAM/SLMDataElement
     where ($a/@label=$i//Data/text())
     return <Data>{$a}</Data>}
 </Result>
```

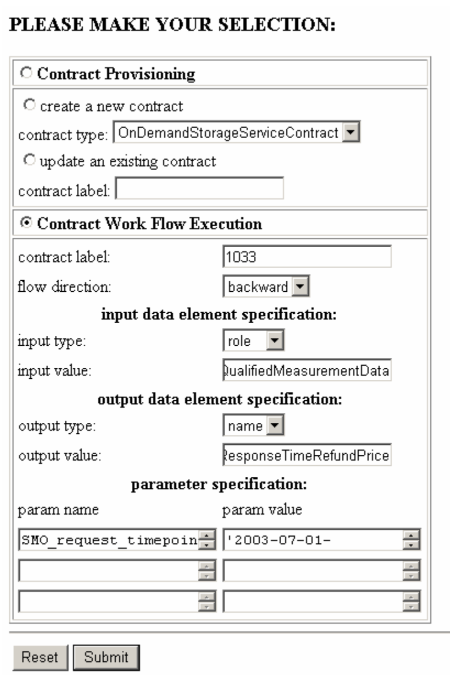
Figure 9. Sample query to retrieve fulfillment solution details referred by a particular SLM relation element.

The data model can also support other queries, such as: queries to obtain the offering template that corresponds to a given contract instance, queries to obtain the "canonical" representation (schema datatype) of a service level evaluation algorithm for a contract instance, and queries to obtain the names of all the contracts that uses a particular application as their service level evaluation algorithm implementation.

3.3 Sample Application

Figure 10 demonstrates (using captured screenshots) a set of computed results for business impact over a range of "what if" scenarios for the month of July, 2003 for our prototype implementation. The left side of Figure 10 shows the GUI used to select end points and required parameters in the computation sub-graph (of the complete set of processing relationships) that are described for a contract instance on the left side of Figure 4. The GUI, which is used for illustrative purposes only (queries would typically be formulated by SLA management application modules) permits the user to specify the data flow traversal direction (in this case "backward" – from the measurement data to the result sets), the starting point (e.g. qualified measurement data) and the ending point (e.g. Contractual SLA Refund Reward).

[2] Sample SLM data elements and relation elements are illustrated in Appendix A.



PLEASE MAKE YOUR SELECTION:

○ Contract Provisioning

○ create a new contract

contract type: OnDemandStorageServiceContract

○ update an existing contract

contract label:

[✓ Contract Work Flow Execution]

contract label:	1033
flow direction:	backward

input data element specification:

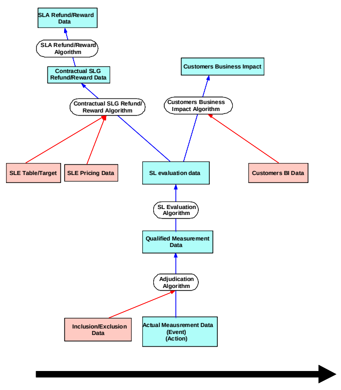
input type:	role
input value:	QualifiedMeasurementData

output data element specification:

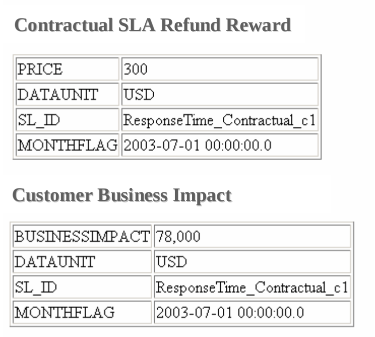
output type:	name
output value:	ResponseTimeRefundPrice

parameter specification:

param name	param value
SMO_request_timepoin	'2003-07-01-


SLA Refund/Reward Data
SLA Refund/Reward Algorithm
Contractual SLG Refund/Reward Data
Customers Business Impact
Contractual SLG Refund/ Reward Algorithm
Customers Business Impact Algorithm
SLE Table/Target
SLE Pricing Data
SL evaluation data
Customers BI Data
SL Evaluation Algorithm
Qualified Measurement Data
Adjudication Algorithm
Inclusion/Exclusion Data
Actual Meausrement Data (Event) (Action)

Data management framework takes actual measurement data, adjudication records, business impact data, etc and generates result sets per user defined processing graphs and query plans.



