RC23661 (W0507-070) July 8, 2005
Computer Science

IBM Research Report

Managing the End-to-End Lifecycle of Global Service Policies

Daniela Rosu, Asit Dan
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Managing End-to-End Lifecycle of Global Service Policies

Daniela Rosu and Asit Dan

IBM T.J. Watson Research Center, 19, Skyline Drive, Hawthorne, NY, 10532, USA
{drosu, asit}@us.ibm.com

Abstract. Enterprise business services are often deployed over complex environments, managed by multiple service-management products. For instance, a business service may be configured as a three-tier environment with multiple services that run on different resource domains and span one or more tiers, and comprising service-management products such as workload managers, business resiliency managers, and resource arbiters. The objective policies of the enterprise business service, henceforth called Global Service Policies, determine the runtime policies used by the various management products. The lifecycle management of global service policies, including the deployment and enforcement stages, inherits the complexity of the enterprise IT environment. This paper proposes a novel framework for efficiently managing the deployment and enforcement lifecycle stages. The framework enables the complete automation of the dissemination and translation of global policies for all service managers, for a low-cost, correct policy deployment. Also, the framework enables the runtime customization of the resource arbitration based on the business value models of the current global service policies, for a high quality of policy enforcement. The proposed framework is prototyped and integrated with several IBM service-management products.

1. Introduction

In a service-oriented architecture, policies associated with business services define the business objectives under which the services are to be managed. Business objectives may be derived from Service Level Agreements (SLAs) [1, 6, 7] established between provider and its customers. For instance, an SLA regarding a web-based application identifies the types of requests to be issued by the customer and the associated response time and availability objectives.

A typical enterprise business service consists of multiple software components, deployed over a complex environment managed by several independent service-management products. For example, a business service might be deployed as a three-tier configuration in an environment comprising web servers, application servers and data servers (see Figure 1). Sample service-management products in this environment include i) workload managers [15,17] that prioritize and distribute service invocations in order to meet response time and throughput objectives, ii) business resiliency managers [12] that manage the backup nodes, and perform appropriate service reconfiguration in response to node failures such to satisfy recovery time and availability objective, iii) resource arbiters [10, 19] that dynamically change allocation of server nodes across tiers such that the service managers can satisfy their objectives. There may also be products or components monitoring performance or availability of these services, e.g., IBM Tivoli Service Level Advisor [16], IBM Tivoli Omegamon [18], which are configured based on the related SLA. Therefore, multiple components in the enterprise infrastructure are involved in managing the same set of services under a common set of business objectives, referred to as Global Service Policies.

In a SOA, the management of global service policies follows the service lifecycle process, which includes the following stages (1) the modeling of a service including the definition of service and process abstractions, (2) the detailed design and implementation of service components, (3) the deployment of a service on a specific resource configuration, and (4) the runtime execution and management of a deployed service. At the modeling stage, service policies are modeled as the business objectives that can be associated with a service, either as a result of a pre-established SLA with a customer, or based on the creation of a template service offering, that will be customized at deployment stage with customer inputs (in establishing SLAs). To match the SOA principles, service policies must be expressed in a platform independent manner, independent of service deployment details. For instance, the emerging WS-Agreement standard [1] allows us to express a service policy as a four-tuple consisting of a scope that defines associated services or service elements, a qualifying condition (if any) for enforcing a service level objective to be met, a service level objective that expresses the desired goal for a key performance indicator (KPI) and one or more business value assertions expressing the importance of meeting these objectives. At the service deployment stage, the service policies associated with a service are

distributed to each of the components involved in managing this service, and furthermore, transformed into formats expressing runtime information used by each of the components. In the service runtime stage, new service policies are deployed, corresponding to newly established customer SLAs or SLA updates. Also, service policies are enforced by ensuring the related service-level objectives are fulfilled through service manager-specific procedures and enterprise-level resource arbitration.

