Abstract: There is a growing trend towards enterprise system integration across organizational and enterprise boundaries on the global Internet platform. The Enterprise Service Computing (ESC) has been adopted by more and more corporations to meet the growing demand from businesses and the global economy. However the ESC as a new distributed computing paradigm poses many challenges and issues of quality of services. For example, how is ESC compliant with the quality of service (QoS)? How do service providers guarantee services which meet service consumers’ needs as well as wants? How do both service consumers and service providers agree with QoS at runtime? In this chapter, SLA-Aware enterprise service computing is first introduced as a solution to the challenges and issues of ESC. Then, SLA-Aware ESC is defined as new architectural styles which include SLA-Aware Enterprise Service-Oriented Architecture (ESOA-SLA) and SLA-Aware Enterprise Cloud Service Architecture (ECSA-SLA). In addition, the enterprise architectural styles are specified through our extended ESOA and ECSA models. The ECSA-SLA styles include SLA-Aware cloud services, SLA-Aware cloud service consumers, SLA-Aware cloud SOA infrastructure, SLA-Aware cloud SOA management, SLA-Aware cloud SOA process and SLA-Aware SOA quality attributes. The main advantages of viewing and defining SLA-Aware ESC as an architectural style are (1) abstracting the common structure, constraints and behaviors of a family of ESC systems, such as ECSA-SLA style systems and (2) defining general design principles for the family of enterprise architectures. The design principles of ECSA-SLA systems are proposed based on the model of ECSA-SLA. Finally, we discuss the challenges of SLA-Aware ESC and suggest that the autonomic service computing, automated service computing, adaptive service computing, real-time SOA, and event-driven architecture can help to address the challenges.

Introduction

Enterprise Service Computing (ESC) is a new distributed computing and architectural style that has been adopted by more and more enterprises. ESC primarily includes Enterprise Service-Oriented Architecture (ESOA) [REF-51][REF-53][REF-54] and Enterprise Cloud Service Architecture (ECSA) [REF-55]. Because of complicated business requirements and high customer demands, ESC poses many challenges and issues, such as performance (latency, loss, and jitter) and dependability (security, trust). The Quality of Service (QoS) becomes crucial for ESC to achieve its vision and meet business requirements and customer demands. Nowadays, most enterprises will only invest in IT when there is a clear return on investment, lower total cost of ownership, and a clear demonstration of cost savings. Investments made in services, web services and cloud service initiatives offer the opportunity to realize these requirements, but these investments need to be deployed in a consistent, repeatable, and manageable fashion. Traditional operation management is incapable of offering the unique management functionality that can help achieve these requirements as compared to service-oriented management which is based on QoS.

Service Level Management (SLM) is one of the most important and fundamental service-oriented management. SLM provides mechanisms and tools for managing individual service and the SOA processes composed of a set of services designed to meet enterprises and their customers QoS requirements and demands. The Service Level Agreement (SLA) is a specification of service or service process functional provisioning and non functional goals - QoS which is agreed to by both service providers and service consumers. The Service Level Objectives (SLO) are key elements of SLA, which are specific and measurable quality attributes in the SLA, such as availability, throughput, frequency, performance (response time), and other quality attributes. SLA has been employed in industry such as networking and telecommunication for
several decades. However, adoption of dynamic SLA in ESOA systems is relatively immature and suffers from lack of standards. Recently, cloud computing and ECSA have become the next generation enterprise service computing. The SLA and SLM have become more and more important because of the dynamic service computing environment and infrastructure. Dynamic and automated SLM provides a SLA-Aware approach in ESOA or ECSA architecture. An architectural style is a coordinating set of architectural constraints. The SOA quality attributes are the architectural constraints of ESOA and ECSA. The QoS and SLA can be part of architectural constraints and contracts at the service level in ESOA and ECSA. Therefore, at the architectural style level, adding SLA-Awareness to ESOA or ECSA generates a kind of specific architectural style, which is called SLA-Aware ESOA or SLA-Aware ECSA. At the ESOA and ESCA system (instance) level, the approach allows SLA to play a QoS role between each service consumer and service provider, which greatly improves the service visibility. It also brings service quality control intelligence and capacity into ESOA or ESCA systems, so that it greatly enhances SOA management capabilities. Therefore ESC can meet service or service process functional provisioning and non functional goals &ndash; QoS so that service providers satisfy service consumers with specific services. In addition, enterprises gain revenue from the services and avoid troubles caused by disputed services.

