Dagstuhl Seminar on Service-Oriented Computing

Session Summary "Service Management"

Asit Dan, IBM

Participants of the Core Group

Luciano Baresi, Politecnico di Milano

Asit Dan, IBM (Session Lead)

Martin von Löwis, Hasso-Plattner-Institut

Mike P. Papazoglou, Tilburg University

Thomas Risse, Fraunhofer IPSI

Additional participants:

Heiko Ludwig, IBM TJ Watson Research Center

Monika Kazcmarek, University of Poznan

Introduction

The primary focus of the service management track was to reach a common view on the state-of-the-art in service management, and to identify key challenging issues related to service monitoring, configuration and management.

In an introductory session we reviewed the following aspects:

- What is service management?
- Conceptual framework for service management
- SLAs and Policies: monitoring & enforcement
- Service life-cycle management
  - design, integrate, deploy and runtime management
- Standards for interfaces and information model
  - WSDM: MUWS and MOWS, WS-Agreement, …
- Challenges and focus areas
  - Virtualization and service abstraction

Service management operations deals with two main aspects of IT functions in making a service run smoothly, and delivering desired quality of service: namely **monitoring** (and analyzing) data associated with a service, and **configuring** a service environment & **enforcing** policies associated with a service (in

Dagstuhl Seminar Proceedings 05462
Service Oriented Computing (SOC)
http://drops.dagstuhl.de/opus/volltexte/2006/549

response to monitored data) to make sure smooth execution of the service with desired service quality. By smooth execution we mean the service is up and running, secure and responsive.

Monitoring is performed by Collecting & Analyzing following service usage information:

- How often? how fast? … => for determining Performance level
- By whom? => for determining Security & Auditing
- Status? => for determining Availability, Problem determination
- Service level objective being met? => for determining violation and further analysis on IT operations to be performed in restoring service quality

Service configuration and SLA/Policy enforcement is performed by following:

- Determining resources over which a service is deployed
    - Start/stop, capacity management (IT operations)
- Configuration of middleware services using policies
    - Workload management (classification, policing, routing),
    - Resource allocation
    - Access control
    - recovery (how fast, …)

In addition to runtime management operations, under the service oriented computing paradigm, the scope of service management extends to Service Life-cycle Management operations covering:

- Management of service metadata for life-cycle management stages: Model & design, Integrate/development, deployment and runtime management
    - For discovery and governance
        - Discovery of what a service does? Service interfaces
        - Discovery of non-functional capabilities?
            - Policies on choices of invocation details
            - SLAs that can be created
        - Who can modify or use what existing services?
        - Compatibility of platforms, i.e., what can be deployed together?
        - End-to-end QoS requirements
        - Transformation of platform independent SLA/Policy information into runtime management artifacts

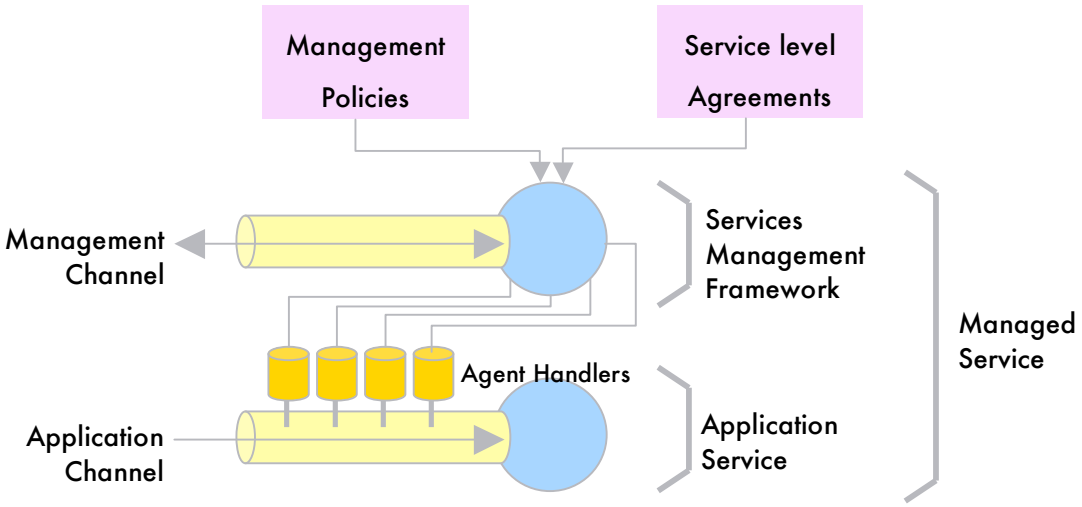

Figure 1: Illustration of service management conceptual framework

Figure 1 illustrates a conceptual framework for service management. The service is invoked by a requester via the normal application channel. The data collection agents are the part of the runtime environment, and collects data either directly being on the execution path, or reading some other status information. The management operations (which can also be services deployed either on the same runtime environment or at a remote environment) receive data collected by the agents for analysis and taking further actions as described above.