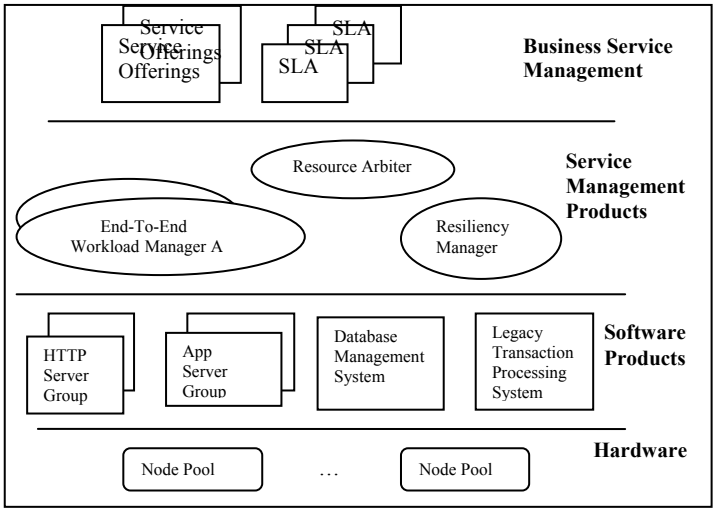

Figure 1 Architecture of Enterprise IT Infrastructure

There are many challenges faced today in managing the lifecycle of global service policies in complex enterprise business services. First and foremost, is the difficulty of setting up multiple management components in a consistent manner, i.e., for managing a common set of business objectives. Setting up these components involves an error-prone, manual process of deriving product specific runtime artifacts. Runtime policies used by each of these components are expressed in a format specific to the component, which includes not just business objectives, but also deployment details such as the domain in which the service is deployed, groupings of service objectives into service classes used for management, and detailed service elements (e.g., ejbs, processes) that the middleware component can manage. Many often the conceptual elements expressed via policies do not match across these components. For example, a workload manager may manage service level objectives associated with a service endpoint, i.e., url or WSDL operation, while a resource arbiter may manage objectives associated with a node or cluster. At best, the administrators repeat common information using component specific tools/GUIs and formats.

Another challenge is the consistent enforcement of Global Service Policies across all service managers. Most prominent is the limitation of resource arbitration products to implement accurately a generic enterprise business value model, as described by business value types (e.g., importance, penalty, reward) and value expressions (e.g., algebraic functions of service KPI values), and by optimization goals (e.g., such as specific to differentiated services, priority-based service, or revenue-based service). Existing enterprise-level arbitration solutions [10, 19] use fixed value models, such as objective importance or service manager priority, and optimization methods suitable for providing differentiated services. For an enterprise using a different business value model, such as one defined by monthly service fees and penalties for objective violations, and optimization of the revenue across all service objectives, the enterprise model must be translated into the arbiter's model, and, most often, this translation results in an approximation of the actual models that leads to inconsistent policy enforcement. Moreover, the enterprise business value models can change in time, due to the evolution of the enterprise business service. Also, the enterprise can use different models for different groups of services, like the customer-facing services and the intranet services. These features require flexible resource arbitration tools that can integrate generic enterprise business value models and can easily adapt as these models evolve.

We submit that supporting an SOA demands novel solutions for the lifecycle management of global service policies. In this paper, we propose a framework for global policy management integrating new solutions for global policy dissemination and for global policy enforcement through enterprise-level resource arbitration. More specifically, we propose a framework for fully automated global policy dissemination and its transformation into the runtime artifacts used by the individual service-management products. The global policy specifications deployed at runtime are filtered and forwarded towards individual service managers based on managers' service scopes and management capabilities (e.g., set of managed services and objective types) Further, manager-specific adapters employ specific service deployment details and policy transformation rules to generate and deploy manager-specific policy specifications and runtime artifacts.

Also, we propose a framework for resource arbitration that can flexibly integrate generic enterprise business service value models. The framework uses the novel "optimization value model" abstraction that describes the relationship between orchestration objectives and the business value models of global policies and has methods that the arbiter can load at runtime and invoke in its decision procedures to assess the values and compare candidate resource allocation states.

A large body of research has addressed the use of SLAs for the management of complex computing environments, composed of Web Services and computational grids. Numerous proposals have focused on issues related to resource management [2, 8] and on protocols for negotiation of agreement terms [3, 4]. The problem of SLA dissemination and transformation has received limited attention. For instance, in the area of heterogeneous public IP networks, the proposal in [9] defines an infrastructure for automated network configuration at multiple management layers based on customer SLAs, where the dissemination is based on network configuration, i.e., service implementation model. Our proposal considers the problem of dissemination of SLAs in which the service specification is decoupled from the deployment details. The problem of resource arbitration has been extensively addressed [10,11,13,14], with main focus on decision methods and monitoring infrastructures. All of these proposals consider a fixed SLA model and optimization goals, while our work starts from the assumption that both SLA and optimization models can change along with the enterprise business service model.

The major contributions of this paper include identification of two of critical limitations of existing global policy management systems, and the proposal of novel solutions that build on the SOA concept of separation between the business service model and the service deployment and implementation details. We have developed a prototype implementation based on the current proposal, which is integrated with IBM service management products in a test environment.

The reminder of this paper is organized as follows. Section 2 presents a sample enterprise business service. Section 3 presents our proposal for a framework for automatic global service policy dissemination and transformation. Section 4 presents our architecture for resource arbitration that can automatically adapt to changes in the global policy value models. Section 5 summarizes our contributions.

2. Sample Enterprise Business Service

A sample Application Service Provider (ASP), running in a complex IT infrastructure as illustrated in Figure 1, provides several computing services, including a Catalog Shopping Web application, an inventory management web service, and a set of batch-mode financial analysis tools. All these services are managed based on SLAs established between the ASP and its customers out of a common pool of physical resources. The goal of the ASP is to appropriately manage these resources such to maximize the outcome across all of the existing SLAs.

The SLAs are developed from templates defined by the ASP's service offerings, and specify the services to be provided, the guarantees to be enforced and the related business values [1]. Appendix A presents a sample SLA in WS-Agreement format defining service objective policies related to a Catalog Shopping service (see Table 1 for namespaces).

