A Methodological Approach toward the Refinement Problem in Policy-Based Management Systems

Javier Rubio-Loyola and Joan Serrat, Universitat Politècnica de Catalunya

Marinos Charalambides, Paris Flegkas, and George Pavlou, University of Surrey

ABSTRACT

Policy refinement is meant to derive low-level enforceable policies from high-level guidelines. Although recent advances have been made to solve this open problem, the holistic implications confronting systematic policy refinement have not been explicitly addressed. This article presents a methodological approach towards the policy refinement problem. We provide a generic procedure to define policy hierarchies, which is essential to achieving systematic policy refinement. We also provide the considerations while defining high-level guidelines, and describe a policy refinement framework that formalizes the requirements to refine high-level guidelines into executable policies. We demonstrate the feasibility of our approach with a scenario applied to the quality of service (QoS) management domain.

INTRODUCTION

Policy-based management has been regarded as a suitable approach to manage and control complex systems. Despite its potential benefits of flexibility and constrained programmability, this solution is still not a reality. In addition to the crucial problem of policy conflicts, a key issue behind the reticence to adopt this technology is the necessity of deriving executable policies aligned with administrative guidelines. The latter is normally referred to as the *policy refinement problem*.

The first solid work that addressed policy refinement proposed a methodology grounded in goal-oriented requirements engineering (GORE) techniques [1]. In this approach, the achievements of policies are viewed as goals and standard goal-oriented refinement methods are used to guide the refinement process at different levels of abstraction. Recent efforts have proposed reactive systems analysis techniques to automatically produce executable policies in goal-oriented refinement environments [2]. Nevertheless, none of these methodologies have been demonstrated in complex systems and their feasibility to address real-life scenarios is questioned.

This article describes how our approach to policy refinement proposed in [2] is applicable to realistic refinement scenarios in the domain of quality of service (QoS) management. We propose a methodological approach towards the policy refinement problem which comprises the following main elements:

- A procedure to define policy hierarchies tailored to addressing policy refinement
- General considerations and parties involved in the definition of high-level guidelines
- A formal and realizable framework that provides support to every activity and phase of the refinement process

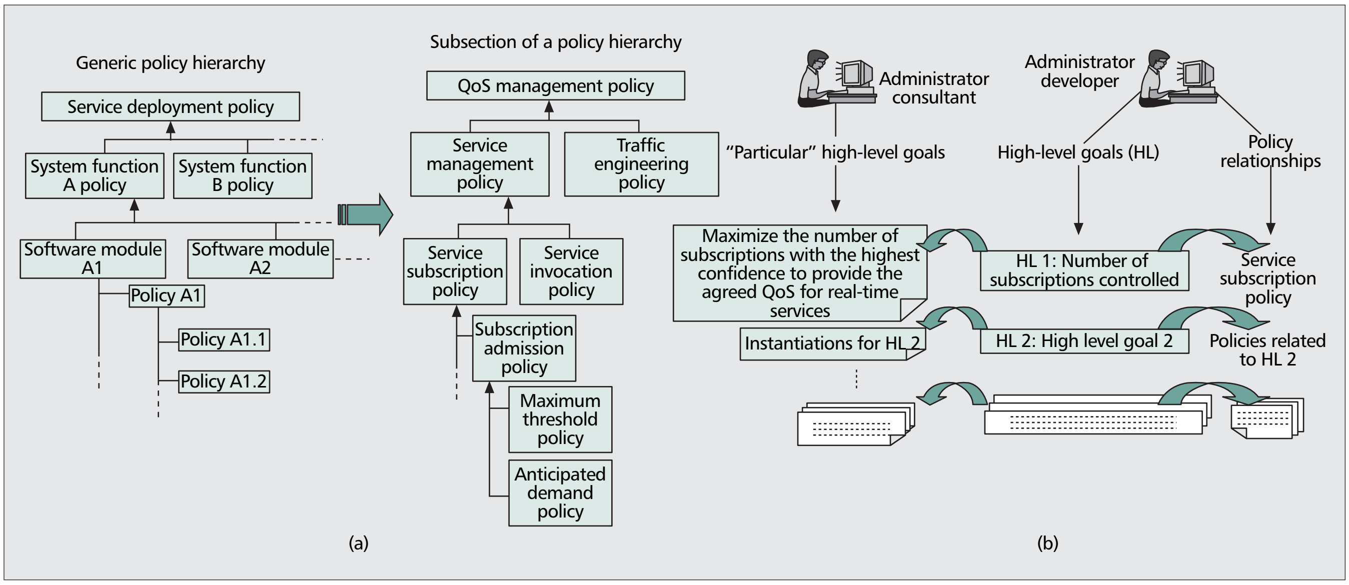
The article is organized as follows: we first describe the main elements of our approach and then exemplify its applicability through a refinement scenario applied to the QoS management domain. We finally describe some related work and conclude this article.

METHODOLOGICAL APPROACH TO POLICY REFINEMENT

In this section we describe a complete approach to policy refinement that integrates the following elements:

- Definition of a policy hierarchy of the policy system
- Identification of high-level guidelines
- Definition of a policy refinement framework that captures the requirements to allow refining executable policies from high-level guidelines systematically

In this article the term *goal* is used to denote both administrative guidelines and business objectives. The term policy is used for executable rules. Policies at the lowest level have an unambiguous interpretation and are capable of being executed automatically. Goals instead are interpreted in context and may be achieved in different ways [3].


Generic policy hierarchy
Service deployment policy
System function A policy
System function B policy
Software module A1
Software module A2
Policy A1
Policy A1.1
Policy A1.2
Subsection of a policy hierarchy
QoS management policy
Service management policy
Traffic engineering policy
Service subscription policy
Service invocation policy
Subscription admission policy
Maximum threshold policy
Anticipated demand policy
(a)
Administrator consultant
Administrator developer
"Particular" high-level goals
High-level goals (HL)
Policy relationships
Maximize the number of subscriptions with the highest confidence to provide the agreed QoS for real-time services
HL 1: Number of subscriptions controlled
Service subscription policy
Instantiations for HL 2
HL 2: High level goal 2
Policies related to HL 2
(b)

■ **Figure 1.** *a) Generic policy hierarchy and a subsection sample; b) generic considerations to define high-level goals.*

Defining Policy Hierarchies

Policy hierarchies have been regarded to be of great value for determining the requirements of satisfying high-level guidelines, tracking changes on lower-level policies due to changes of high-level ones, and automatically generating lower-level policies [4]. Nevertheless, to the best of our knowledge, the principles behind the derivation of policy hierarchies have not been explicitly addressed.