Figure 2 provides further details on how a collection of management services for monitoring, analysis, planning and execution (of configuration) can be organized to provide a feedback loop for managing service level. It also illustrates that the managed service environment provides a standards set apis that can be invoked by the management services, and the associated metadata describes the types of information (e.g., metrics) provided by the environment. The interfaces and description of metadata is standardized by OASIS through the Web Services Distributed Management (WSDM) specifications.

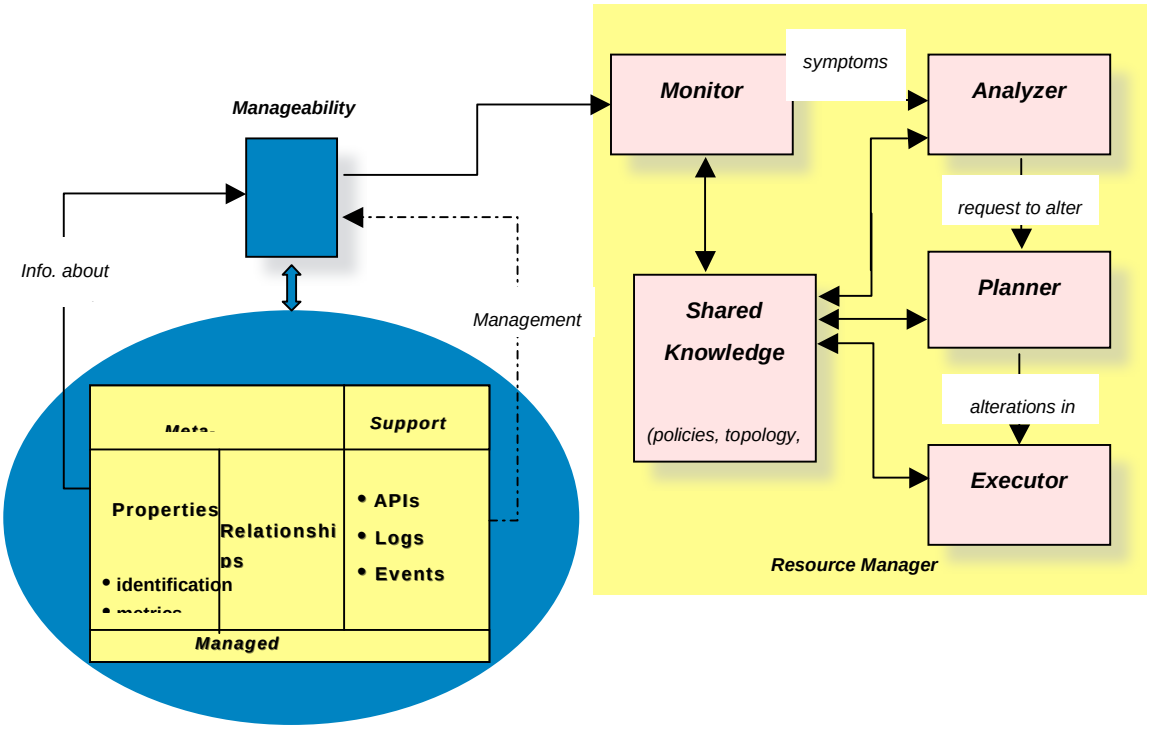

Figure 2: Illustration of further details of service management conceptual framework

Evolution in Management of IT Services

Distributed management of IT systems has evolved over the years, and there are a number of technologies and standards that are currently being used for monitoring and resource management spanning network, storage, processing nodes and software stack (e.g., SNMP, CIM, DMTF Policy, just to name a few). Service oriented computing that aligns IT components with business services and uses open standards in integrating multiple services (as a business process) has lead to new approaches in distributed management. Two key benefits of service oriented computing is resuse of a service (i.e., IT application), and ease of integration of services provided by different service providers using standards based inter-operable interfaces. Like business services, management processes (see Figure 3) of a large scale heterogeneous environment (with components possibly developed by different vendors,) can be developed by integrating reusable management services and/or independent services managing different IT components. There are two key emerging standards in facilitating this. As mentioned earlier, a part of WSDM specification (Management using Web Services or MUWS) defines web services based interfaces and capabilities for all managed environments. Another part of WSDM (Management of Web Services or MOWS) defines management related capabilities and states for managed web (application) services.