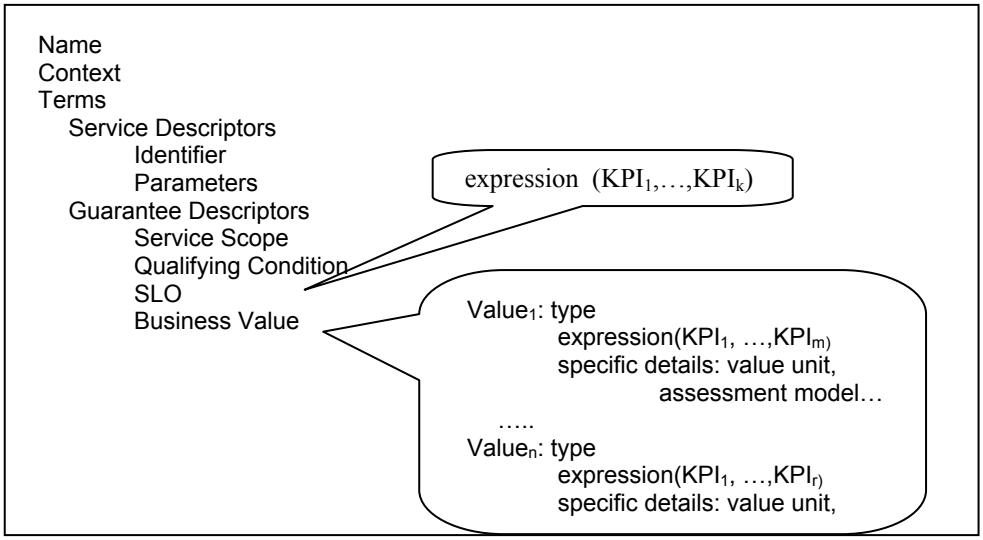

Figure 2 SLA model based on WS-Agreement standard

Figure 2 illustrates an SLA model based on the WS-Agreement standard [1]. More specifically, the SLA comprises (1) a name, which uniquely identifies the SLA in the provider context and can be used by the customer to make related queries or service invocations, (2) a context descriptor, which includes customer and provider elements used in managing the SLA, such as the customer endpoint address, and (3) the agreement terms, which describe the services and related guarantees subject of the agreement. A guarantee term identifies (1) the related service terms, (2) the qualifying conditions, such as time of the day, which must be met in order for the guarantee to be enforced, (3) the Service Level Objective (SLO), which describes the condition to be satisfied by one or more service Key Performance Indicators (KPIs), and (4) the Business Value, which defines value assertions by service clients or providers in meeting the SLO. The Business Value can comprise several value types (e.g., importance and penalty) and their associated KPI expressions.

Table 1 Namespaces used in SLA sample

Namespace Tag	Standard & URL
Wsag	WS-Agreement: http://www.ggf.org/namespaces/ws-agreement
Wsa	WS-Addressing : http://schemas.xmlsoap.org/ws/2003/03/addressing
providerns	Provider-specific namespace for service and objective model
Acel	Autonomic Computing Expression Langauge"[5] http://www.ibm.com/namespaces/autonomic/policy/expressions/1.2

In the sample SLA in Appendix A, the `wsag:ServiceReference` element provides the service identifier, namely `/CatalogShopping`. Further, the two `wsag:GuaranteeTerm` 's define related response time and availability objectives. The business values of the two objectives include a penalty expression, function of specific service KPIs. For instance, the penalty of the response-time objective is a function of the difference between the KPI defining the total number of transactions and the KPI defining the transactions that completed by the target response time. Besides penalty, the business value of the availability objective also includes an importance level (`providerns:RelativeImportance`).

As illustrated in Figure 1, the ASP environment comprises multiple pools of computing nodes. These resources are organized in several three- or two-tier configurations, where each configuration may include a group of HTTP servers, a group of application servers, and a group of database servers or legacy transactional processing systems. Each multi-tier server configuration supports a set of com-

puting services. For instance, a three-tier configuration supports the Catalog Shopping service, which may include several catalog instances, one for each customer buying the service. The inventory management service is mapped to a two-tier configuration.

Several service-management products manage the execution of these services with respect to specific objectives. For instance, the three-tier configuration running the Catalog Shopping application is managed by Workload Manager A, a product such as eWLM [15] or WebSphere Extended Deployment [17], which manages service response-time objectives, by controlling the prioritization and routing of requests across the managed server groups. The Workload Manager B manages the response time for the inventory management service. A business resiliency manager, such as BMC Virtualizer for High Availability [12], manages service availability objectives throughout the entire infrastructure. At a higher level, a resource arbiter, such as IBM Tivoli Intelligent Orchestrator [10] or CoroSoft[11], manages the runtime allocation of resources among the various node groups (tiers and configurations) based on the global service policies defined by SLAs and specific resource-orchestration policies, such as the threshold of value improvement for an acceptable resource reallocation.

The objectives managed by the various service management products derive from global service policies. For instance, the response time objectives managed by Workload Manager A derive from the SLA guarantee descriptors with SLOs expressing response time objectives (see sample in Appendix A).