We propose the formulation of policy hierarchies that mirror the policy system architecture, taking advantage of inherent hierarchical relationships. We recommend that policy hierarchies should be defined during the design of the policy system.

A generic policy hierarchy and a sample for the case we use are shown in Fig. 1a. The following are the principles used to define this type of hierarchy:

• Define the highest-level policy which will correspond to the service provided by the policy system, namely, the service deployment policy. For instance, if the policy system controls QoS provisioning, the highest-level policy would be defined as the QoS management policy.

• Define the policy-controlled system functions. Linked to the highest-level policy, there would be as many lower-level policies as policy-controlled system functions. For example, if the QoS management system integrates service-management and traffic-engineering system functions, these functions would define the service management policy and the traffic engineering policy in the hierarchy.

• Define the policy-controlled software modules (i.e., system components) that would execute each system function. Each "software module" would enforce a "software module policy" in the policy hierarchy. For example, if the service management function is implemented by two modules, service subscription and a service invocation, this function will be controlled by the service subscription policy and service invocation policy. There would be as many software module policies as policy-controlled modules enforcing each system function.

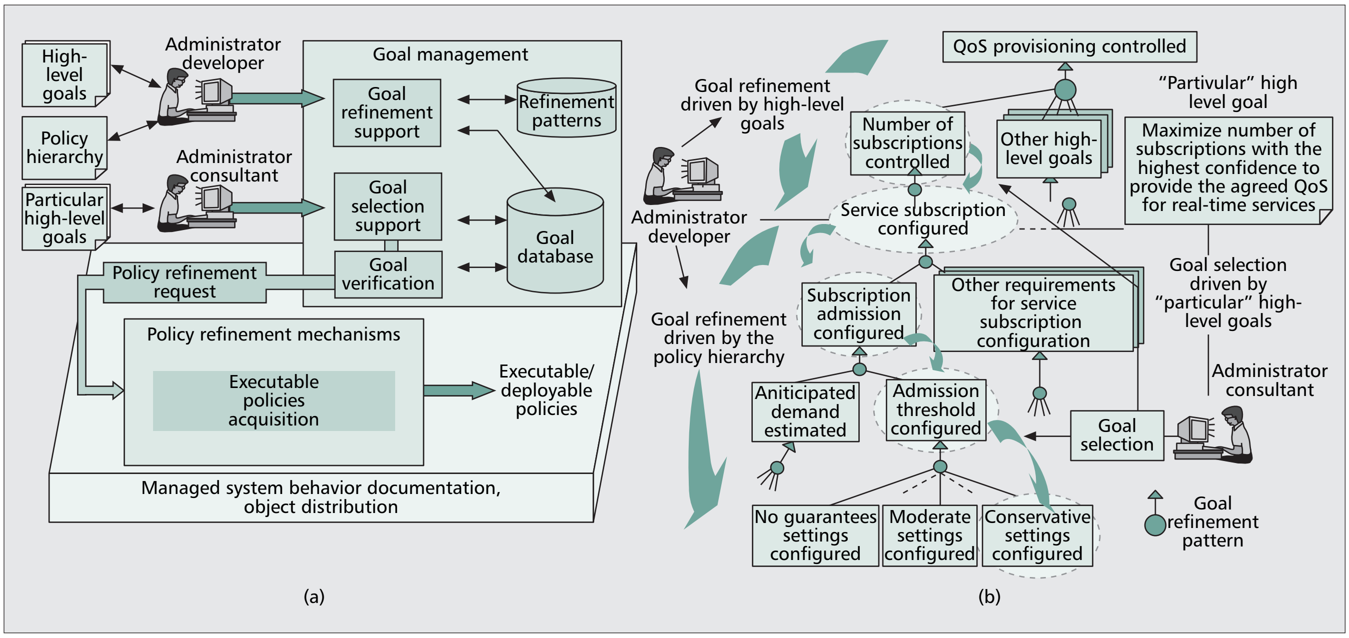
• Define the policy-controlled subfunctions in each software module. Each subfunction would define a specific type of policy (e.g., Policy A1 controls the subfunction 1 of software module A). When different policy-controlled parameters influence a given subfunction in a specific software module, a lower-level policy would be defined for each of these parameters (Policy A1.1 and Policy A1.2 would define Policy A1). Consider that subscription admission control is a subfunction of the service subscription module, and that the former is influenced by two policy-controlled parameters: maximum threshold and anticipated demand. In this case, the subscription admission policy would be implemented by both the maximum threshold and anticipated demand policies.

It is worth mentioning that the policy refinement process should produce instances of lowest-level enforceable policies committing to the actual implementation of the system.

Defining High-Level Goals

Different methods have been proposed to specify business objectives and to provide indicators to assess IT performance as it relates to them [5]. Although this is an application-oriented issue, it is imperative to express and represent business objectives in an approachable manner. In this section we describe the general considerations during the definition of business objectives, further referred to as high-level goals.

We identify two administrative actors: the administrator developer and the administrator consultant. The former defines, during the design of the system, the high-level goals that the system can handle and the different ways to achieve them. This party also establishes the relationship between high-level goals and the policies that help in achieving them. On the other hand, the administrator consultant interprets high-level goals to define the "particular" goals that better


High-level goals
Administrator developer
Goal management
Goal refinement support
Refinement patterns
Policy hierarchy
Administrator consultant
Goal selection support
Goal database
Particular high-level goals
Goal verification
Policy refinement request
Policy refinement mechanisms
Executable policies acquisition
Executable/ deployable policies
Managed system behavior documentation, object distribution
(a)
QoS provisioning controlled
Goal refinement driven by high-level goals
"Partivular" high level goal
Number of subscriptions controlled
Other high-level goals
Maximize number of subscriptions with the highest confidence to provide the agreed QoS for real-time services
Administrator developer
Service subscription configured
Goal selection driven by "particular" high-level goals
Subscription admission configured
Other requirements for service subscription configuration
Goal refinement driven by the policy hierarchy
Administrator consultant
Aniticipated demand estimated
Admission threshold configured
Goal selection
No guarantees settings configured
Moderate settings configured
Conservative settings configured
Goal refinement pattern
(b)

■ **Figure 2.** *a) Goal-oriented policy refinement framework; b) goal refinement and goal selection example.*

reflect the administrative views of how to achieve high-level goals at system operation time.

