# A Novel Architecture for Utility Driven Management

Issam Aib[1,2], Raouf Boutaba[1], and Guy Pujolle[2]

[1] Networks and Distributed Systems Laboratory,
University of Waterloo, Canada,
iaib@bcr2.uwaterloo.ca
[2] Lip6 Laboratory, University of Paris 6, France

**Abstract.** In this paper, we specify and implement a framework for utility driven generation and scheduling of management actions based on Business context and Service Level Agreements (SLAs). SLAs are compiled into low level management policies; as well as sets of performance metrics and utility functions. These are subsequently used to drive the scheduling of the low level policy actions. Each action is associated with a utility participation value based on parameters relevant to the contract(s) it is related to; as well as the run-time context of its triggering and execution times. A Web hosting company case study is used to illustrate the benefit of taking into account business level implications when scheduling the execution of management tasks. We measure the overall business profitability as a pondered linear function of other business metrics such as overall raw financial profit and overall customer satisfaction. Finally, we discuss the difficulties and challenges related to the correct estimation of utility costs associated with the low level management/control actions.

**Key words:** Utility Driven Management, Service Level Agreement, Policy based Management

## 1 Introduction

With the increasing number, complexity and frequency of IT related decisions, the mechanisms to determine the optimal utilization of IT resources must become an integral part of automated IT Operations. Given the timescales involved, the decision making process has to be implemented through management rules (policies) whose objective is to maximize the business profit (value) of the services offered by the IT system.

We propose a management approach [2] which stems from the observation that however successful an enterprize might be with its adoption of a management solution, it must be remembered that its IT infrastructure is aimed at the provision of a service which is exchanged for economic value. Therefore, it is extremely important to make the low-level management capability clearly

aware of business level considerations. Low-level configuration and management actions are either triggered by events and system states or manually issued by a system administrator. We consider the use of policies to model this behavior reasonable as it does not lure out the generality of our approach. The reason is that any automatic action or rule can always fit within the Event Condition Action (ECA) paradigm. Hence, we model the dynamics of an IT system through a dynamic set of ECA rules that govern all of its behavior. Manually enforced actions make no exception as they fall too within the ECA rule set where the event part is something like "admin-decision".

In our framework low-level configuration and management policies are generated from the set of contracts (SLAs) the IT provider has concluded as well as his own Business objectives and high level management policies. Our study does not concern itself with how these low-level configuration and management actions are actually generated. We are rather interested in how to monetize and maximize the business level utility at run-time through the appropriate scheduling of low level actions. Each action is associated with a utility (or penalty if negative) participation value based on parameters related to the contract(s) it is related to and the run-time context of its triggering and execution times. We compare this new business-level utility-driven scheduling to the default mode which is the simple FIFO execution of actions as soon as they are triggered. We show that our technique would always result in a better performance and more optimized value for the overall Business Profit of the IT provider.

## 2 Related Work

Driving IT management from business objectives is quite a novel proposition. In [6][7], Buco et. al. present a business-objectives-based utility computing SLA management system. The business objective(s) that they consider is the minimization of the exposed business impact of service level violation, for which a high-level solution is presented in [15]. However, the Management by Business Objectives component of the Framework presented in this paper goes far beyond just using impact of service level violations. It provides a comprehensive method for IT management that can take into account strategic business objectives; thereby, going a long way towards the much needed synchronization of IT and business objectives. For a more detailed discussion of MBO capability applied to the incident management domain see [5].

In another respect, the area of SLA-driven management has been closely interested in the subject of SLA modeling. WSLA [10][11] from IBM research and WSMN [12][13] from HP Labs analyze and define SLAs for Web Services by building new constructs over existing Web Services formalisms. [13] specifies SLOs within SLAs and relates each SLO to a set of Clauses. Clauses provide the exact details on the expected service performance. Each clause represents an event-triggered function over a measured item which evaluates an SLO and triggers an action in the case the SLO has not been respected. In [4], an FSA

(Finite State Automata) is defined for SLA state management in which each state specifies the set of SLA clauses that are active. Transitions between states can be either events generated by an SLA monitoring layer or actions taken by parties in the SLA.