The process of deriving from SLAs the service manager-specific objectives and their deployment through manager-specific runtime artifacts involves several operations:

- **Filtering the SLA content:** Typically, SLAs include global policies that relate to several service managers. For instance, the sample SLA includes global policies that are related to the management scopes of both the Workload Manager and the Business Resiliency Manager. In order to transform the global policies into a manager's specific objectives, one has to identify and filter those elements of the SLA that are related to the manager's services and objectives.
- **Transformation of SLA content to service-manager runtime artifacts:** Typically, the runtime artifacts used to configure each **s**ervice manager for the provisioning of the related service objectives differ across software components. For instance, eWLM uses a specific XML schema for specification of all of its objectives and related deployment details, while WebSphere uses a specific tool and command syntax to specify the service objectives. Moreover, not all global policy elements can be mapped to service manager abstractions. For instance, the only business value type handled by eWLM and WebSphere is the 'relative importance'; they cannot handle penalty or revenue expressions. Also, they cannot handle objective qualifying conditions describing time intervals. As a result, specialized software components must transform and deploy the global policy objectives as service-manager specific runtime artifacts.
- **Aggregation across multiple SLAs at service-manager level:** The complete set of objectives managed by a service manager derives from the global policies specified in multiple SLAs. For instance, the response time objectives managed by Workload Manager A are related to multiple Catalog Shopping instances, each related to a different customer and defined by a different SLA. This requires that the guarantee descriptors are extracted from all of the SLAs and aggregated into a workload manager-specific descriptor (e.g., an eWLM DomainPolicy file, or a script instantiating all of the WebSphere service classes).

Regarding the resource arbitration process, there are several operations necessary for ensuring that the arbitration complies with the ASP business service value model, as expressed by business value types and expressions, and optimization objectives.

- **Awareness of the global policy objectives and their business value models:** The arbiter must estimate the 'value' of candidate resource reallocation options based on their impact on the actual global policy objectives. The values of these options must be computed based on the expected objective business values, resulted from the evaluation of the objective business value expressions for the service KPI values expected to be achieved after the options are implemented. The arbiter must extract the necessary objective business value details from SLAs.

- **Integration of enterprise-specific optimization objectives/methods:** The arbiter must accurately implement the enterprise business value optimization goals. These goals can vary in time as the enterprise refines its business service model, such as moving from a differentiated-services model to a revenue-penalty model. Also, these goals can be different across groups of services or underlying resource pools. As a result, the arbiter must be able to integrate at runtime new optimizations models and to assess the conditions that determine when given models must be employed.

In the following sections we present our proposals for automated global policy dissemination and adaptive arbitration decision method that enables us to efficiently address the requirements of the sample ASP scenario.

3. Automated Global Policy Dissemination

This section presents the framework for automated global policy dissemination. Building on the SOA approach of separation between the service model and the service deployment and implementation details, the framework comprises (1) components that perform policy filtering and distribution based on a generic global policy model, and (2) components that perform transformation based on the deployment and implementation details.

The central component of the framework is the Policy Disseminator (see Figure 3), which receives global policy specifications produced by business service management components. It filters these specifications for each of the registered service managers and forwards the content to service manager-specific Global Policy Adapters. The adapters translate the global policy specifications to manager-specific runtime artifacts and deploy them to the service managers. The policy disseminator stores the global policy content in order to accommodate runtime registration of new service managers.

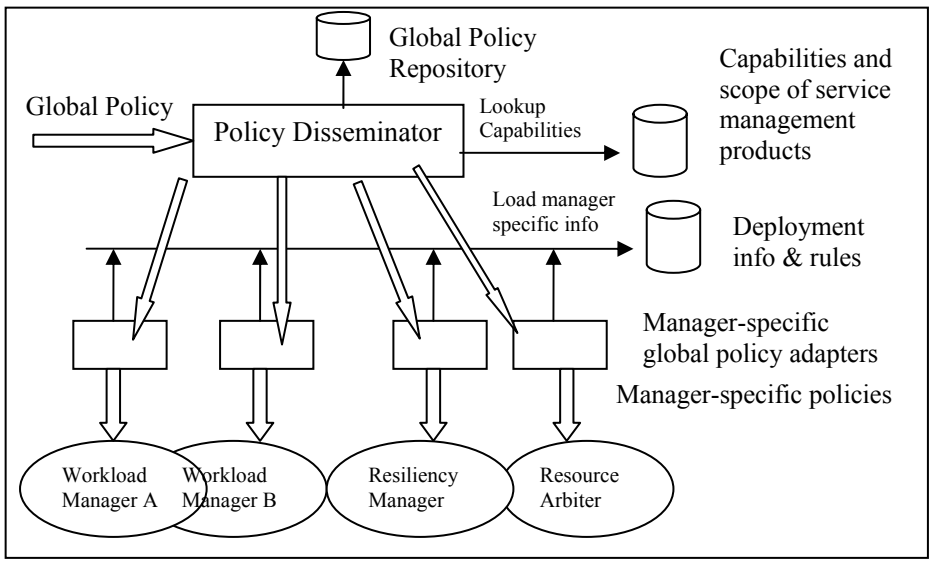

Figure 3 Architecture for global policy dissemination

The framework handles add and discard operations. Policies are added and discarded in groups, and each global policy group is uniquely identified in the infrastructure. A sample type global policy group is derived from a customer SLA and comprises all of the business service objectives specified in that SLA.