A graphical example of the above is shown in Fig. 1b. The administrator developer defines, during the design of the system, the high-level goal "number of subscriptions controlled" to control the number of subscriptions of the managed system. This goal is in turn related to the policy dedicated to control the service subscriptions, namely, the service subscription policy. During the operation of the system the administrator consultant interprets this high-level goal and defines a "particular" goal instantiating the former with "maximize the number of subscriptions with the highest confidence to provide the agreed quality of service for real-time services." Several instantiations may be done for other types of services and for other high-level goals; moreover, this should result in a set of executable policies reflecting such administrative decisions.

Policy Refinement Framework

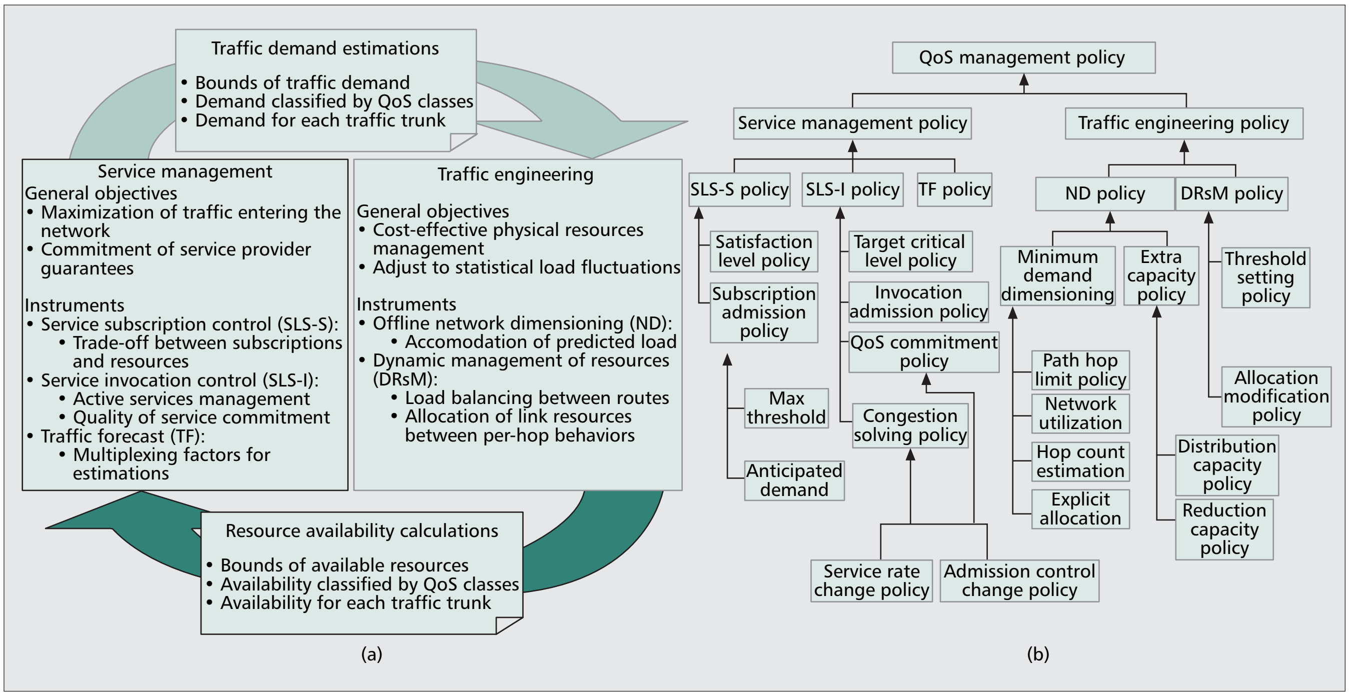
Our policy refinement framework allows the refining of lowest-level executable policies from high-level goals in a systematic manner, providing support during both design and operation of the system. It integrates two main functions: *goal management* and *policy refinement mechanisms*, as demonstrated in Fig. 2a.

Goal Management — The goal management is integrated by the subfunctions goal refinement support, goal selection support, and goal verification.

Goal Refinement Support — Through goal refinement, the administrator developer formalizes the requirements and alternatives with which high-level goals can be achieved. This activity is application-dependent and should be carried out during system design. It should be accomplished in two phases:

1) Goal refinement driven by high-level goals: This phase is carried out establishing associations between the highest-level goal of the policy system and the high-level goals that such system can handle. For example, given that our sample system is aimed at controlling the QoS provisioning, the highest-level goal of this policy system would be formalized as "QoS provisioning controlled." This highest-level goal would be in turn refined into the "number of subscriptions controlled" goal and "other high-level goals," as shown in Fig 2b. Each high-level goal is in turn refined into the achievements of the policies controlling such high-level goal. For example, due to the relationship between the "number of subscriptions controlled" high-level goal and the "service subscription policy" (Fig. 1b), the "number of subscriptions controlled" goal should be refined into the goal "service subscription configured" (Fig. 2b).

2) Goal refinement driven by the policy hierarchy: In this second phase each goal-graph subtree is refined into the achievements prescribed by the policy hierarchy. For example, in the context of our QoS provisioning system, the "service subscription policy" is enforced (among others) by the "subscription admission policy" which is in turn enforced by the "maximum threshold policy" and "anticipated demand policy" (Fig. 1a). In this sense, the goal refinement process should consider that the "service subscription configured" goal would be refined into the goal "subscription admission configured" and "other requirements for service subscription configuration." Following on, the "subscription admission configured" goal would be refined into the "anticipated demand estimated" goal and the "admission threshold configured" goal. Further refinements for each of these would represent the alternatives to fulfill the corresponding goal. This is graphically illustrated in the lower part of the goal-graph structure shown in Fig. 2b.