Additionally, WS-Agreement defines protocols and schema for establishing (and monitoring) service level agreement between independent IT components, that makes possible to orchestrate multiple independently managed components in achieving end-to-end overall service level objectives. Service level agreement is also an essential concept in virtualizing an underlying collection of heterogeneous resources and hiding the management complexities.

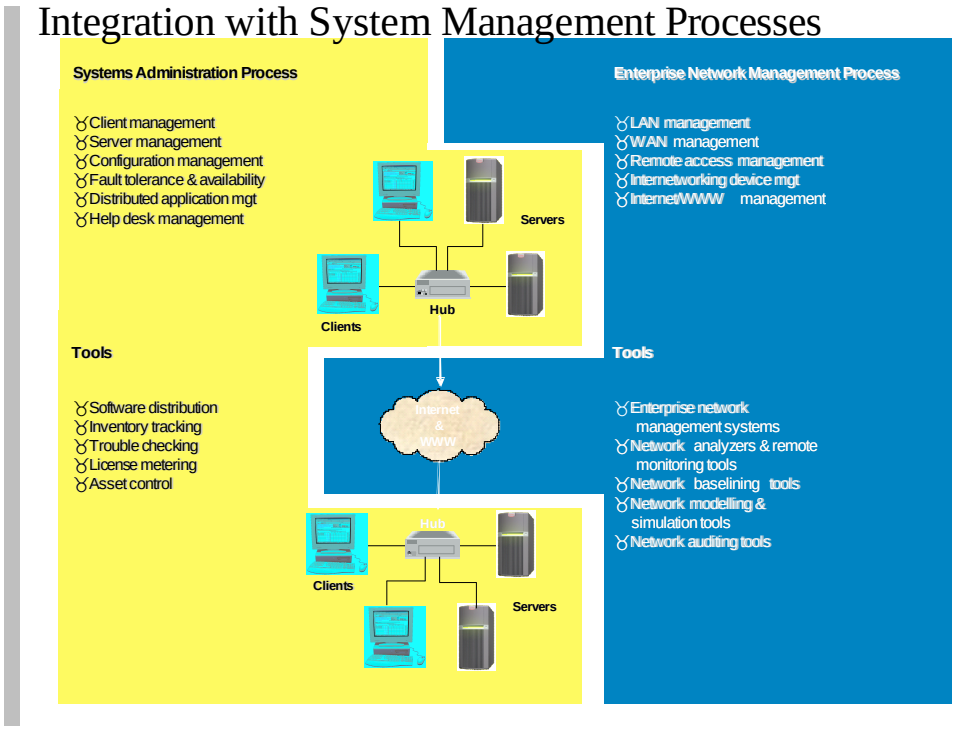

Figure 3: Integration of system management processes

Use of web services for invoking both business services and management services makes it feasible to integrate both business level services and management services in an integrated process (say, when hiring a new a employee includes setting up his/her IT accounts).