In this chapter, we first discuss the challenges and issues of ESC. Second, we discuss general QoS and SLA concepts, their ontology, standards (such as WS-Agreement), languages (such as WSLA), and classification in enterprise service computing. Third, we define SLA-Aware ESOA and ESCA architectural styles. The styles include:

- **SLA-Aware SOA Quality Attributes**
  
  The SLA-Aware quality attributes are fundamental to the design of SLO and SLA for ESC.

- **SLA-Aware Services**

  The measurable SLA quality attributes are the service constraints of which the service provider is aware in the service at runtime.

- **SLA-Aware service consumers**

  The service consumer is aware of the SLA and can visit it through client-side self-management portal.

- **SLA-Aware service process**

  The SLA-Aware SOA process consists of a set of SLA-Aware services for executing business processes. The SOA process itself is also aware of a process-wide SLA.

- **SLA-Aware SOA infrastructure**

  We define a SLA-Aware SOA infrastructure as a set of SLA-Aware infrastructure services such as SLA-Aware (or QoS-Aware) network services and SLA-Aware storage services.

- **SLA-Aware SOA management**

  SLA-Aware SOA management is defined as a set of SLA-Aware management services which provide SOA system services, including SLA management services, SLA monitoring/measuring services, SLA negotiation services, and SLA reporting services.

- **SLA-Aware Cloud Service Provision and Subscription**

  SLA-Aware cloud service provisioning and subscription will be discussed. The end-to-end SLA-Aware cloud service architectural style is also described
Finally, we discuss the challenges of the SLA-Aware approach in both research and practice including automatic service computing and self-adaptive service computing. In this chapter, we assume all services are web services unless otherwise stated.

**Motivation**

Web services are increasingly adopted by enterprises with the spurt in the growth of e-commerce. The web services can be differentiated by the following standard and dynamic characteristics:

- They can be accessed on the web with the Uniform Resource Locator (URL) and message/document exchange protocol SOAP.
- They are discoverable through the service registry by using standard UDDI.
- They are composable in a standard way. The web services composition can be either static or dynamic.
- They have formal interfaces with their consumers, which are described by standard service language, such as XML-based WSDL.
- They follow basic agreements on listed protocols and standards for communicating and interoperating with each other.

Recent cross-enterprise dynamic services and web service compositions have become reality, such as Amazon's EC2 [REF-1][REF-2] web service cloud. The agreements on standard language and message exchange protocols are not enough for dynamic environment and dynamic service demand; therefore some issues have emerged, which are:

- How will web service providers agree upon what to provide to their service consumers?
- How will web services agree on how good the service is (Quality of Service &ndash; performance, availability, security, etc)?
- Who will complete the required tasks and who will be responsible for failures to execute the tasks?
- How will web services trust each other?

Service Level Agreement (SLA) is a way to address these issues in web service based enterprise architecture. The traditional SLAs between organizations and/or enterprises define the agreements on QoS, including cost and penalty. However, they are mostly static and are not machine-processable, so that the static SLA restricts the dynamic nature of web services in a cross-domain and cross-enterprise environment. Let us consider the following scenario.

A travel reservation service company named TravelRes provides the online service of airlines ticket reservations for travel agents via using web services running in its data center on a SOA infrastructure consisting of multi-tiered clusters with web servers, application servers and databases. The ticket reservation web applications of travel agents are clients of the web services provided by TravelRes data center. Clearly, the performance and availability of the web services are critical for their clients. We assume the QoS guarantees (along with pricing and penalties that are specified in a static SLA) an absolute maximum ticket process response time, such as 40 seconds, and availability, defined as up-time greater than or equal to 99.5% of web services. Moreover, different clients have different guarantee requirements based on their QoS. Since better QoS guarantees require more resources for implementing web services and infrastructure, QoS guarantees are also associated with a number of requests per minute on client-side, such as 1000 requests per minutes. If the number of requests per minutes is greater than 1000 then performance guarantees will
not be given. Finally, the client request demands vary daily and seasonally. To satisfy clients’ different QoS guarantees, different endpoints are given to different clients.

TravelRes builds an enterprise service-oriented data center. The web service cluster connects to a storage area network (SAN) where data is managed. It uses an off-site data center as data backup through a VPN network. A monitoring system watches the web service execution and transactions and checks the compliance with QoS guarantees defined in the SLA. If any of QoS guarantees are not satisfied, the client’s monthly bill will be reduced according to the rules defined in the SLA. Moreover, to minimize resource consumption, all clients’ requests will be routed to workload managers which prioritize requests according to the QoS level. If demand exceeds its cluster capacity, requests with lower penalty are relayed. Therefore, cluster’s capacity and networks are adjusted for profit maximization and not for serving clients’ peak demand.