Traffic demand estimations
• Bounds of traffic demand
• Demand classified by QoS classes
• Demand for each traffic trunk
Service management
General objectives
• Maximization of traffic entering the network
• Commitment of service provider guarantees
Instruments
• Service subscription control (SLS-S):
• Trade-off between subscriptions and resources
• Service invocation control (SLS-I):
• Active services management
• Quality of service commitment
• Traffic forecast (TF):
• Multiplexing factors for estimations
Traffic engineering
General objectives
• Cost-effective physical resources management
• Adjust to statistical load fluctuations
Instruments
• Offline network dimensioning (ND):
• Accomodation of predicted load
• Dynamic management of resources (DRsM):
• Load balancing between routes
• Allocation of link resources between per-hop behaviors
Resource availability calculations
• Bounds of available resources
• Availability classified by QoS classes
• Availability for each traffic trunk
(a)
QoS management policy
Service management policy
Traffic engineering policy
SLS-S policy
SLS-I policy
TF policy
ND policy
DRsM policy
Satisfaction level policy
Subscription admission policy
Target critical level policy
Invocation admission policy
QoS commitment policy
Minimum demand dimensioning
Extra capacity policy
Threshold setting policy
Max threshold
Anticipated demand
Congestion solving policy
Path hop limit policy
Network utilization
Hop count estimation
Explicit allocation
Allocation modification policy
Distribution capacity policy
Reduction capacity policy
Service rate change policy
Admission control change policy
(b)

■ **Figure 3.** *a) Generic approach for QoS management; b) policy hierarchy for QoS management.*

Goal Selection Support — Through goal selection, the administrator consultant selects, among the alternatives, the "particular" high-level goals that better reflect the administrative criteria at service operation time. Consider for instance that the consultant of our QoS provisioning example opts to "maximize the number of subscriptions with the highest confidence to provide the agreed quality of service for real-time services." The goal selection support will direct the consultant to instantiate the pattern of goals, as marked with dotted lines in Fig. 2b.

Goal Verification — This subfunction verifies that the goal selections are complete and logically correct. In other words, these verify that the selected strategies certainly satisfy the "particular" high-level goals. For example, having the administrator consultant selected the goal "subscription admission configured," a simple verification would be that the goal selection certainly includes both refinements: "anticipated demand estimated" and "admission threshold configured" goals.

Policy Refinement Mechanisms — The policy refinement mechanisms function provides support to acquire the lowest-level executable policies that would fulfill the "particular" high-level goals at operation time. These should commit with the actual object distribution of the policy system for further deployment. These mechanisms implement different processes for which the following inputs are provided:
• Policy refinement requests: The verified goal selections provided by the goal management functions.
• Managed system behavior documentation: Documentation of how the managed elements behave, interact and collaborate. This is provided in standard notations like UML.
• Object distribution: Inventory of the actual managed objects.

A detailed description of the analysis techniques of our framework can be found in [2]. The details of a realizable prototype can be found in [6].

VALIDATION OF OUR METHODOLOGICAL APPROACH

This section presents the validation of our methodological approach. We initially describe the application domain that we have used for this purpose and then define a policy hierarchy and realistic high-level goals for this domain. Lastly, we show the practicality of our approach using our framework implementation.

A QOS MANAGEMENT APPROACH

The QoS Management solution in which we have validated our approach relies on the principles developed in the context of the IST project TEQUILA — Traffic Engineering for Quality of Service for the Internet at Large Scale [7]. A simplified representation of this approach is depicted in Fig. 3a., which shows the integration of service management and traffic engineering functions to achieve QoS provisioning in IP Networks.

The service management functionality has two objectives: the maximization of traffic entering the network, and the commitment of the service provider's QoS guarantees. As the traffic entering the network is a function of the number of subscribed contracts and active services, admission control mechanisms are defined for service subscriptions and invocation requests. QoS commitment is addressed by enforcing pre-

The SLS-I policy controls the number and type of active services and consequently, the volume of injected traffic. It addresses the trade-off between the number of admitted invocations and preventing QoS degradation due to network overloading.

ventive and corrective actions as a means to police misbehaving users and to resolve congestion.

The traffic engineering functionality is concerned with the management of resources. An off-line dimensioning process is responsible for mapping the predicted traffic demand to physical network resources. In addition, real-time operations are implemented as a means to first, balance the load amongst the established label-switched paths (LSPs) in the network, and second, to ensure that link capacities are appropriately distributed among the different per-hop behaviors (PHBs) sharing each link.

A Policy Hierarchy for QoS Management

Following the principles of our methodological approach, a QoS management policy must consider both service management and traffic engineering policies given that QoS delivery involves these two functions. A policy hierarchy for QoS management is shown in Fig. 3b for which we provide a brief description.

The service management functions are carried out by three components: service subscription (SLS-S), service invocation (SLS-I), and traffic forecast (TF) that define the *SLS-S policy*, *SLS-I policy*, and *TF policy*, respectively .

The SLS-S policy offers the necessary programmability to control service subscriptions: acceptance, negotiation, and rejection. It controls the trade-off between the number of subscriptions and the confidence for ensuring a given QoS. It is enforced by two lower-level policies: the satisfaction level policy and the subscription admission policy.

- The satisfaction level policy is used to define levels of confidence for service fulfillment.
- The subscription admission policy deals with subscription admission control. Given that the admission control logic is a function of the maximum anticipated demand, this policy is enforced by two lowest-level policies: the max threshold, and the anticipated demand. These two have been derived as the means to define thresholds for admission control and the multiplexing factors that would influence the considerations for the total anticipated demand, and consequently the admission control logic.

The SLS-I policy controls the number and type of active services and consequently, the volume of injected traffic. It addresses the trade-off between the number of admitted invocations and preventing QoS degradation due to network overloading. This policy controls different subfunctions defining four lower-level policies: target critical level policy, invocation admission policy, QoS commitment policy, and congestion solving policy.

- The target critical level policy defines the level at which the likelihood of overwhelming the network is considered critical.
- The invocation admission policy controls the invocation admission functionalities.
- The QoS commitment policy prevents QoS degradation by enforcing proactive actions.
- The congestion solving policy executes penalty actions to avoid congestion.

The enforcement of the latter two policies may result in service-rate reallocations and/or admission control readjustments for new invocations. These two define the service rate change policy and the admission control change policy, respectively.

The TF policy is used to define the criteria by which a service is considered to enjoy an "almost satisfied" and a "fully satisfied" rate, both of which are used to define the bounds of traffic demand estimations.

The *traffic engineering* functions are supported by the network dimensioning (ND) and dynamic resource management (DRsM) components, which in turn define the *ND policy* and *DRsM policy* in the hierarchy.