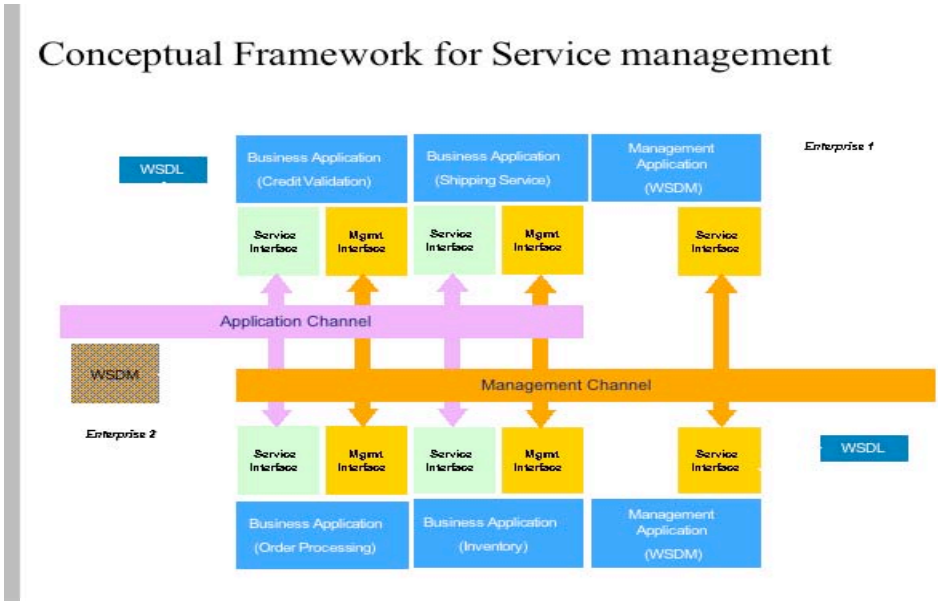

Figure 4: Integration of business and management services

Evolutionary stages in Service Level Management

There are many challenges in managing service level in a complex, heterogeneous, large scale distributed system. Here is a possible roadmap in how the service level management technology/methodology may evolve over time.

- Stage 1: Static Allocation of Resources
  - *Deploy wisely and forget*
  - Technology/Toolset:
    - Capacity planning tool: Analytical or trace based
    - Empirical data with past configurations

- Stage 2: Service Level Monitoring and Manual Re-allocation
  - *Guard against violation of service levels*
  - Technology/Toolset
    - Real time data gathering and analysis
    - Automated deployment of monitoring policies and configuration of data instrumentation
    - Tools for facilitating manual resource allocation: bottleneck visualization, impact analysis/capacity planning, automated re-provisioning

- Stage 3: Simple Adaptive Management of Service Level Objectives
  - *Manage according to runtime artifacts derived from service level policies*
  - Technology/Toolset
    - Discipline and/or middleware specific service level managers: workload manager (e.g., IBM EWLM, IBM Websphere XD), resiliency manager, …
    - Automated derivation of product specific runtime artifacts from service level policies
    - Simple automated re-provisioning

- Stage 4: Business Value/Multi-objective/Pro-active Management
  - *Better management of business objectives*
  - Technology/Toolset
    - Business value analysis and arbitration across multiple types of objectives
    - Integrated SLA life-cycle management – from SLA creation/termination/deployment and runtime arbitration
    - Pro-active/model driven management of business level objectives: model of future workload, model of resource behavior, …
    - Change management in live system meeting service level objectives

Emerging Standards

There are two key aspects in distributed system managements are being standardized in OASIS and in GGF: namely the WSDM and WS-Agreement specifications, respectively.

WSDM: MUWS & MOWS

- **Management Using Web Services (MUWS)**

  http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part2-1.0.pdf

  Specifies

  - Identity
  - Capabilities : events, management operations, …
  - State
  - Resource properties
  - Operational status

- **Management of Web Services (MOWS)**

  http://docs.oasis-open.org/wsdm/2004/12/wsdm-mows-1.0.pdf

  Specifies

  - Metrics – number of requests, number of failed requests, service response time, max response time,…
  - Properties – metrics, …
  - Operational states – upState, downState, busyState, idleState, crashedState, …