Keller A.and Ludwig H. [10][11] define the Web Service Level Agreement (WSLA) Language for the Specification and Monitoring of SLAs for Web Services. The framework provides differentiated levels of Web services to different customers on the basis of SLAs. In this work, an SLA is defined as a bilateral contract made up of two signatory parties, a Customer and a Provider. Service provider and service customer are ultimately responsible for all obligations, mainly in the case of the service provider, and the ultimate beneficiary of obligations. WSLA defines an SLO as a commitment to maintain a particular state of the service in a given period. An action guarantee performs a particular activity if a given precondition is met. Action guarantees are used as a means to meet SLOs. [9] adds on this work by proposing an approach of using CIM for the SLA-driven management of distributed systems. It proposes a mapping of SLAs, defined using the WSLA framework, onto the CIM information model. Finally, [8] considers a direct application of WSLA within UCEs.

The GSLA model we propose for SLA specification [1] has the novelty of considering each contracted service relationship as a set of parties playing an SLA game in which each party plays one or more roles to achieve the SLA objectives. GSLA party behavior is captured into a unique semantic component; modeling a role that the party plays. SLOs are specified for each role and enforcement policies are generated to meet them. These policies need not be specified at contract sign time, they can change according to run-time circumstances. Ultimately, roles represent a high-level representation of a set of low-level enforcement policies which are generated, enabled, disabled, and removed as a whole and help keep a consistent relationship between what is high-level behavior and its corresponding low-level actions.

Finally, the use of policies for the management of utility computing infrastructures has been recently addressed by Akhil et al. [14] from HP Labs where policy is used to assist in service deployment. We consider this component as part of the policy deployment and resource configuration component of the PDP.

## 3 The Business Driven Management Framework (BDMF)

The main objective of the business driven management (BDMF) framework is to drive the management of IT resources and services from the business point of view. Most of the times, when tradeoff-kind of decisions are to be made, the IT managers have a feeling for which is the option available to them that guarantees the minimum cost or least disruption to the service. But unless the impact of carrying out the chosen course of action onto the business layer is understood, one may run the risk of solving the wrong problem optimally. Because of this,

the BDMF was designed according to the principle of making information that pertains to the business visible from the IT and vice versa.

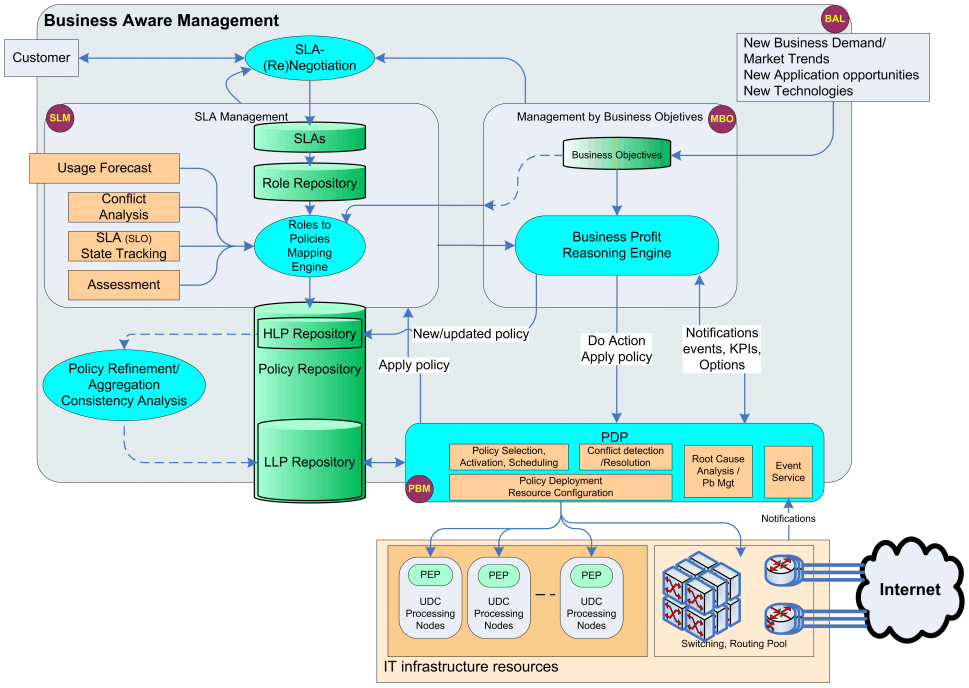


**Fig. 1. The Business Driven Management Framework (BDMF)**

As presented in fig. 1, the BDM architecture is divided into two main layers. On the top is the business aware layer (BAL), which is intended to host the long term control of the IT infrastructure based on the business objectives and market strategy of the IT provider. Beneath it is a resource control layer (RCL) which hosts the real time logic for the reactive short term control of the IT infrastructure.