The ND policy controls the accommodation of traffic estimations into the physical resources and is enforced by two lower-level policies; the minimum demand dimensioning policy controls the strategy of allocation of the minimum estimated traffic demand and the extra capacity policy controls the allocation of the remaining resources.

The *DRsM policy* drives the dynamic resource management functions. It controls the criteria on how resources allocation should be redefined when considerable load fluctuations take place. The lowest-level policies for the ND policy and the DRsM policy are shown in Fig. 2b.

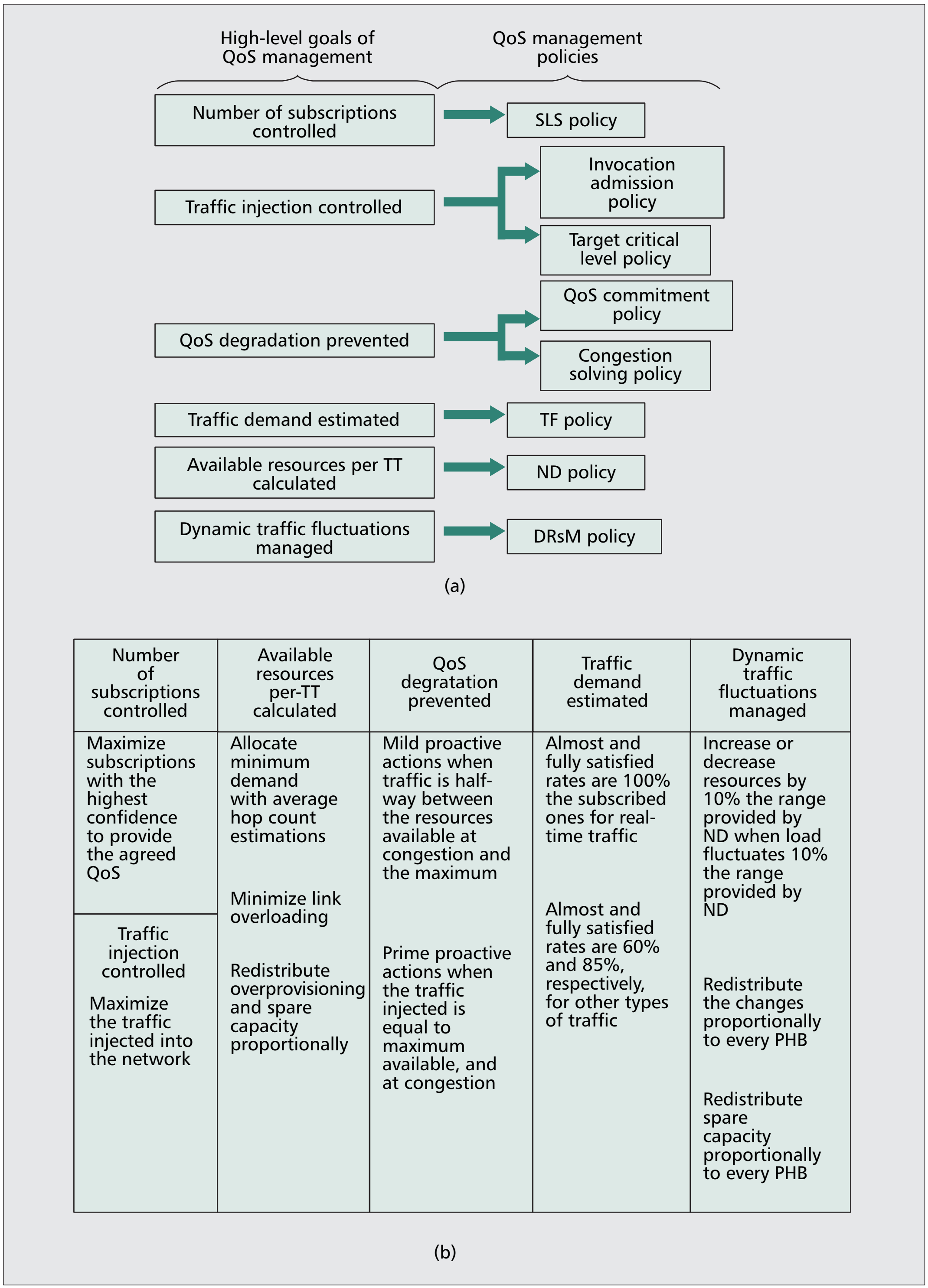
High-Level Goals for QoS Management

Different methods and mechanisms can be found in the literature [5] to specify business objectives and to provide indicators to assess IT performance related to them. In the context of our scenario, these represent the high-level goals with which the administrator developer controls QoS provisioning.

In our QoS management scenario the administrator developer considers the following high-level goals: *number of subscriptions controlled*, *traffic injection controlled*, *QoS degradation prevented*, *traffic demand estimated*, *available resources per-traffic trunk calculated*, and *dynamic traffic fluctuations managed*. The administrator developer should establish a relationship between these high-level goals with the achievement of the policies specified in the policy hierarchy.

For example, the high-level goal number of subscriptions controlled is directly influenced by the service subscription policy given that the latter is used to control the acceptance and rejection/negotiation of service subscriptions. This way, the developer establishes a relationship between the number of subscriptions controlled high-level goal and the SLS policy. Similarly, the high-level goal traffic injection controlled is influenced by the traffic entering the network and the QoS enjoyed by the active services. The latter two aspects are influenced by the invocation admission policy and target critical level policy and, consequently, these policies are related to the traffic injection controlled high-level goal. Figure 4a summarizes the high-level goals and their association(s) with the policy hierarchy in our scenario.

