1 Introduction

rSLA is a domain specific language (DSL) for expressing and managing service level agreements (SLAs) in a cloud environment. rSLA is coded in Ruby [**?**], a dynamic scripting language that enables rapid prototyping and application development.

The rSLA DSL is described by an alphabet and by production rules that help to extend the language. The rSLA programming library provides a runtime engine for deploying and running an rSLA service in a cloud environment.

Although the scientific literature provides plentiful results on automated management of SLAs for distributed computing [**?**, **?**, 1], cloud markets hesitate to adopt such solutions. Provisioning of cloud services is handled either manually or with software tools that do not embrace cloud service characteristics.

Cloud service management does not yet support automated and transparent solutions for the management of leased resources. Additionally, there is no established standard yet for the automatic expression and management of SLAs for cloud services.

rSLA provides a DSL library for the definition of rSLA objects and a runtime engine to create and process such objects. The DSL enables the automated generation of customized SLAs and the transparent management of cloud service compliance.

The rSLA is deployed on the IBM Bluemix platform [**?**] as a ruby web service using the sinatra gem[1]. A pilot version of the language is currently running for an IBM financial client. The monthly results from using the rSLA language to evaluate the service level compliance of resources leased by the client, showcase the rSLA DSL adequacy in managing cloud services.

How is the paper structured

-what is the problem that the language solves, motivation to solve this problem -language structure, alphabet, production rules -language runtime -current testing, future testing

2 Problem definition/ motivation

Cloud service management does not yet support enough automated and transparent solutions for the management of leased resources.

There is no established standard so far for the automatic expression and management of SLAs for cloud services.

The rSLA language is not intended to be used only by engineers. An important goal in the language design is to provide a high-level, easy to use and to extend tool that is suitable either for human or machine consumption.

3 rSLA DSL

alphabet, vocabulary, language structure

[1] Sinatra, `http://www.sinatrarb.com/`

production rules

3.1 rSLA language structure, alphabet

point of this subsection?

The rSLA language follows the semantic decomposition of the WSLA specification [1], where an SLA takes the form of a hierarchical tree with a single root node and numerous uni-directional edges. Figure 1 illustrates the rSLA vocabulary as a tree of classes that the rSLA DSL implements. Connections between nodes in the rSLA tree highlight the nesting of SLA context.

In the rSLA alphabet the root node of an rSLA tree represents an SLA object. In Figure 1 nodes that are positioned close to the root, designate branches of SLA context (e.g. base and composite metrics, service level objectives). Edges between nodes are uni-directed to illustrate the rSLA tree hierarchical schema.

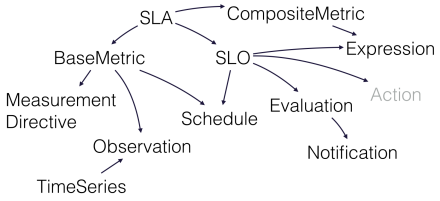

Fig. 1: rSLA DSL class diagram

rSLA supports the creation of programming blocks that express SLA context and that are necessary to run and manage rSLAs in a cloud environment. Listing 1 describes the rSLA vocabulary using set notation to highlight the nesting between language objects:

Listing 1: rSLA vocabulary

```
1 SLA ⊃ { BaseMetric+, CompositeMetric*, SLO+ }
2 BaseMetric ⊃ { MeasurementDirective, Observation*, Schedule* }
3 CompositeMetric ⊃ { Expression* }
4 SLO ⊃ { Expression*, Action*, Evaluation*, Schedule* }
```

In the rSLA language nested relationships denote inclusive associations between objects. For example, an SLA includes base and composite metrics as well as SLOs. Inclusive relationships of rSLA objects do not share same multiplicity rules (listing 1). The rSLA DSL follows the WSLA specification [1] with respect to the definition of rSLA objects and of their basic attributes.

In Figure 1 the *Notification* and *TimeSeries* classes do not appear in the rSLA set representation. These two classes produce objects that help with service level management operations like the statistical analysis of data coming from

monitoring or automated notification reports on scheduled events of service level evaluation. Such two classes are not initially required to build and run SLA instances in a cloud environment, but may be required while one or more SLA management tasks are processed.

conclude that

3.2 rSLA language production rules

rSLA language constructs: elements have relationships/dependencies, there is nesting and management dependencies

production rules

4 rSLA editing, runtime

ready to use functions

The rSLA alphabet consists of language elements that require user input for their creation in an rSLA runtime environment. A DSL user can directly edit code-block scripts in ruby, start a cloud rSLA service and create an active SLA object. In the diagram solid black arrows indicate that user-input is required to create the equivalent objects in an rSLA running environment.

In Figure 1 language elements that require user-input represent tree branches for the creation of service level agreements. DSL user-input requires editing a ruby script to describe the attributes for any new rSLA object.

The rSLA language also contains elements, whose definition with an rSLA service requires that at least one SLA object exists (ex. base metric, slo). A DSL user can edit the context of such elements and associate them with required objects using code blocks in ruby SLA scripts.

5 rSLA deployment

evaluation examples

References

1. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: Web Service Level Agreement (WSLA) Language Specification. Tech. rep., IBM Corporation (Jan 2003)