Contractual SLA Refund Reward

PRICE	300
DATAUNIT	USD
SL_ID	ResponseTime_Contractual_c1
MONTHFLAG	2003-07-01 00:00:00.0

Customer Business Impact

BUSINESSIMPACT	78,000
DATAUNIT	USD
SL_ID	ResponseTime_Contractual_c1
MONTHFLAG	2003-07-01 00:00:00.0

Figure 10. Sample DIRECT GUI screenshots demonstrating the results from data flow sub-graph traversal in support of contract template defined business impact analytics.

The result sets illustrated in the right side of Figure 10 show business impact computations for "Contractual SLA Refund Reward" which is the refund afforded to the customer based on the months performance, and "Customer Business Impact" which provides a customer defined perspective of the impact to their enterprise operations.

The prototype was implemented using Galax [23], the Open Source implementation of XQuery, for its knowledge based (SLM datastore) and DB2 Universal Database V8.1 for its measurement data and runtime data stores. [24] DIRECT is implemented using J2EE V1.3. Because of the nature of the queries received the command processor design pattern [25] is an ideal mechanism for coordinating several concurrent outstanding queries. The Component Configurator pattern likewise facilities the expansion of the service APIs to accommodate new application functional requirements [26]

4. RELATED WORK

In terms of contract management systems in the *e-commerce* domain [27,28] provide multiple features of contract management, including a contract repository, notary, contract monitoring, contract enforcement, contract validation, competence, clarity, legal purpose, and consideration. The system provides a set of contract clause templates to facilitate contract provisioning, and then records signed contract instances and policy documents as XML documents. The effort does not address SLM data management issues *per se*, but rather contract document management with messaging managed via Microsoft's BizTalk tools. Similar to DIRECT, contract specific issues such as legal policies and rules are considered. Policies and rules within contracts are expressed by using clause templates and policy schema. Each contract instance is derived from standard clause templates so that only contract specific parameter values need to be stored.

[29] attempts to automate service level attainment monitoring using a "flexible but precise" XML-based specification. The specification of each service level comprises of two parts: a time constraint and one or more metric evaluator clauses. The clauses specify quality measurement sources, evaluation functions, when the evaluation is to occur, and definitions of input data. WSDL and WSFL are used as part of the service level specification to describe operations, activities and flows of the evaluation process. A SLM engine utilizes the SLA specification to monitor SLA. In particular, the SLM engine creates operators on-the-fly based on WSDL/WSFL specifications of evaluation function. The approach invites some observations. First each type of data and function is defined precisely using XML schema. This ensures the integrity of the service level specification, and enables the information exchange of service level evaluation results. Second, WSDL and WSFL are used to specify operations and process flows. This provides a standard interface between service level specification and (Web Service) fulfillment implementations. However, the specification does not cover all SLA contract related details (e.g. adjudication, SLA refund/reward) that are covered by the DIRECT's semantic data model. Also, the service level specification does not explicitly define the mapping from contractual specification to fulfillment solutions (e.g. the name of the web service used in the fulfillment solution to collect measurement data) as does DIRECT. Also, in their usage scenario the entire sequence from quality measurement to service level evaluation, including non-contractual threshold violation handling, is managed by the SLM engine. This assumption is challenged in real world situations where each part of the process may be handled by different systems (e.g. the service level

evaluator may be required to retrieve measurement data from an external database). Therefore (as in DIRECT) it is necessary to specify fulfillment solutions, as well as their mapping to contractual specifications. Finally, the design of the solution does not address a central design point of DIRECT – that it be a knowledge base (subject to query) for data relationships.

SweetDeal [30] is a DAML+OIL ontology of eBusiness contracts, with a main focus on exception handling. ("Exception" refers to the case when a provider fails to meet contractual commitments, e.g. delivery of an ordered part within three days). SweetDeal is part of the SWEET system (Semantic WEB Enabling Technology), a set of tools that enables communication and inference of e-business rules represented in SCLP (Situated Courteous Logic Programs) RuleML. In SweetDeal, service violation exceptions and exception handling rules are defined using RuleML and embedded in DAML ontology. Therefore the contract representation can be easily interpreted by SWEET system for exception handling. SweetDeal is an ontology rather than "stand alone" formal specification language. It not only provides a way to represent contracts, but also provides taxonomy of contractual exception rules. Again, the SweetDeal ontology does not express all elements of a contract that are handled by DIRECT, as identified in [9] (e.g. SLA refund/reward) and does not provide linkage to a generalized fulfillment implementation.