The interpretation of the above high-level goals would define a "particular" view for QoS management. For example, the instantiations shown in Fig. 4b provide an administrator con-


High-level goals of QoS management
QoS management policies
Number of subscriptions controlled
SLS policy
Traffic injection controlled
Invocation admission policy
Target critical level policy
QoS commitment policy
QoS degradation prevented
Congestion solving policy
Traffic demand estimated
TF policy
Available resources per TT calculated
ND policy
Dynamic traffic fluctuations managed
DRsM policy
(a)
(b)

Number of subscriptions controlled	Available resources per-TT calculated	QoS degradation prevented	Traffic demand estimated	Dynamic traffic fluctuations managed
Maximize subscriptions with the highest confidence to provide the agreed QoS	Allocate minimum demand with average hop count estimations	Mild proactive actions when traffic is half-way between the resources available at congestion and the maximum	Almost and fully satisfied rates are 100% the subscribed ones for real-time traffic	Increase or decrease resources by 10% the range provided by ND when load fluctuates 10% the range provided by ND
Traffic injection controlled	Minimize link overloading			
Maximize the traffic injected into the network	Redistribute overprovisioning and spare capacity proportionally	Prime proactive actions when the traffic injected is equal to maximum available, and at congestion	Almost and fully satisfied rates are 60% and 85%, respectively, for other types of traffic	Redistribute the changes proportionally to every PHB
				Redistribute spare capacity proportionally to every PHB

Figure 4. *a) High-level goals and policy relationships for QoS management; b) "particular" view of QoS provisioning management.*

The QoS provisioning management goal has been refined into the six high-level goals for QoS management. Then,the goal graph is built upon six subtrees. Subsequently, the developer documents the relationship of these high-level goals with the policies that control the corresponding high-level goals.

sultant's view of QoS management at system operation time. Another consultant could define different views according to previous experiences, statistical data, and so forth.

The first column, for example, concerns the view of QoS delivery with respect to the high-level goals number of subscriptions controlled and traffic injection controlled. For the former, the consultant opts to maximize the number of subscriptions at the highest confidence levels. This directive implies that congestion occurrence would be highly unlikely and hence the consultant has opted to maximize the traffic injected into to the network for the latter directive.

Policy Refinement Process

In order to show the practicality of the refinement process, we have used a prototype of our framework which supports the following features:

- *Industrial support for goal management*. The prototype [6] integrates Objectiver, a formal tool for goal-oriented specifications that provides support for goal refinement and goal selection.
- *Tool support for system documentation*. The prototype integrates an ArgoUML toolkit to document the managed system features in UML standard notations.
- *Support to acquire executable policies automatically*.

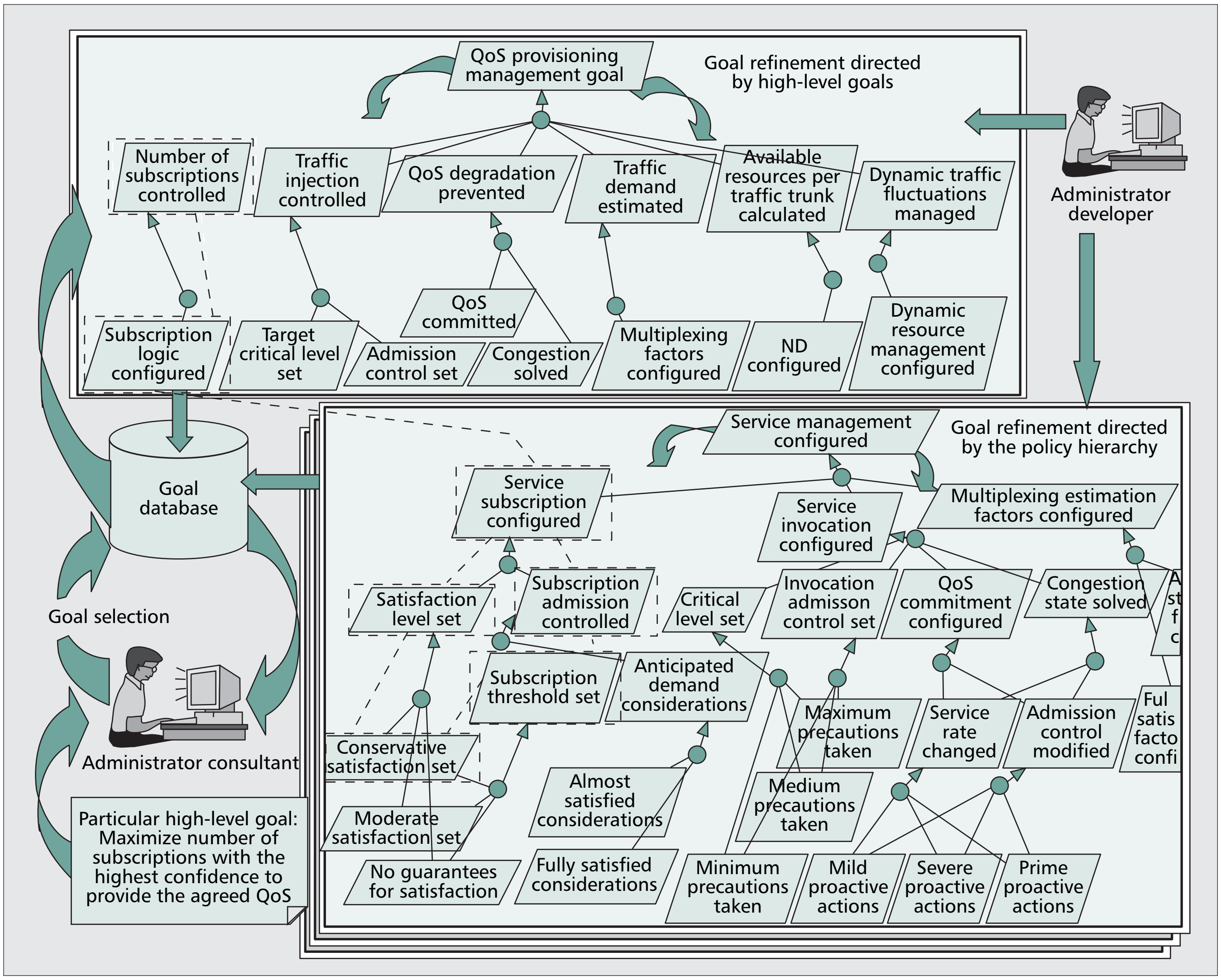
Different tools and ad hoc software modules specialize the automated policy refinement Mechanisms of the framework. The prototype includes:

- A Hugo/RT toolkit to allow translating system documentation into code for formal analysis
- A SPIN searching engine to find the necessary system behavior that fulfils specific system goals
- A modified Ponder toolkit for policy specification

The prototype allows goal refinement considering the QoS-oriented high-level goals and the policy hierarchy. Figure 5 shows two out of the four generic goal graphs elaborated by the developer in the scenario:

1 The QoS provisioning management goal (top) has been refined into the six high-level goals for QoS management. Then,the goal graph is built upon six subtrees. Subsequently, the developer documents the relationship of these high-level goals with the policies that control the corresponding high-level goals.
2 The service management configured goal (goal graph on the bottom right) has been refined considering the policy hierarchy, in this particular case, the service management policy. The result of this process is a goal graph representing the different strategies for the service management function. Similar goal graphs have been defined for the traffic engineering functions.