However, if clients want to increase their web service capacity, they have to call the department of TravelRes and make a request, and then the company needs to purchase the necessary hardware and software to increase the demand of capacity. As a result, TravelRes needs to schedule a configuration change in order to take the additional workload into account. Therefore, increasing capacity demand may take a long time, and impact the business of both clients and service providers. To satisfy clients’ planned demand, TravelRes needs to build a standard interface for its clients in order to automate the additional web service capacity request. Because of increasing market activities or various travel seasons, unpredicted traffic increase is sometimes beyond the current capacity. TravelRes needs to be able to manage a sudden onset of demand at runtime, such that its SOA system should be fully automated in order to reconcile the unplanned demand increase in close to real-time fashion.

From this simple case study, we can see several requirements for both service providers and service consumers. First, performance parameters (response time and throughput) in QoS change with the web services client workload, given a fixed number of allocated resources. If the service provider wants to guarantee a QoS level, it has to foresee its clients’ workload and increase resource dynamically. A viable SLA in a cross-organizational scenario should provide a mechanism for managing clients’ workload requests on demand. Second, service consumers may want to establish SLAs ahead of time in order to ensure that they can get their desired QoS in an SLA. Third, if service consumers require more web service capacity at runtime, they will search multiple service providers to get the best price. Thus, they need to have a mechanism to select better service providers and to get agreements with them. Fourth, to serve short-term capacity requests as shown in the previous example, the service provider needs to support fully automated resource management based on SLA. Finally, service consumers have to monitor their web service activities in order to identify the real service requests and must be capable of delivering their requests to service providers based on contracted capacity in SLA.

To meet the requirements of dynamically managing service capacity from service providers and service consumers we need to establish dynamic SLA mechanisms in a standard and automated way that is integrated with traditional ESOA and ECSA. The mechanism must be SLA-Aware. We define SLA-Awareness as a capacity and a design principle with the machine-processable SLA plus dynamically automated SLA management (SLM) in this chapter. Adding SLA-Awareness to ESOA and ECSA extends the ESOA and ECSA architectural styles. It is a refinement of ESOA and ESCA. We will define them as SLA-Aware ESOA and SLA-Aware ECSA in Section 4. Therefore, treating SLA-Aware ESC architectural style as its refinement is helpful for analyzing and designing higher quality and dynamic ESOA or ECSA systems.

Related Work
A body of research exists related to our work, which can be categorized as follows: (1) SLA standards and
languages: (2) Modeling SLA and QoS; (3) SLA-Aware SOI; (4) SLA Management and SLM; and (5) Adaptive and Automated Computing.

**SLA Frameworks, Standards and Languages**

There are several SLA frameworks, standards and languages for SOA systems based on web services. This section introduces SLA frameworks, standards and languages as well as some related research work.

The Web Service Level Agreement (WSLA) [REF-17, 26, 31] is a specification and reference implementation proposed by IBM. The WSLA provides a framework for specifying and monitoring SLA for web services, which includes:

- A Runtime WSLA architecture, and
- A XML-based WSLA language

*The WS-Agreement* [REF-30] is a specification from the Open Grid Forum (OGF) which provides an agreement protocol between service consumers and service providers. It uses an extensible XML language for specifying the agreement which includes a negotiation constraint. The specification mainly includes three parties:

- A schema for specifying an agreement;
- A schema for specifying agreement templates to facilitate discovery of compatible agreement parties;
- A set of port types and operations for managing agreement life-cycle which includes creation, expiration and monitoring of agreement states.

*The WS-Policy and WS-Policy Attachment* [REF-29] are specifications of service qualities which are part of SLA developed by World Wide Web (W3C). It is often used in conjunction with other web service specifications such as WS-Security policy, WS-ReliableMessage policy, and WS-Transaction policy. The specification is not based on agreement but on service quality requirements.

*The SLAng* [REF-47] is an XML language for defining SLA which is part of the contracts between web service clients and web services. It is developed by the TAPAS project at UCL.

*The Web Service Offering Language (WSOL)* [REF-57, 58] is a formal XML language compatible with the Web Services Description Language (WSDL). While WSDL is used for describing operations provided by web services, WSOL provides a formal specification of multiple classes of service for one web service. The classes of service for a web service are distinguished by different combination of functional provisions and QoS constraints (non-functional requirements [REF-15]), such as response time, simple access right and cost/ performance. It allows service consumers to select different classes of service in depth, or based on cost; therefore, it can be applied to enable service provider’s provisioning models and consumer pay-as-you-go business models.