[31] proposed a multi-level ontology architecture. The upper level core contract ontology defines essential features of all contracts. Each lower-level ontology specifies a particular type of contract (e.g. sale of goods business contract). Lower level ontologies inherit the features defined in the core contract ontology. Unlike DIRECT, the specifications concentrate on representation of contract components without consideration for the fulfillment solution, its linkage to the fulfillment implementation or the business logic required for SLA reporting.

Finally, in view of the nature of the relationships that are being managed (e.g. processing related relationships) there is a natural tendency to compare DIRECT with aspects of Web Services [7] in general and WS-Choreography [32] and BPEL4WS [33] in particular. This is a misguided association. Whereas Web services are self-contained business functions that operate over the Internet, written to strict specifications to work together and with other similar kinds of components, DIRECT is an ontology enriched knowledge base allowing a multiplicity of queries that are not (presently) possible in the WS- framework. DIRECT facilitates access to canonical representations of data and has differing properties (e.g. a knowledge base and location of equivalent data from one of several sources) than the WS-framework.

5. CONCLUSIONS

A new approach to capturing SLA contract data and its associated internal SLM data is needed in light of the complex nature of SLA contract performance management (with the multiplicity of differing service level specifications, contractual quality measure inclusion/exclusion rules, and diverse service level management data requirements demands). This paper describes DIRECT, a data directed knowledge management framework incorporating a data integration model that possesses unique features designed to address SLM related data management challenges. In particular, DIRECT explicitly accommodates:

1. SLA contract data to internal SLM data mapping: The data integration model supports the representation of both SLA contract data and internal SLM data along with a flexible mechanism for characterization and linkage of these data. The characteristics used in this model include role, category, and type.
2. SLA management related business impact analytics representation: The data integration model supports the dynamic dataflow configuration of various SLM tasks (Figure 6) to accommodate contract performance management and related business impact analytics. Dataflow information is represented by a set of processing relationships with respect for the nature of SLA contracts (i.e. SLA contract defined service level evaluations and rebate/reward reporting data computations). The flexible design also facilitates various "what if" analysis such as customer business impact due to poor quality management.
3. Ontology augmented data extraction: For a specific data element the ontology augmented XML representation utilized in our data model provides a capability to extract either the data or the data specification, in support of typical SLA contract management related queries.
4. Template based service offering specifications: The data integration model enables data and relationships management of SLM data that relate to multiple contracts. It also benefits the deployment of individual contracts by providing customer-neutral contract templates.

The results from our prototype are most encouraging in that we are able to accommodate existing SLA refund/reward business impact analysis needs using the design and have been able to fully represent the SLA contract data for the real deployment scenarios we have encountered to date. The framework also readily extends SLA reporting business logic to accommodate new forms of analysis (e.g., internal business impact analysis) without modification to the extensible data management framework yet capitalizing on its strengths.

6. REFERENCES

[1] IDC. *500 Series: eBusiness Service-Level Management Survey.* www.idc.com

[2] IDC. *U.S. Application Management Forecast 2003-2007: Homing in on the Vertical Opportunity.* www.idc.com

[3] Gartner. *Comprehensive SLA's Needed for Hosted Applications.* May 2003. www.gartner.com

[4] Bhoj, P., Singhal S., and Chutani, S., *SLA Management in federated environments.* Computer Networks, 35(2001), 2001, 5-24.

[5] Lewis L. and Ray, P., Service Level Management Definition, Architecture, and Research Challenges. *In Proceedings of the Global Telecommunications Conference (GLOBCOM'99)* 1999, 1974-1978.

[6] Buco, M., Chang, R., Li, N., Luan, L., Ward, C., Integration and Exploitation of SLA Management Data Using the DIRECT Metadata Management Framework. To appear in *Proceedings of the 2nd International Computer, Communications and Control Technologies: CCCT04*, August 14-17, Austin, Texas, USA, 2004.

[7] Chinnici, R., *et al*, (ed) *Web Services Description Language (WSDL) Version 2.0 Part 1 : Core Language*, W3C Working Draft 25 March 2004, http://www.w3.org/TR/2004/WD-wsdl20-20040326/

[8] Mitra, N. (ed), *SOAP Version 1.2 Part 0 : Primer*, W3C Recommendation 24 June 2003, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[9] Ward, C., Buco, M., Chang R. and Luan. L. A Generic SLA Semantic Model for the Execution Management of e-Business Outsourcing Contracts. In *Proceedings of the 3rd International Conference on e-Commerce (EC-Web 2002)*, Aix-en-provence, France, 2002, 363-376.