At service operation time the administrator consultant browses through the goal data base to carry out a goal selection process. For instance, for the high-level-goal number of subscriptions controlled in Fig. 4b, the system guides the consultant to the subscription logic configured goal, and subsequently to the service subscription configured goal, the latter included in the service management configured goal graph. At this moment of the selection, the consultant should

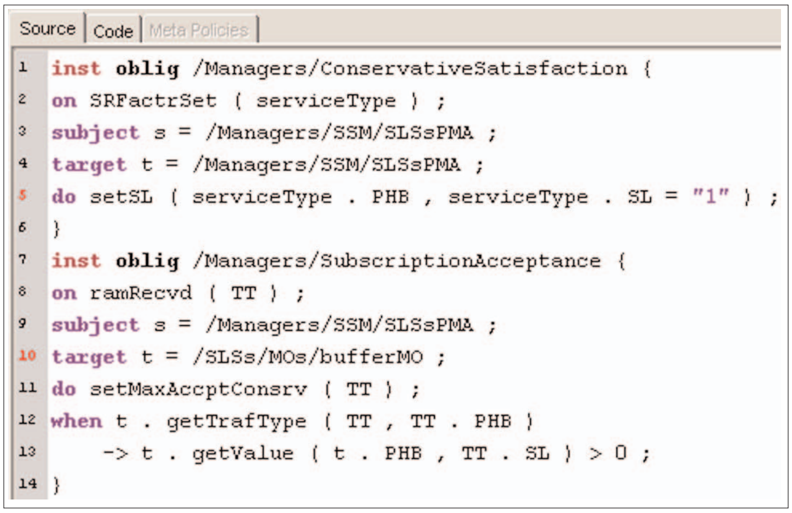

QoS provisioning management goal
Goal refinement directed by high-level goals
Number of subscriptions controlled
Traffic injection controlled
QoS degradation prevented
Traffic demand estimated
Available resources per traffic trunk calculated
Dynamic traffic fluctuations managed
Administrator developer
QoS committed
Subscription logic configured
Target critical level set
Admission control set
Congestion solved
Multiplexing factors configured
ND configured
Dynamic resource management configured
Goal database
Goal selection
Administrator consultant
Particular high-level goal: Maximize number of subscriptions with the highest confidence to provide the agreed QoS
Service management configured
Goal refinement directed by the policy hierarchy
Service subscription configured
Service invocation configured
Multiplexing estimation factors configured
Satisfaction level set
Subscription admission controlled
Critical level set
Invocation admisson control set
QoS commitment configured
Congestion state solved
Subscription threshold set
Anticipated demand considerations
Conservative satisfaction set
Maximum precautions taken
Service rate changed
Admission control modified
Medium precautions taken
Almost satisfied considerations
Moderate satisfaction set
No guarantees for satisfaction
Fully satisfied considerations
Minimum precautions taken
Mild proactive actions
Severe proactive actions
Prime proactive actions

■ **Figure 5.** *Practicality of the policy refinement process: goal refinement and goal selection.*

decide the guidelines that reflect the administrative view about the service subscription configured goal. For these, the "conservative settings" for both, the satisfaction level and the admission control, reflect the view of "Maximize subscriptions with the highest confidence to provide the agreed QoS" (selection marked with dotted lines in Fig. 5). Similar selections should be achieved for the remaining "particular" goals of Fig. 4b.

Once the operative view for QoS provisioning has been defined, the resulting goal selection is verified for correctness and completeness. Later, the policy refinement mechanisms produce the lowest-level executable policies that commit with such selection and, consequently, with the view of QoS management.

In this scenario, five ND policies, four TF policies, two SLS-S policies, six SLS-I policies, and eight DRsM policies have been produced in less than three seconds. Two of these lowest-level executable policies are shown in Fig. 6. These are related to the number of subscriptions controlled high-level goal. The conservative satisfaction policy defines appropriate values on which the service admission mechanisms would rely; namely, the satisfaction level. The value assigned to the latter suggests that the active SLSs would enjoy their QoS at their almost satisfied rates, even at congestion. Other policies have been produced to influence the parameters to define the almost satisfied rates. The subscription acceptance policy sets thresholds for incoming subscription requests. It is worth noticing that the events, subjects, targets, actions, and constraints of the 25 lowest-level policies of our scenario have been abstracted automatically by the policy refinement mechanisms.



```
inst oblig /Managers/ConservativeSatisfaction {
on SRFactrSet ( serviceType ) ;
subject s = /Managers/SSM/SLSsPMA ;
target t = /Managers/SSM/SLSsPMA ;
do setSL ( serviceType . PHB , serviceType . SL = "1" ) ;
}
inst oblig /Managers/SubscriptionAcceptance {
on ramRecvd ( TT ) ;
subject s = /Managers/SSM/SLSsPMA ;
target t = /SLSs/MOs/bufferMO ;
do setMaxAccptConsrv ( TT ) ;
when t . getTrafType ( TT , TT . PHB )
    -> t . getValue ( t . PHB , TT . SL ) > 0 ;
}
```

Figure 6. *Subset of refined lowest-level policies in our scenario.*

Related Work

We have presented a methodological approach to policy refinement that consists of three main elements: defining policy hierarchies that exploit inherent relationships in hierarchical systems; identifying high-level goals and their relationship with the policies along the policy hierarchy; and a policy refinement framework that considers the two previous elements for dealing with the critical nature of refining policies from administrative guidelines systematically. To the best of our knowledge, no other work has addressed this complete view of policy refinement, thereby capturing the requirements, processes, actors, and phases involved.

Most of the research efforts have concentrated on refining policies from abstract requirements. Work by Bandara *et al.* [1] has been a major contribution to the field. They propose an approach for transforming both policy and system behavior specifications into event calculus (EC) notations from which abduction is used to derive strategies that would achieve abstract requirements. From these strategies, policies are encoded. While EC and abduction are used in the former to infer the actions that would achieve particular goals, our approach goes through automated state exploration to obtain system trace executions that fulfill temporally ordered lower-level goals. An advantage of our approach over the EC-based approach is that our framework addresses explicit temporal execution of goals, whereas this is not addressed in the EC-based approach.