The format and the protocol used by the policy disseminator for the forwarded content depend on the format of the input. For instance, for input in WS-Agreement format, the output is also in WS-Agreement format and includes the relevant subset of context and term elements. The output agreement

has a name derived from that of the original agreement, its agreement provider is the corresponding service manager, and its agreement initiator is a designated business service management component, such as the resource arbiter. The content is forwarded via a `WS-AgreementFactory:CreateAgreement` call. Therefore, the documents received by global policy adapters preserve the boundaries of the higher-level global policy documents. These features facilitate the tracing of global policy objectives across the infrastructure.

Global Policy Filters

The global policy filtering is based on service manager capabilities. These capabilities are registered at runtime, and describe the manager's service scope and management objectives. The service scope identifies the set of enterprise services that the manager controls. The management objectives identify the type of SLO that the manager can enforce for a particular service. A combination of service identifier and SLO type should match no more than one service manager. For instance, in the sample configuration described in Section 2, the Workload Manager A manages the `ResponseTimeObjective` of the `CatalogShopping` service (see Appendix A), therefore its service scope includes 'CatalogShopping', and its only management objective is the `ResponseTimeObjective`. The resiliency manager has the same service scope, and its management objectives includes the `AvailabilityObjective`.

In a sample, generic implementation, the capabilities are described by a set of elements, each element corresponding to an SLA component type (e.g., SLO, ServiceIdentifier see Figure 2), and a sequence of pairs of namespace URI and schema element name. For a correct match, for each element in the set, the SLA content corresponding to the type should include a substring of XML elements matching the sequence of namespace/element name pairs. For instance, the capabilities of the Workload Manager A are described by: `{[ServiceIdentifier, ((null, /CatalogShopping))],[SLO,((providerns-URI, ResponseTimeObjective))]}`

In order to perform the filtering operation, the policy disseminator traverses the registered set of manager capabilities and matches them against each global policy item. The matching is performed by global policy interpreters specialized for the specific type of global policy representation. For instance, when handling global policy specified as WS-Agreement (see Appendix A), the policy disseminator uses a specialized WS-Agreement translator. These translators are pluggable components that implement an interface for capability matching and for extraction and composition of the filtered content based on the summary of matches.

The set of registered service manager capabilities can change at runtime by adding or removing service manager to/from the infrastructure or by changing the service scope of an existing manager. Upon such changes, the policy disseminator analyzes all of the global policy groups in its local repository against the new capabilities of the manager, and it identifies and forwards the related updates.

Generation of Service Manager-specific Policies

The transformation of global policies into manager-specific runtime artifacts performed by Global Policy Adapters uses several types of information including (1) manager and service specific deployment information available in databases or configuration files, and (2) manager-specific rules for transformation of global policy abstractions. For instance, for a workload manager like eWLM, the specific deployment information includes the description of the request filters that correspond to a service, and the identification of the type of servers used to run the related requests. Sample transformation rules include the conversion of global policy response time objective model to the eWLM-specific model, and the conversion of the penalty value expression to eWLM importance level.

In the translation process, the adapters aggregate all of the received global policy documents. Also, the adapters handle the global policy elements that the related service managers are not designed to handle. For instance, the eWLM cannot handle time-based qualifying conditions. As a consequence, the eWLM adapter has to track the status of qualifying conditions, and to produce and deploy new eWLM policy any time a status change occurs.

For improved scalability in IT infrastructures with a large number of service management products, the illustrated dissemination architecture can be naturally extended to a multi-level architecture by having global-policy adapters act as policy disseminators for a set of designated managers.

4. Resource Arbitration for Flexible Global Policy Value Model

This section presents the proposed framework for automated, runtime customization of the resource arbitration decision according to the enterprise business service model and global service policies.

The proposal is based on the novel "optimization value model" (OVM) abstraction. An OVM identifies an enterprise objective for optimization of resource allocation, such as "minimize the overall penalty value" or "maximize number of fulfilled objectives, in importance order". OVMs are defined by business service management components as orchestration policies. They are deployed at runtime to the resource arbiter, which uses them to customize the decision method based on the active global service policies. Multiple OVMs may be defined concurrently, each with specific qualifying conditions. The resource arbiter determines which OVM is applicable for a decision instance and uses the associated implementation in the decision process.

Descriptor: signature set, set of business value types and service KPIs ids to be aggregation, and related value units

Method	Function
aggregateServiceForManager	Aggregate all objectives of a service managed by a service manager
aggregateServiceAcrossManagers	Aggregate all manager-level aggregates related to a service
aggregateAcrossServices	Aggregate all service-level aggregates related to analyzed allocation state
compareStateValue	Compare state-level aggregates

Figure 4 Optimization Value Model Interface

Figure 4 illustrates the main components of an OVM. First, the OVM includes a set of methods used for the computation of values for the resource allocation states evaluated by the resource arbiter during its decision process. A state value is computed through the hierarchical aggregation of global policy objective business values and service KPI values into a value predicted for the particular state. The aggregation starts at the service level (i.e., aggregateServiceForManager and aggregateServiceAcrossManagers) by combining the business values of all of the related objectives and possible service KPIs. The types of objective business values (like penalty and importance) and the identifiers of the service KPIs (like as 'distance from goal') used in the aggregation represent the 'signature' of the OVM and are part of its specification. For each signature metric, the OVM specifies a value unit to be used in the aggregation. For instance, for the penalty type, the OVM can indicate the value unit of "USD". Therefore, for the availability policy in Appendix A, the value obtained from expression evaluation has to be converted from "Thousand USD" to "USD" before aggregation.