**Modeling and Formalizing SLA and QoS**

Modeling and formalizing SLA and QoS receives much attention in enterprise service computing research community. Traditional SLA is typically specified by plain-text document such as Amazon’s EC2 Service Level Agreement (http://aws.amazon.com/ec2-sla/). The machine unreadable format could not be used for QoS management and automated negotiation in today’s dynamic and on-demand service computing environment. Enterprise cloud service computing provides a pay-as-you-use business model. Consumers pay for the services and QoS. Without using machine-processable SLA, the service billing system could not automatically
calculate charges when users are using the cloud service. Moreover the service billing system could not automatically reduce the customers’ charges when the system fails or exhibits slower performance. Therefore, much research focuses on specifying SLA and QoS as machine readable and processable languages. Moreover, service-oriented enterprises are hard to manage and it is difficult to monitor quality of their systems, to satisfy their customers, and to reduce service cost. WSLA [REF-31], WS-Agreement [REF-4], SLAng [REF-47] and WSOL [REF-57], introduced in Section 3.1, not only make SLA and QoS machine readable and processable, but also provide formal specifications for system modeling and management. Keller and Ludwig describe a novel WSLA framework for specifying and monitoring SLA for Web services [REF-26]. In addition, Tosic and colleagues developed a management infrastructure to show how WSOL manages web service applications [REF-58].

There is ontology-based SLA and QoS modeling research. Dobson and Sánchez-Macián proposed a unified QoS and SLA ontology [REF-18]. Zhou et al. developed a DAML-QoS ontology [REF-70] to provide better QoS metric models. They proposed a semantic modeling framework for QoS specification [REF-72]. Zhou and Niemela [REF-71] extended OWL-S by including a QoS specification ontology. In addition, they proposed a novel matchmaking algorithm, which is based on the concept of QoS profile compatibility. Fritikos and Plexousakis developed a semantic QoS-based framework for web server description and discovery using OWL-Q [REF-21].

Rigorous formal modeling is helpful towards reasoning the structure and behavior of SLA as well as QoS based systems and investigating the issue of the description of SLA. Meng proposed a QCCS [REF-34] formal model to enforce QoS requirements in service composition based on Milner’s CCS [REF-50]. Nicola et al. defined a process calculus for QoS-Aware applications [REF-36]. Chothia and Kleijn introduced Q-Automata [REF-13] for modeling QoS on trust and other quality attributes, such as availability and response time.

**SLA-Aware Enterprise Service Computing**

SLA-Aware enterprise service computing is receiving attention from many researchers since SLA-Awareness brings software quality management and QoS into enterprise service computing and implements the enterprise non-functional requirements. Zeng et al. proposed a QoS-Aware middleware Agflow [REF-66] for supporting web service composition based on the QoS model they developed. McGough et al. defined an end-to-end workflow pipeline – Workflow Management Service (WFMS) [REF-33] which is a real-time QoS aware workflow management system based on both strict and loose QoS guarantees. The guarantee requirements are defined in an XPath document, which is connected to a BPEL engine. Wada et al. proposed a multiobjective optimization framework E3 for SLA-Aware service composition. SLA-Aware or QoS-Aware approach is also applied to web service selection [REF-29]. The aforementioned work does not include SLA negotiation and dynamic resource scheduling. Brandic et al. presented novel meta-negotiation architecture for SLA-Aware grid services [REF-11]. Song et al. proposed a framework which supports resource scheduling in a virtualization environment for achieving QoS [REF-49].

**SLA Management and SLM**

SLA management and Service Level management (SLM) play important roles in SLA-Aware enterprise service computing. While some research focuses on aspects such as SLA-Aware service composition and workflow, SLA modeling, and specification; there are some research works which emphasize SLA management which addresses end-to-end scenarios across all layers, including internal and external service interfaces, in an enterprise service computing stack. The SLA@SOI consortium published a series of their research works [REF-48] about SLA-Aware Service Oriented Infrastructure (SOI) empowering the service economy in a flexible and dependable way. Their research works include general as well as multi-level SLA management
for SOI [REF-48] and SLA-Aware resource management [REF-16]. The Open group published the SLA Management Handbook [REF-38] from Enterprise Perspective as Volume 4 of a series of SLA management handbooks edited by TeleManagement FORUM. The book is based on a lot of research and practice in SLA management and aims at a true end-to-end SLA. Yeom et al. proposed a contract-based web service QoS management system architecture [REF-64]. Badidi et al. presented a broker-based architecture for web service QoS management (WS-QoS [REF-5]) which is QoS-Aware web service management architecture based on the common concept of brokerage service to mediate between web services providers and consumers. The management operations are executed by the QoS broker. Bhoj et al. described SLA management architecture in the federated environments which share selective management information across administrative boundaries [REF-9]. The SLM focuses on managing SLA commitments at the service level according to the SLA. Figure 1 describes the relationship of Key Quality Indicators (KQI), Key Performance Indicators (KPI), SLA and SLA Monitoring in SLM [REF-38]:

![Figure 1 - Relationship of KQI, KPI, SLA in SLM](image)

Traditional SLM architectures fail to cope with the dynamic runtime nature of enterprise service oriented architecture (ESOA). Schmid and Froeger [REF-45] proposed a decentralized QoS-Management architecture in SOA based on the self-management framework of Service Component Architecture (SCA). Nurmela [REF-37] developed an evaluation framework for SLM in the federated service management context. The SLM not only provides service management for achieving the QoS required by service consumers (enterprise business customers), but also differentiates services [REF-17, 24, 69]. For instance, a web service can be differentiated into Gold, Silver and Bronze service classes based on KQI and KPI, as defined in the SLO and SLA, with the price of service being associated with each of the service classes. This approach provides a dynamic service provisioning framework and is playing an important role in enterprise cloud service computing.
Adaptive and Automated Computing

SLA-Aware enterprise service computing provides a way to allow enterprises to achieve higher quality assurance and cost-effectiveness in their service oriented architecture systems. However, it also brings challenges to distributed service computing in enterprises. The challenges include higher adaptability and automation of enterprise service computing. There is a body of research around the challenges. Yau and An discussed the challenges of adaptive resource allocation for service-based systems [REF-65]. Gao and colleagues presented a QoS analysis technology of adaptive SOA based on a dynamic reconfiguration approach [REF-22]. Wang and colleagues proposed a SLM framework by using QoS monitoring, diagnostics and adaptation for networked enterprise service oriented systems [REF-62, 63]. Self-management [REF-27] and self-adaptive automatic computing [REF-14, 22, 65] are new challenges for today's SLA-Aware enterprise cloud service computing, such as ECSA [REF-55].

References


Service-Oriented Architecture. IEEE International Symposium on Service-Oriented System Engineering (SOSE) (pp. 219-226).


Disclaimer


Jing Dong

Jing Dong received the BS degree in computer science from Peking University and the PhD degree in computer science from the University of Waterloo. He has been on the faculty of Computer Science department at the University of Texas at Dallas and consulting in software industry. His research interests include services computing, formal and automated methods for software engineering, software modeling and design, and visualization. He is a senior member of the IEEE and the ACM.

Contributions

■ SLA-Aware Enterprise Service Computing - Part I

Longji Tang

Longji Tang serves as a Senior Technical Advisor at FedEx’s Information Technology Division where he has acted as a tech lead and/or architect on several critical eCommerce projects. Currently, Longji is the lead project manager for FedEx.com’s Data Center Modernization project. His research focuses on software architecture and design, service-oriented architecture, service-oriented cloud computing and application, and system modeling and formalism. Prior to his tenure with FedEx, Longji worked from 1995-2000 as an Information System and Software Engineering Consultant at Caterpillar and IBM. He has published more than 20 research papers from numeric analysis to computer applications in Journal of Computational Mathematics, Acta Mathematica Scienia and other publications. After graduating from Hunan University with a Bachelor of Engineering degree in Electrical Engineering in 1980, he worked as an associate research fellow at the Hunan Computing Center from 1980 to 1992. He began graduate studies at Penn State University in 1992 and graduated in 1995 with a Master of Engineering degree in Computer Science & Engineering and a Master of Art degree in Applied Mathematics. Longji has undertaken his PhD studies in Software Engineering as a part-time student at the University of Texas at Dallas since June, 2002. He obtained his PhD degree in 2011.

Contributions

■ Modeling and Analyzing Enterprise Cloud Service Architecture - Part I
■ Modeling and Analyzing Enterprise Cloud Service Architecture - Part II
■ SLA-Aware Enterprise Service Computing - Part I
Yajing Zhao

Yajing Zhao received the B.S. degree in computer science from Nankai University in 2005. She received the M.S. degree and Ph.D. degree in software engineering from the University of Texas at Dallas in 2007 and 2010, respectively. She is a software engineer working on real-time multi-threaded systems. Her research interests include software architecture, design patterns, loosely coupled software designs, algorithms and performance, web services, semantic web services, Ontology, cyber physical systems, and real-time systems, network security.

Contributions

- SLA-Aware Enterprise Service Computing - Part I