SLAng: A Language for Defining Service Level Agreements*

D.Davide Lamanna, James Skene and Wolfgang Emmerich
Department of Computer Science (UCL)
{D.Lamanna|J.Skene|W.Emmerich}@cs.ucl.ac.uk

E-business is impeded by technical integration barriers. A large number of industry standards have emerged that support the construction of distributed systems using web services and distributed component technologies. Functional integration may also include the provisioning of infrastructure by one organisation for another, as in the case of Internet Service Provisioning (ISP), Storage Service Provisioning (SSP) and application hosting (ASP). Unfortunately, combining functionality is not the only requirement for e-business integration. Non-functional, quality requirements must also be met. Service Level Agreements (SLA)s capture the mutual responsibilities of the provider of a service and its client with respect to non-functional properties. Our language SLAng provides a format for the description of QoS properties, the means to capture these properties unambiguously for inclusion in SLAs and a language appropriate as input for automated reasoning systems or QoS-aware adaptive middleware.

SLAng defines QoS targets (e.g., performance, availability, reliability, etc.) based on the level of abstraction at which the system is being described (e.g., network, middleware, application level). In order to identify opportunities for SLA between two parties, we have defined a reference model. In our model, applications are clients that use either components or web services to deliver end-user services. Web services may be implemented by invoking components. Components provide an abstraction of the underlying resources. Containers host component instances and are responsible for managing the underlying resource services for communication, persistence, transactions, security and so forth. In addition to tier-specific differentiation, we define Horizontal SLAs, governing the interaction between coordinated peers, and Vertical SLAs between subordinated pairs. SLAng is structured to handle every possible combination of business interactions.

The SLAng syntax is defined using XML Schema. SLAng defines seven different types of SLA. They regulate the possible agreements between the different types of parties identified in our reference model. The Vertical SLAs are *Application* (between applications or web services and components), *Hosting* (between container and component providers), *Persistence* (between a container provider and an SSP) and *Communication* (between container and network service providers). The Horizontal SLAs that parties enter into by composing vertical SLAs are *Service* (between component and web service providers), *Container* (between container providers) and *Networking* (between network providers). For each kind of SLA, a general structure is defined, including client, server and mutual responsibilities.

As a case study, we used the Common Picture eXchange environment (CPXe), an I3A (International Imaging Industry Association) initiative to develop Internet-based digital photo services. CPXe is an architecture that links digital devices, Internet storage and printing, and retail photo finishing together. It takes advantage of Web Services technologies such as SOAP, WSDL and UDDI and supports a large number of scenarios for imaging applications that are distributed across a number of parties. These scenarios we have been analysing put in place collaborations that need to be regulated by SLAs, whenever organisational boundaries are crossed. Our case-study scenarios intends to show that choosing a business partner can be based on choosing the best service level offer.

Using an industrial case study, we have convinced ourselves that SLAng is expressive enough to represent the QoS parameters required for the complete definition of interfaces in multi-party deployments. We have achieved this by exploiting the different abstractions that we have identified in our reference model and by using abstraction-specific parameters for the necessary interfaces. We noted that the SLAs at these different tiers are precise and fairly concise SLA specifications (no SLA was longer than 2 KBytes). We also noted, however, that further work is necessary on the definition of the semantics of SLAng. Right now, the semantics are defined informally, which has turned out to be a weakness. We also intend to test the effectiveness of SLAng for monitoring compliance to SLAs.

*This work is partly funded through the EU IST Project 34069 (TAPAS) and Kodak.