The next aggregation level is the state level (i.e., aggregateAcrossServices), in which the aggregation combines the service-level aggregate values computed in the previous step. The type of the values produced by the OVM aggregation methods is specific to the OVM implementation. For instance, the value can be a double, or an array of specific data structures. The OVM also includes a method of comparing state-level aggregates (i.e., compareStateValue) and methods for initialization of aggregation at various levels (not shown in the picture).

A sample OVM can implement the optimization objective "minimize overall penalty". The metric set is defined by the penalty value type with "USD" value unit. The aggregation method summates the

per-objective penalties at the corresponding levels, and produces a double value. The comparison method takes two double parameters and indicates as best, the parameter with the lower value (i.e., lower penalty). Alternatively, the OVM can implement the optimization objective "maximize objective compliance in importance order", which supports a guarantee-based service model. The metric set of this OVM is defined by the importance value type and the 'distance from goal' service KPI. The aggregation methods produce arrays with the maximum 'distance from goal' for each importance level. The comparison method takes two array parameters and indicates as best the parameter which has the 'least distance from goal' for the higher importance levels.

The resource arbitration infrastructure that enables the use of OVMs for customization of arbitration decisions is illustrated in Figure 5. The infrastructure comprises the resource arbiter that makes the runtime decisions on how to allocate the available system resources among a set of service manager components, and a resource provisioner that enacts these decisions. Service managers interact with the arbiter through specific global policy adapters, which provide service KPI values and status on how global policy SLOs are satisfied.

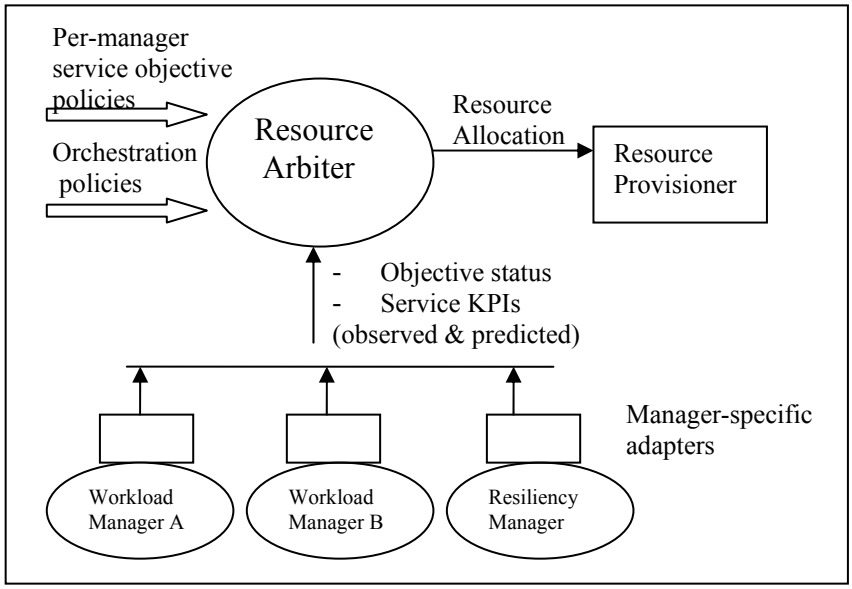

Figure 5 Architecture for resource orchestration based on global policies

The resource arbiter receives at runtime several types of input:

- global policy objective specifications deployed in the system, provided by policy disseminator;
- notification of objective activation updates, provided by service manager-specific adapters. Objective activation status can change when the objective starts or stops to be enforced (i.e., becomes active or inactive, respectively) because of the related qualifying conditions;
- orchestration policies (i.e., global policies related to resource arbitration), from the policy disseminator. Orchestration policies define OVMs to be used in arbitration decisions, possible constraints to be enforced on candidate resource allocation solutions (e.g., number of servers assigned to a particular service or manager), or/and the specification of the decision triggering events (e.g., period, or length of the time interval with critical objective states).

Based on this information, the arbiter customizes its decision method (see Figure 6). Namely, for each resource pool subject to resource arbitration, the arbiter identifies the active global policies, and determines the common set of business value types across all these objectives. For instance, for the sample in Appendix A, the common type is penalty. Further, using the orchestration policy, the arbiter identifies the OVM corresponding to the set of common value types, and loads its implementation from

a library of models. This configuration process incurs minimal overhead because it must be performed only upon changes of the set of active policies or deployment of new OVM specifications.