The Business Management Layer is responsible for optimizing the alignment of IT resources usage with the objectives of the utility provider based on a set of business objectives defined and audited over relatively long periods of time (monthly, quarterly, etc.). Business objectives are the reflection of the utility provider's business strategy and range over diverse key performance indicators, which can be related to service operations, service level agreements, or any other business indicators.

Business relationships contracted by the utility provider are formalized by SLAs and modeled using the GSLA information model [1]. Using the GSLA, each contracted service relationship is modeled as a set of parties playing an SLA game in which each party plays one or more roles to achieve the SLA objec-

tives. Each role in the SLA is associated with a set of Service Level Objectives (SLOs) to be achieved; as well as a set of intrinsic policies related to the role behavior per see. A special engine, we call the Role-to-policies mapping engine, translates Roles, SLOs and rules into a set of enabling policies. These policies are further refined to lower level policies (LLPs) that enclose all the low level logic required to correctly drive the utility resources. Note that the automation of the translation per see is still an open research problem which is out of the scope of this paper. For our use case we use a pre-established mapping between SLA templates and their manually generated enforcement policies.

Business objectives affect the way SLAs are defined and managed at the resource control layer. So whenever a business objective is changed, added, or removed, important impact takes place at the long term time scale on the SLA database.

Low level policies (LLPs) are dealt with by the Policy Decision Point (PDP) module [3] of the resource control layer. Part of the PDP‘s task is to monitor and respond to system events and notifications by selecting, activating, and scheduling the enforcement of the appropriate policies at the appropriate utility resources. The PDP contains also sub-components for policy run-time conflict detection, root cause analysis, generation of the set of options available in the presence of some incident or problem, as well as a the generate of appropriate configuration flows in order to enforce active policies.

As it is impossible to define policies upfront to cover all run-time events, it will happen that LLPs may not be sufficient to deal with certain conditions. In those cases, our PDP passes up the control to the BDM. Given the various options, the BDM will select the one that will maximize the value to the utility provider. That is, the option that will result in the closest alignment to the business objectives. Such interactions offer also the opportunity for the architecture to learn and refine the policy repository.

## 4 A utility computing web hosting use case

We consider a Utility Computing Provider, named UCP, which offers web hosting services to client companies. UCP offers a generic SLA template as described in fig. 2. For simplicity, we assume that UCP provides two different web hosting packages each with different QoS assurances and pricing. These are advertised as two SLA types, SLA-T1 and SLA-T2, which are actually instances of the generic SLA of fig. 2. The SLA instances are depicted in table 1.

UCP is free to enforce those SLA types the way it wants as long as it is able to meet its customers' requirements. In our case a set of ECA rules (policies) are generated for each SLA type. The detail of this is explained after describing the overall management architecture.

C is provided a Web-Server Service with schedule SC
Capacity is of CP simultaneous connections
Initial storage capacity is S GB
C is monthly charged $ a x Capacity +
b x Storage.initial + c x Storage.Extra.Average
Monthly availability of the hosted service will be A%
Otherwise, refund C RA% of period PR
Any service unavailability will be fixed within at most MinTTR min
Otherwise, C will be refunded RMTTR of period RMTTRPR
Average time to process any customer service request over a month period will be less than ART ms
Otherwise, credit C RRT% of period RRTPR
IF C fails to pay the monthly charge for MaxM successive months the contract becomes void.

**Fig. 2. Generic UCP SLA Template**

**Table 1.** Template SLA instances of the Generic UCP SLA

| **Param** | **SLA-T1** | **SLA-T2** |
|---|---|---|
| SC | SC | SC |
| CP | 2000 | 5000 |
| S | 0.5 | 1 |
| a | 0.1 | 0.15 |
| b | 5 | 5 |
| c | 10 | 10 |
| A | 99 | 99.9 |
| RA | 100 | 100 |
| RAPR | breach period | month |
| MinTTR | $\infty$ | 20 |
| RMTTR | NA | 100 |
| RMTTRPR | NA | breach period |
| ART | 300 | 200 |
| RRT | 20 | 80 |
| RRTPR | month | month |
| MaxM | 3 | 6 |

It is interesting to notice that the deduction of the appropriate set of parameters and the number of SLA templates UCP desires to offer is a complex issue that depends not only on UCP's IT infrastructure but also on the market state and its evolution. Through simulations, it is possible to get a feeling of which templates it can more "safely" go with.