Regarding functional approaches, POWER [8] is one of the few implementations hitherto presented in the literature. POWER refines policies from policy templates designed by an expert so that consultants can create business policies from them. While POWER is an environment designed to guide the user to choose policies from predesigned policy templates, our framework is a goal-oriented approach in which the consultant defines "particular" views of management, having the possibility to formulate any combination of views as required with no predefinition of policy template choices.

Conclusions

The innovative aspects of our methodological approach can be summarized as follows:

- We have provided the principles to define policy hierarchies intended to address the refinement process. In this sense, the hierarchical relationships of policy systems can be exploited in favor of policy refinement.
- We have identified the general considerations for the definition of high-level goals. In this respect, the identification of high-level goals and their relationships with policies controlling such goals allow the assessment of systematic policy refinement. It would be practically impossible to achieve systematic policy refinement with arbitrary relationships.
- We have laid down the above aspects in the context of a policy refinement framework. We have demonstrated how these should be used to produce executable policies committing to administrative views.
- The validity of the methodology has been demonstrated through a scenario applied to the QoS management domain.

It is feasible to address policy refinement with the methodology presented in this article. Nevertheless, we must acknowledge that policy refinement is a complex issue that certainly deserves more attention. For instance, essential

We hope that the ideas presented in this article may encourage policy designers and researchers to address the policy refinement problem in a similar way in other application domains. Policy refinement is still at its initial stage; hence, substantial efforts should be made to solve it.

processes are carried out with human intervention and additional analysis techniques should be integrated in order to reduce or if possible avoid potential human mistakes.

So far, our approach does not consider feedback mechanisms to analyse the impact of high-level goals with respect to managed system performance and customers' behaviors [9]. This is a critical and challenging issue that may imply the integration of other mechanisms to relate managed system performance and customers' behaviors with "particular" high-level goal achievement. Future work will also be directed to explore this research area.

We hope that the ideas presented in this article may encourage policy designers and researchers to address the policy refinement problem in a similar way in other application domains. Policy refinement is still at its initial stage; hence, substantial efforts should be made to solve it.

Acknowledgments

This article was supported in part by the EC IST-EMANICS Network of Excellence (#26854).

References

[1] A. Bandara *et al.*, "A Goal-Based Approach to Policy Refinement," *IEEE Policy Wksp.*, June 2004.
[2] J. Rubio-Loyola *et al.*, "Using Linear Temporal Model Checking for Goal-Oriented Policy Refinement Frameworks," *IEEE Policy Wksp.*, June 2005.
[3] S. Calo and J. Lobo, "A Basis for Comparing Characteristics of Policy Systems," *IEEE Policy Wksp.*, June 2006.
[4] J. Moffet and M. Sloman, "Policy Hierarchies for Distributed Systems Management," *IEEE JSAC*, Dec. 1993.
[5] C. Bartolini, M. Sallé, and D. Trastour, "IT Service Management Driven by Business Objectives: An Application to Incident Management," *IEEE/IFIP NOMS*, Apr. 2006.
[6] J. Rubio-Loyola *et al.*, "A Functional Solution for Goal-Oriented Policy Refinement," *IEEE Policy Wksp.*, June 2006.
[7] P. Trimintzios *et al.*, "A Management and Control Architecture for Providing IP Differentiated Services in MPLS-Based Networks," *IEEE Commun. Mag.*, May 2001.
[8] M. Casassa, A. Baldwin, and C. Goh, "POWER Prototype: Towards Integrated Policy-Based Management," *NOMS 2000*.
[9] S. Parekh *et al.*, "Managing the Performance Impact of Administrative Utilities," *IEEE DSOM*, Oct. 2003.

Biographies

JAVIER RUBIO-LOYOLA (jrloyola@tsc.upc.edu) holds a degree in communications and electronics engineering, and a Master's degree in digital systems, both from Instituto Politécnico Nacional de México. He is currently working toward his Ph.D. at Universitat Politècnica de Catalunya (UPC). His interests are in formal methods applied to policy-based management and reactive systems analysis.

JOAN SERRAT-FERNANDEZ (serrat@tsc.upc.edu) received his degree of telecommunication engineer in 1977, and his doctorate degree in telecommunication engineering in 1983, both from UPC. Currently he is an associate professor at UPC, and his topics of interest are in the field of network management and service engineering.

MARINOS CHARALAMBIDES (M.Charalambides@surrey.ac.uk) received a B.Eng. (First Class Hons.) in electronic and electrical engineering, and an M.Sc. (Distinction) in communications networks and software, both from the University of Surrey, United Kingdom, in 2001 and 2002, respectively. He is currently a Ph.D. candidate at the Centre for Communication Systems Research, University of Surrey, and his research interests are in the areas of policy-based management, policy analysis, and IP quality of service.

PARIS FLEGKAS (P.Flegkas@surrey.ac.uk) received a Diploma in electrical and computer engineering from Aristotle University, Thessaloniki, Greece, an M.Sc. in telematics, and a Ph.D. from the University of Surrey in 1998, 1999, and 2005, respectively. From 2001 to 2005 he was a research fellow at the Centre for Communication Systems Research (CCSR), University of Surrey, working on European and U.K. national research projects. He is currently a collaborative researcher at CCSR, and his research interests are in the areas of policy-based networking, network and servicve management, traffic engineering, and IP quality of service.

GEORGE PAVLOU (G.Pavlou@surrey.ac.uk) is professor of communication and information systems at the Centre for Communication Systems Research, Department of Electrical Engineering, University of Surrey, where he leads the activities of the Networks Research Group. He received a Diploma in electrical and mechanical engineering from the National Technical University of Athens, Greece, and M.Sc. and Ph.D. degrees in computer science from University College London, United Kingdom. His research interests focus on network management, networking, and service engineering, including policy-based management, traffic engineering, multimedia service control, and object-oriented communications middleware. He has been instrumental in a number of European and U.K. research projects and has contributed to standardization activities in ISO, ITU-T, and IETF. He was the technical program co-chair of the Seventh IFIP/IEEE Integrated Management Symposium (IM 2001).