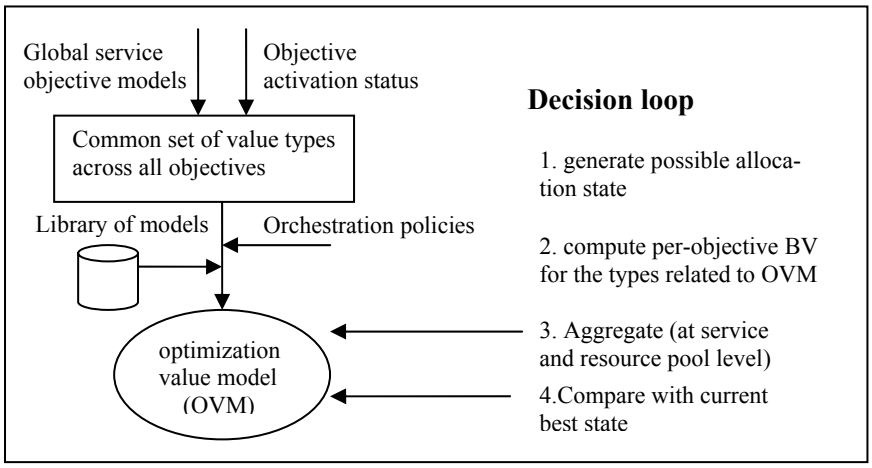

Figure 6 Model of arbitration decision with configurable OVM

The OVM is integrated in the decision process as follows (see Figure 6, left side). Upon generating a new candidate resource allocation state, for all global policies, the arbiter evaluates the value expressions of the business value types in the OVM signature. These values are computed by interpreting the expressions extracted from global policy specifications, and using the service KPI values predicted by service managers or their adapters for the particular allocation. Next, the arbiter uses the OVM methods to aggregate these objective values into an allocation state value. Finally, OVM methods are used to compare the values of candidate allocation states to determine the one best fit for deployment. For improved scalability, partial aggregation of objective value can be performed by global policy adapters.

5. Summary and Conclusions

This paper addresses two problems related to the lifecycle management of global service policies. One is the translation of global service policy updates into the actual service manager runtime artifacts, and the second is the enforcement of global service policies in enterprise-level resource arbitration.

First, we propose a framework for global policy dissemination that enables us to fully automate the process of dissemination of the relevant policies for each service manager, and the process of translating these policies into manger-specific policy specification and runtime artifacts. By enabling the automated configuration of the service management infrastructure in response to global service policy updates, the proposal helps improve the productivity and reduce the occurrence of configuration errors. Also, the proposal addresses several critical requirements, including:

- **support for runtime reconfiguration of the service-management layer:** The runtime registration of service manager capabilities enables us to accommodate runtime changes of the dissemination architecture, such as adding or removing service managers, or of services from their service scopes. As a result, the automated policy dissemination integrates well with the overall system management of the IT infrastructure.
- **support for generic global policy representation:** The framework enables the use of multiple schemas for global policy representation, and can be easily extended to use of new schemas. This facilitates the integration of enterprises created through mergers and acquisitions, and the staged transition of business service management from one global policy model to another.

Second, we propose a new framework for resource arbitration that enables the enforcement of global policy objectives in compliance with the actual enterprise business service value models. The resource

arbiter uses the global service policy specifications to determine at runtime the appropriate customization of its decision method in response to changes of active service objectives and business value optimization goals. Also, the proposal addresses several critical requirements, including:

- **low cost and rapid integration of business service model updates:** The proposal enables us to integrate these updates with no need for major upgrades of the arbitration and service management products.
- **business management control over the quality of arbitration decisions:** The proposal enables the business service management to control the two main elements that define an arbitration decision, the objective business value model and the method for evaluation of solution candidates. We submit that this feature increases the trust in the quality of arbitration decision, and fosters the acceptance of automated resource arbitration.

The prototype implementation integrated with the IBM Tivoli Intelligent Orchestrator, eWLM and WebSphere Extended Deployment demonstrates the feasibility of our proposals.

Acknowledgements: The authors would like thank and acknowledge the contributions of Mircea Avram, Andrew Trossman on policy based arbitration, and Donna Dillenberger, Allen Gilbert, David Kaminsky, Robert Kearney, Mark Linehan, Heiko Ludwig, C. J. Paul, Nataraj Ragantanam, Bala Rajaraman, John Rofrano and John Sweitzer on policy life-cycle management