### 4.1 Business Metrics

Although T1 and T2 SLA types (Sec. 4) are specified over two service parameters, namely availability and service latency, they each define different service levels and have different penalties.

In our use case, we assume that UCP defines two Business Level metrics as input to it its overall Business Profit Function: *RP* representing the raw

financial profit in some monetary unit, and $CS$, a measure of overall customer satisfaction.

The Business Profit Function $\Psi$ represents a measurement of the overall utility gained and is defined as:

$$\Psi = \alpha \times RP + \beta \times CS \tag{1}$$

$$RP = \delta \sum_{s \in SLASet} RP(s) \tag{2}$$

$$CS = \sum_{i \in TemplateSet} \gamma_i \sum_{s \in T1SLASet} CS(s) \tag{3}$$

$\alpha, \beta, \gamma_i$, and $\delta$ represent pondering wheighs that are tunable to reflect the importance the IT operator (UCP) gives to each business indicator (metric). Simulations can be used to determine and tune these values.

Customer importance is translated by the existence of pondering factors related to each type of SLA. A more fine grained view of Customer importance involves per customer importance pondering factors. These factors are calculated based on Customer history (profit generated from that customer, service importance, etc.)

$RP$, although it involves a whole computation tree, is relatively easy and more accurate to measure compared to $CS$. The latter business indicator is less evident to measure as it generally depends on a relatively subjective appreciation of the customer of the overall service performance and might vary from customer to customer even with the same service performance. For the sake of keeping the use case simple enough, we assume that Customer Satisfaction is calculated as follows:

For the first month of service usage, customer satisfaction ($CS$) is measured by the number of experienced service failures with a maximum of $\%FedUp$ at the occurrence of which the customer becomes completely unhappy with the provided service ($CS = 0$).

$$CS_1 = 1 - \frac{1}{FedUp} Min\left(Nb\left(SLOFailures\right), FedUp\right) \tag{4}$$

On the following evaluation periods, the relationship becomes:

$$CS_{n+1} = \beta CS_n + (1-\beta) \left(1 - \frac{1}{FedUp} Min\left(Nb\left(SLOFailures\right), FedUp\right)\right) \tag{5}$$

This means that Customer Satisfaction is a pondered sum of the last customer satisfaction experience and the number of experienced Service Failures majored by the predefined threshold $FedUp$.

### 4.2 Generation of configuration policies

In order to enforce each contracted SLA, the utility provider translates the set of Service Level Objectives (SLOs) it has to guarantee into a set of access control and management policies that are stored into the policy repository.

Given that the UCP Roles-to-policies mapping engine knows that a UCP web server resource instance can serve up to 500 clients without reducing the contracted QoS, we understand from the policy set of Fig. 3 that the UCP considered the "lazy policy" of per-need provisioning to meet its SLAs. Whenever there is a need, an additional web server instance is installed and provisioned for customers. For brevity, we include the generated policy set of SLA-T2.

*T2-SLA*{...Role T2-SLA.UCP-Role{

SLO1 = WS.Capacity.Max = "5000"
SLO2 = WS.Availability.Month.Average.Percent $\geq$ 99.9%"
SLO3 = MinTimeToRepair( WS.availability) $\leq$ 20 min
SLO4 = WS.SingleRequest.Duration.Month.Average $\leq$ 200 ms
WSThreshold = 80%;
WS.SingleInstance.capacity = 200
policy P21= on SLA.Schedule.deployTime do (WS.installNew (WS.Config, WS.SingleInstance.capacity)
policy P22= on WS.installed do ConfigureAccessRights(C)
policy P23= on WS.installed do SwitchFabric.AssignQoSTag (WS.TrafficID, EF)
policy P24= on (WS.load == WSThreshold)
do WS.installNew (WS.Config, WS.SingleInstance.capacity )
when (WS.NbInstances * WS.SingleInstance.capacity $\leq$ WS.Capacity.Max )
priority = AboveNormal
policy P25= on (Web-Server.InstanceFailure)
do WS.installBackup (WS.Config,WS.SingleInstance.capacity )
priority = 5 (Critical)
policy P26= on WS.load¡WSThreshold-20% do WS.LastInstance.free()
policy P27= on fail(SLO2) do C.Credit(C.Charge)
policy P28= on fail(SLO3) do C.Credit ( C.Charge * duration(fail(SLO3) ) / Time.Month.Duration)
policy P29= on fail(SLO4) do C.Credit ( 0.8 * C.Charge )
policy P210= on fail (C-Role.Policy-1, 6) do TerminateContract ( C-SLA )}}