[10] Frey, N., Matlus R. and Maurer. W., *A Guide to Successful SLA Development and Management.* Gartner Group Research, Strategic Analysis Report, 2000

[11] Hiles. A., *The Complete IT Guide to Service Level Agreements - Matching Service Quality to Business Needs.* Rothstein Associates Inc., Brookfield, Connecticut, USA 1999/2000.

[12] Buco, M., Chang, R., Luan, L., Ward, C., Wolf, J. and Yu, P., Managing eBusiness on Demand SLA Contracts in Business Terms Using the Cross-SLA Execution Manager SAM. In *Proceedings of the Sixth International Symposium on Autonomous Decentralized Systems* (ISADS 2003), Piza, Italy, 2003, 157-164.

[13] Buco, M., Chang, R., Luan, L., Ward, C., Wolf, J. and Yu., P., Utility computing SLA management based upon business objectives. *IBM Systems Journal*, 43(1), 2004, 159-178.

[14] Clark, E. (Ed), *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation 16 November 1999, http://www.w3.org/TR/1999/REC-xslt-19991116

[15] Fallside, D. C. (ed), *XML Schema Part 0 : Primer* W3C Recommendation, 2 May 2001, http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

[16] Bray, T., *et. al.* (ed), *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation 04 February 2004, http://www.w3.org/TR/2004/REC-xml-20040204/

[17] Bray, T., *et. al.* (ed), *Namespaces in XML 1.1.*, W3C Recommendation 4 February 2004, http://www.w3.org/TR/2004/REC-xml-names11-20040204/

[18] Bechhofer, S., *et. al.* (ed), *OWL Web Ontology Language Reference*, WE3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[19] Boag, S., *et. al.* (ed), *XQuery 1.0 : An XML Query Language*, W3C Working Draft 12 November 2003, http://www.w3.org/TR/2003/WD-xquery-20031112/

[20] Draper, D., *et. al.* (ed), *XQuery 1.0 and XPath 2.0 Formal Semantics*, W3C Working Draft 20 February 2004, http://www.w3.org/TR/2004/WD-xquery-semantics-20040220/

[21] Malhotra, A., *et. al.* (ed), *XML Syntax for XQuery 1,0 (XQueryX)*, W3C Working Draft 19 December 2003, http://www.w3.org/TR/2003/WD-xqueryx-20031219/

[22] Li, Q., Kim, M., So, E. and Wood., S., A Bridge between XML Data and its Behavior. In *Proceedings of www2004*, May 17-22, 2004.

[23] Galax. *Galax – an Open Source Implementation of XQuery 1.0.* http://db.bell-labs.com/galax/

[24] IBM. DB2 *Universal Database Version 8.1.* http://www-306.ibm.com/software/data/db2/udb/

[25] Buschmann, F., *et al*, *Pattern-Oriented Software Architecture: A System of Patterns.* John Wiley and Sons, 1996.

[26] Schmidt, D., *et al*, *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects.* John Wiley and Sons, 2000.

[27] Goodchild, A., Herring, C. and Milosevic, Z., Business Contracts for B2B. *In Proceedings of Infrastructures for Dynamic Business-to-Business Service Outsourcing (ISDO 2000)* (Stockholm, 5-6 June, 2000), 2000, pp??

[28] Herring, C. and Milosevic, Z., Implementing B2B Contracts using BizTalk. *In Proceedings of 34th Hawaii International Conference on System Sciences (HICSS 2001)* (Manoa, Hawaii, January 3-6, 2001) , IEEE, New York, NY 10016, 2001, 9080-9090.

[29] Sahai, A., Machiraju, V., Sayal, M., van Moorsel, P. A., and Casati, F., Automated SLA Monitoring for Web Services. *In Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems Operations and Management (DSOM 2002)*, (Montreal, Canada, October 21-23, 2002) Springer-Verlag, Netherlands, 2002, 28-41.

[30] Grosof, B. N. and Poon, T. C.,: “SweetDeal: representing agent contracts with exceptions using XML rules, ontologies, and process descriptions,” *In Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, (Budapest, Hungary, May 20-24, 2003), W3C, 340-349.

[31] Kabilan, V. and Johannesson, P. Semantic Representation of Contract Knowledge using Multi Tier Ontology. *Proceedings of the first International Workshop on Semantic Web and Databases (SWDB* 2003*)*, (Berlin, Germany, September 7-8, 2000), 2000, 395-414.