WS-Agreement

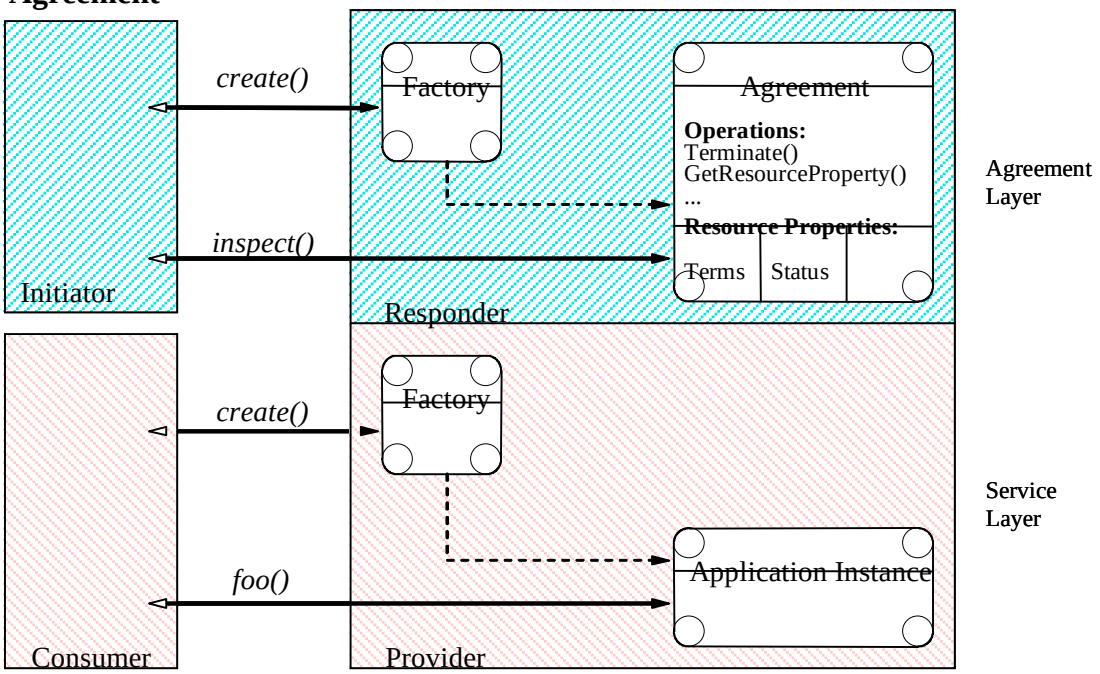

Figure 5: WS-Agreement Conceptual Layered Service Model.

Web Services Agreement Specification (WS-Agreement) defines

  - a Web Services protocol for establishing agreement between two parties, such as between a service provider and consumer, using an extensible XML language for specifying the nature of the agreement,
  - and agreement templates to facilitate discovery of compatible agreement parties.

- The specification consists of three parts which may be used in a composable manner:
  - a schema for specifying an agreement,
  - a schema for specifying an agreement template,
  - and a set of port types and operations for managing agreement life-cycle, including creation, expiration, and monitoring of agreement states.
- Status: 2nd call for public comments. Latest Draft: "WS-AgreementSpecificationDraft.doc": https://forge.gridforum.org/docman2/ViewProperties.php?group_id=71&category_id=659&document_content_id=4560

WS-Agreement specifies a protocol for establishing service level agreement, for querying capability of a service provider and obtaining service level agreement template and for monitoring agreement state. It also defines schema for specifying service level agreement, and agreement templates, starting point for creating an agreement. Figure 5 illustrates use of agreement in establishing client expectations and guarantees by the providers prior to actual invocation of s service. A service is configured by the provider's management system according to the established SLA objectives. During runtime the provider's system monitors the service level objectives (and exposes agreement state using Ws-Agreement protocol) and the service level objectives are enforced.

Need for additional standards

- KPIs (Key Performance Indicators - service level objectives define their targets) and their semantics must be well-defined in order to use in agreements for web services
  - a small set will cover commonly used scenarios
  - applicable to all web services
  - use of agreement, i.e., customization based on client input is important for configuring services
  - Performance (Example KPIs)
    - Average response time (Parameters: averaging window, unit, measurement point)
    - Percentile response time (Parameters: Additionally Percentile target)
    - Throughput (Parameters: Window, unit)
  - Resiliency (Example KPIs)
    - Availability (Parameters: window)
    - Maximum down time (Parameter: unit)
- Monitoring
  - User oriented language to express QoS (non-functional and functional)
    - E.g. to formulate triggers
    - Compliance to business specifications

New Trends and Challenging Issues in Service Management

Use of SOA in all layers of middleware

- Componentization of middleware
  - Example: Resource Allocation service - that uses SLA and associated Business Value in resource allocation
- Resource virtualization
  - Meta-scheduling across multiple job/resource schedulers
  - Application configuration: for both interactive and parallel applications;
    - Pre-configured reusable parallel application instances that can be matched to high level service level objectives can reduce start up latency (provisioning delay)
    - Learning application execution behavior for management - capacity planning, translating into resource requirements, …

Adaptive Data Instrumentation in middleware

- For what service transaction/users to collect data?
  - It‘s need by an external source
  - Overhead in logging/sending messages
- Configurable?
  - Middleware configuration
  - Weaving new process steps during deployment
- Dynamically turning on/off (further optimization)
  - Rule based checking

How to manage & monitor end-to-end QoS?

- Service Monitoring: rather than underlying resource monitoring
  - Use of deployment information in mapping service information
  - Concern on performance
  - Scalability of data collection
- Problem determination and isolation of failing component – what data need to be collected, and what analysis needs to be performed?
  - What to do when there is a problem?
  - Has the alternatives been specified during design time?

Global Policy based orchestration of multiple service level managers

- Automating the deployment process deriving runtime-artifacts across all management systems
  - Transformation of platform independent service level objectives and policies into runtime monitoring and management artifacts

- Orchestration protocol and algorithm across management services
  - Arbitration of service level objectives across users and apps

Acknowledgements

This session summary contains contributions from all participants. In particular, it includes material from presentations by Asit Dan, Luciano Baresi, and several illustrations developed by Mike Papazoglou.