**Fig. 3. Generated SLOs and Policies for UCP-T2 Role**

### 4.3 Runtime conflicts

by analyzing the sets of generated policies for SLA-T1 and SLA-T2 it is possible to expect the occurence of runtime conflicts between each subset of elements of the policy set $\{p21, p24, p25, p12, p14, p15\}$. This happens when the UCP provider accepts to signs as much number of SLA-T1 and SLA-T2 SLAs so as the statistical average of their resource consumption meets the actual available system capacity. Hence, the system needs to work with cases where it might become in shortage of resources.

Default priorities between the generated policies will give policies belonging to SLA-T2 priority over those belonging to SLA-T1. The reason is that SLA-T2 customers bring more money per allocated resource and have also more stringent QoS requirements. However, at runtime and depending on the SLO

state of each SLA it might happen that at some point in time policies of SLA-T1 instances become more important to execute otherwise their respective SLOs will fail and generate real loss for the provider while if executed to the expense of delaying SLA-T2 policies will not generate penalties as long as there is still time for their respective SLOs to fail. This is an example of a runtime policy conflict which cannot be dealt with at compile time.

When the PDP is confronted here with a run-time policy conflict, it needs the assistance of the MBO for a wiser (business-driven) decision as a run-time policy conflict is generally symptom of service degradation that the PDP cannot measure correctly. The PDP hence sends a set of options for the MBO to decide which to apply.

The MBO engine will normally take into consideration service and business level parameters related to (i) SLAs (current total time of service unavailability, time to recover from unavailability, penalty amounts, Expectation of the evolution pattern for the number of customers for each SLA (to decide whether to allocate resources or just do not if the congestion period is expected to be temporary); and (ii) Current customers satisfaction indicators. In our implementation, it currently evaluates the impact of SLO failure on the BPF and decides by minimizing the overall impact on it.

It is clear that it is not necessarily the FIFO treatment of active policies which will lead to the best business profit. When the PDP knows that there will be inevitably a degradation of service (that might lead or not to some SLO violation) the MBO answers about the best options (strategy) to follow so as to achieve minimal degradation of the business profit function.

### 4.4 BDMF Simulation Testbed

In order to validate our framework, we designed a java based discrete event simulation system which features the main components of BDMF. The system is composed of three major packages:

- A base package of a process-based discrete event simulation system.
- Package edu.bdmf which implements all the components of BDMF.
- A testbed package which actually implements our running example.

The edu.bdmf package is the core component of the implementation. It builds on the base discrete process-based simulation package and features the core of a policy based management system supported by the high-level features of BDMF. These include the support for SLA specification according to the GSLA model [1], definition of business objectives; specification of metrics at resource, service and up to the overall business profit function; as well the connection with the BPF maximization (impact minimization) engine by determining which decision making algorithm to apply with regards to the scheduling of active policies (actions).

The generic SLA (sec. 2) of our running example is implemented as a java class with the associated attributes as class parameters. The two SLA templates

we used are subclasses of it. A similar reuse technique is used in the Roles [1] and policy specifications. For example, policy p21 of SLA template T2 and p12 (not shown here) of SLA T1 are the same and hence specified using the same class.

We compare the plain FIFO scheduling method, which launches triggered policy actions in the order of their arrival, to the second scheduling method employs a greedy approach to BPF maximization by selecting each time the policy which at the time being is expected to lead to the best positive participation (or least negative participation) to the BPF.

The simulation shows no difference between the two techniques as far as the set of available server units has not been exhausted. However, When the system operates in congested conditions either by increasing the number of contracted SLAs or the end customer connection rate, the greedy BPF maximization scheduling technique shows its power by succeeding to keep the BPF always at a better value than the plain FIFO method. The gain interval tends to reach a maximum value after which the two functions tend to converge to the same loss levels.