[32] Burdett, D., *et al* (ed), *WS Choreography Model Overview*, http://www.w3.org/TR/2004/WD-ws-chor-model-20040324/

[33] Andrews, T. el al (ed), *Business Process Execution Language for Web Services Version 1.105.* May 2003, http://www-106.ibm.com/developerworks/library/ws-bpel/

APPENDIX A

Figure A.1 provides a sample SLM Data Element for a particular semantic fragment in the fulfillment solution. The onological information regarding this SLM Data Element is directly representated by the Characteristics. In this case, the element is “Internal SLM Data” and is the “Actual Measurement Data” used in some service level evaluation. The canonical representation (Type) of the resulting data set is “Measurement Data Time Period” and is described by the schema specified in the “Structure Description” (MD_TimePeriod.xsd) element. Access into data resident in the fulfillment implementation is provided through the Stucture Binding element. This element permits the specification of a variety of common data access mechanisms including database and file system access. The Structure Binding also makes provision for data transformation through either scripts, algorithm references or XSL transformations.

```
<SLMDataElement label="151">
   <Characteristics>
      <Name>
         StorageProvisioningRequestResponseTimeMD
      </Name>
      <Type>MD_TimePeriod</Type>
      <Category>InternalSlmData</Category>
      <Role>ActualMeasurementData</Role>
   </Characteristics>
   <StructureDescription>
      <XMLSchema>MD_TimePeriod.xsd</XMLSchema>
   </StructureDescription>
   <StructureBinding>
      <JDBC>
         <Connection>runtime</Connection>
          <UserID>db2inst1</UserID>
          <Password>****</Password>
          <RetrieveSQL>
             SELECT begin, end, sl_n, month
             FROM RequestResponse
             WHERE ( month = ${request_time}) AND
                    sl_n='ResponseTime_c1')
          </RetrieveSQL>
        </JDBC>
   </StructureBinding>
</SLMDataElement>
```

Figure A.1. An illustrative sample of Service Level Management (SLM) data element from the DIRECT metadata store for a particular contract instance.

Figure A.2 provides sample SLM relation elements for Schema Relation, SLM-processing Relation, and SLA-SLM Mapping Relation respectively. These SLM relation elements are used to link SLM data elements for a particular contract instance. SLM data elements are uniquely identified by their label numbers.

```
<SchemaRelation label="R1002">
  <CentralData
          role="ContractRoot" category="SlaData"
          name="OnDemandStorageServiceContract">1033
  </CentralData>
  <LinkData
          role="Customer_Contract_Info" category="SlaData"
          name="OnDemandStorageServiceCustomer">1038
  </LinkData>
  <LinkData
          role="Provider_Contract_Info" category="SlaData"
          name="OnDemandStorageServiceProvider">1039
  </LinkData>
  <LinkData
          role="ServiceEntity" category="SlaData"
          name="FileSystemServiceEntity">1031
  </LinkData>
  <LinkRelation
          role="SLM-Processing"
          name="OnDemandStorageServiceOrchestration">R1000
  </LinkRelation>
  <LinkRelation
          role="SLASLMMapping"
          name="OnDemandStorageServiceSLASLMMapping">R1001
  </LinkRelation>
</SchemaRelation>

<SLM-Processing
          name="OnDemandStorageServiceOrchestration"
          label="R1000">
 <ProcessingRelationship>
  <seq>20</seq>
  <AlgorithmRef>
    <Data role="AdjudicationAlgorithm" type="Authoritative"
          name="MaintenanceExclusionAlgorithm">152
    </Data>
  </AlgorithmRef>
  <InputParameter>
    <Data role="ActualMeasurementData"
          name="StorageProvisioningRequestResponseTime">151
    </Data>
    <Data role="ExclusionData"
          name="ActualMaintenance">153
    </Data>
  </InputParameter>
  <OutputParameter>
    <Data role="QualifiedMeasurementData"
          name="QualifiedResponseTimeMD">154
    </Data>
  </OutputParameter>
</ProcessingRelationship>
...
</SLM-Processing>

<SLASLMMapping
          name="OnDemandStorageServiceSLASLMMapping"
          label="R1001">
  <InterCategoryRelationship role="ActualMeasurementData">
          <Group category="SlaData">
                    <Member
name="StorageProvisioningRequestResponseTime" >101
                    </Member>
          </Group>
          <Group category="InternalSlmData">
                    <Member
name="StorageProvisioningRequestResponseTime" >151
                    </Member>
          </Group>
  </InterCategoryRelationship>
...
</SLASLMMapping>
```

Figure A.2. Illustrative samples of Service Level Management (SLM) relation elements from the DIRECT metadata store for a particular contract instance.