References

1. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, M. Xu: Web Services Agreement Specification. Version 1.1, Draft 18, submitted to the Global Grid Forum, May 14, 2004.
2. R. Buyya, D. Abramson, J. Giddy, H. Stockinger: Economic models for resource management and scheduling in grid computing. *The Journal of Concurrency and Computation: Practice and Experience*, 14(13-15), pp. 1507–1542, 2002.
3. K. Czajkowski, I. Foster, C. Kesselman, V. Sander, S. Tuecke: SNAP: A Protocol for Negotiation of Service Level Agreements and Coordinated Resource Management in Distributed Systems. *Job Scheduling Strategies for Parallel Processing: 8th International Workshop (JSSPP 2002)*. Edinburgh, 2002.
4. H. Gimpel, H. Ludwig, A. Dan, B. Kearney: PANDA: Specifying Policies for Automated Negotiations of Service Contracts. *Service Oriented Computing – Proceedings of ICSOC 03*, Springer LNCS 2910, pp. 287-302, 2003
5. IBM Corporation: PMAC Expression Language Users Guide. Alphaworks PMAC distribution, www.alphaworks.ibm.com, 2005.
6. H. Ludwig, A. Keller, A. Dan, R. King: A Service Level Agreement Language for Dynamic Electronic Services. *Proceedings of WECWIS 2002*, Newport Beach, 2002.
7. A. Sahai, A. Durante, V. Machiraju: Towards Automated SLA Management for Web Services. *Hewlett-Packard Research Report HPL-2001-310 (R.1)*. Palo Alto, 2002.
8. K. Appleby et al: Oceano – SLA based management of a computing utility, Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management. (2001)
9. K. Fukuda et al: Policy-Based Networking Service over Heterogeneous Public IP Networks (DynaServ), Proceedings of Policy Conference, 1999.
10. IBM Tivoli Intelligent Orchestrator, http://www-306.ibm.com/software/tivoli/products/intell-orch.
11. CoroSoft Technologies: http://www.corosoft.com/
12. BMC Software : http://www.bmc.com/products/proddocview /0,2832,19052_19429_31452409_124990,00 .html
13. W. Walsh et al: Utility Functions in Autonomic Systems, Proceedings of ICAC'2004
14. R. Levy et al: Performance management for cluster based web services, Proceedings of IM 2003.
15. searchdomino.com: IBM Enterprise Workload Manager: http://searchdomino.techtarget.com/whitepaperPage/0,293857,sid4_ gci1012290,00.html
16. IBM Tivoli Service Level Advisor: http://www-306.ibm.com/software/tivoli/products/service-level-advisor/

17. IBM Corporation: WebSphere Extended Deployment, http://www-306.ibm.com/software/webservers/appserv/extend/
18. IBM Tivoli OMEGAMON: http://www-306.ibm.com/software/tivoli/features/candle/
19. SYCHRON: OnDemand Policy Manager. A Look Inside; http://www.sychron.com/downloads.shtml

Appendix A: Sample WS-Agreement-based SLA

```
<wsag:AgreementOffer …>
  <wsag:Name>SLA-x34rt05</wsag:Name>
  <wsag:Context>
     <wsag:AgreementInitiator><acel:StringConstant><Value>ABC Inc.</Value>…
     <wsag:AgreementProvider><acel:StringConstant><Value>ASP Inc.</Value>…
   </wsag:Context>
  <wsag:Terms> ..
     <wsag:ServiceReference wsag:Name="Service0Ref" wsag:ServiceName="Catalog">
        <wsa:EndpointReference><wsa:Address>/CatalogShopping</wsa:Address>…
     </wsag:ServiceReference>
     <wsag:GuaranteeTerm wsag:Name="Goal-Performance">
      <wsag:ServiceScope wsag:ServiceName="Catalog" />
      <wsag:QualifyingCondition><providerns:PeriodName>Primetime</PeriodName>...
       <wsag:ServiceLevelObjective>
          <providerns:ResponseTimeObjective>
               <TimeSecs>2.0</TimeSecs>  <Percentile>98</Percentile>
          </providerns:ResponseTimeObjective>
       </wsag:ServiceLevelObjective>
       <wsag:BusinessValueList>
          <wsag:Penalty>
            <wsag:AssessmentInterval> <wsag:Count>1</wsag:Count>…
           <wsag:ValueUnit>USD</wsag:ValueUnit>
            <wsag:ValueExpression>
                <acel:Product>
                   <acel:Minus>
                      <acel:PropertySensor name="providerns:TransactionCnt" />
                      <acel:PropertySensor name="providerns:OnTimeTransCnt"/>
                   </acel:Minus>
                    <acel:FloatConstant><Value>1.00</Value></acel:FloatConstant>
                </acel:Product> …
      </wsag:BusinessValueList>
  </wsag:GuaranteeTerm>
  <wsag:GuaranteeTerm wsag:Name="Goal-Availability">
    <wsag:ServiceScope wsag:ServiceName="Catalog" />
    <wsag:QualifyingCondition />
    <wsag:ServiceLevelObjective>
          <providerns:AvailabilityObjective>
               <AccumulationIntervalDays>365</AccumulationIn…
               <PercentageAvailability>99.99</PercentageAvailability>
          </providerns:AvailabilityObjective>…
    <wsag:BusinessValueList>
       <wsag:Penalty>
         <wsag:ValueUnit>Thousand USD</wsag:ValueUnit>
         <wsag:ValueExpression>
            <acel:Product>
               <acel:Max>
                <acel:FloatConstant><Value>0</Value></acel:FloatConstant>
                 <acel:Minus>
                     <acel:PropertySensor name="providerns:Downtime" />
                     <acel:PropertySensor name= "providerns: DowntimeObjective"/>
                   </acel:Minus>
              </acel:Max>
              <acel:FloatConstant><Value>1000.00</Value></acel:FloatConstant>
            </acel:Product>
        </wsag:ValueExpression>
       </wsag:Penalty>
       <wsag:CustomBusinessValue><providerns:RelativeImportance>High</providerns..
      </wsag:BusinessValueList> …
```