## 5 Conclusion

This paper presented the BDMF management framework which extends policy-based management with a wider scope decision ability that is informed and driven through the business objectives and the contractual obligations of the enterprise supported by the IT systems being managed.

We described preliminary results of the implementation of the BDMF framework on the base of a discrete process-based event simulation system. These include the utility generation and scheduling of management actions based on business context and the set of service level agreements an IT operator has contracted. We used a web hosting company example to illustrate the system functionalities. The use case showed the importance of management actions scheduling for maximizing the business profit function of IT providers.

The BDMF simulation environment can serve as a basis for the analysis and validation of policy performance at run time. In addition, our implementation illustrates how class inheritance can serve as a powerful tool to reinforce reusability of SLAs, Roles, and Policies at both template and instance levels.

We believe that this work represents a first step towards management actions scheduling and execution based on high level business metrics. As a future work, we plan to further validate our framework through simulations using other business level objectives and different scheduling techniques.

## References

1. I. Aib, N. Agoulmine, and G. Pujolle. A multi-party approach to SLA modeling, application to WLANs. In *Second IEEE Consumer Communications and*

*Networking Conference (CCNC)*, pages 451 – 455. IEEE, Jan 3-5 2005.
2. I. Aib, M. Salle, C. Bartolini, and A. Boulmakoul. A business driven management framework for utility computing environments. In *proceedings of the Ninth IFIP/IEEE International Symposium on Integrated Network Management (IM 2005)*. IEEE, 16-19 May 2005.
3. E. B. Moore. Policy core information model (PCIM) extensions, rfc 3460, January 2003.
4. C. Bartolini, A. Boulmakoul, A. Christodoulou, A. Farrell, M. Salle, and D. Trastour. Management by contract: IT management driven by business objectives. In P. University of Evry, editor, *HP Open University Association (HPOVUA)*, volume 10. HPL, June 2004.
5. C. Bartolini and M. Sall. Business driven prioritization of service incidents. In *Utility Computing: 15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM*, volume 3278 of *Lecture Notes in Computer Science*, pages 64–75. Springer-Verlag GmbH, Jan 2004.
6. M. Buco, R. Chang, L. Luan, C. Ward, J. Wolf, P. Yu, T. Kosar, and S. U. Shah. Managing ebusiness on demand SLA contracts in business terms using the cross-SLA execution manager SAM. In *ISADS '03: Proceedings of the The Sixth International Symposium on Autonomous Decentralized Systems (ISADS'03)*, page 157, Washington, DC, USA, 2003. IEEE Computer Society.
7. M. J. Buco, R. N. Chang, L. Z. Luan, C. Ward, J. L. Wolf, and P. S. Yu. Utility computing SLA management based upon business objectives. *IBM Systems Journal*, 43(1):159–178, 2004.
8. A. Dan, D. Davis, R. Kearney, A. Keller, R. P. King, D. Kuebler, H. Ludwig, M. Polan, Mike, Spreitzer, and A. Youssef. Web services on demand: WSLA-driven automated management. *IBM Systems Journal*, 43(1):136–158, 2004.
9. M. Debusmann and A. Keller. SLA-driven management of distributed systems using the common information model. In *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 563, 2003.
10. A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Networks and Systems Management*, 11(1), 2003.
11. H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. Web service level agreement (WSLA) language specification. Technical report, IBM T.J. Watson Research Center, 2003.
12. V. Machiraju, A. Sahai, and A. van Moorsel. Web services management network: An overlay network for federated service management. In *Proceedings of the VIII IFIP/IEEE IM conference on network management*, 2003.
13. A. Sahai, V. Machiraju, M. Sayal, A. P. A. van Moorsel, and F. Casati. Automated SLA monitoring for web services. In M. Feridun, P. G. Kropf, and G. Babin, editors, *DSOM*, volume 2506 of *Lecture Notes in Computer Science*, pages 28–41. Springer, Nov. 16 2002.
14. A. Sahai, S. Singhal, R. Joshi, and V. Machiraju. Automated policy-based resource construction in utility computing environments. In *IEEE/IFIP NOMS*, 2004.
15. M. Salle and C. Bartolini. Management by contract. In *IEEE/IFIP NOMS*, 2004.