Gavin A. Campbell and Kenneth J. Turner. Goals and Policies for Sensor Network Management.
In M. Benveniste, B. Braem, C. Dini, G. Fortino, R. Karnapke, J. L. Mauri and M. S. H. Monsi, editors,
Proc. 2nd International Conference on Sensor Technologies and Applications,
pages 354-359, IEEE Computer Society, Los Alamitos, California, August 2008.

Goals and Policies for Sensor Network Management

Gavin A. Campbell and Kenneth J. Turner
Department of Computing Science and Mathematics
University of Stirling
Stirling FK9 4PA, Scotland, UK
Email: gca | kjt @cs.stir.ac.uk

***Abstract*—This paper describes a goal-directed, policy-based approach to managing sensor networks in the context of a wind farm. It describes the use of policies to enable end-users to detect and handle events, and discusses how high-level goals can be used to configure both the sensor network and the monitored system. Such use of goals and policies for sensor network management is a new area of research. While this paper discusses the approach in relation to the wind farm domain, the frameworks described are applicable to sensor network management in general.**

I. Introduction

Within a wind farm environment, turbines are prone to mechanical failures and degradation caused by (extreme) weather conditions and the stress this can cause. The downtime associated with these issues reduces the energy and, in turn, revenue generated from the site. Deploying sensors to detect potential faults before they occur is highly beneficial in what are often remote and challenging environments, where human observation is difficult and expensive. This paper outlines a goal-directed approach to proactive condition-monitoring of sensor networks, in the domain of wind farm operation. The work forms part of the PROSEN project (Proactive Condition Monitoring of Sensor Networks, *http://www.prosen.org.uk*). The use of high-level goals to direct system configuration is a new and novel approach to sensor network management. The use of policies in sensor network management is also new. Both goals and policies offer considerable control over a system. The approach is also usable by non-technical domain experts, such as wind farm operators.

A. Background and Related Work

Historically, policies have been used for systems administration purposes, for tasks such as system security and access control. More recently, policies have been adopted in a wider range of domains such as telephony [1]. A variety of policy languages and systems exist. The ACCENT policy system (Advanced Component Control Enhancing Network Technologies, *www.cs.stir.ac.uk/accent*), is a generic, highly flexible framework and is the basis of the work reported here. Other well-known policy systems include Rei (*http://rei.umbc.edu*) and Ponder [2]. A comparison between popular policy-based approaches is given in [3].

Wireless sensor networks have been employed in many different areas to date, but there has been little research using goal or policy-based techniques to manage them. A policy-based approach to managing wireless body sensor networks for health monitoring has been developed [4]. However, despite a common approach of using policies to achieve proactive configuration, this policy framework is designed to run on portable devices (such as PDAs) and is not extensible for large-scale distributed sensor networks on a wind farm. In addition, the target users are the individuals being monitored, whereas the system presented in this paper is geared toward multiple administrative operators of a wind farm.

Goal-directed systems have been developed extensively by the Artificial Intelligence (AI) community, and are typically utilised in agent-based systems and robotics. Goals have also been used extensively in Requirements Engineering (RE), to derive software specifications in system design [5], [6]. Goal-directed configuration in the context of sensor networks and wind farm operation is entirely novel.

Goal to policy refinement is a relatively new topic, with limited evidence of previous work in this area. A formal method of defining goals and refinement into policy sets is presented in [7]. That work is formally based, whereas the current paper report a practical and pragmatic approach that exploits domain knowledge in the form of ontologies.

In summary, techniques for policy-based systems, goals, goal to policy refinement, and sensor network management have been researched individually to varying degrees. However, as far as the authors are aware, no previous work has combined and integrated these ideas in the field of sensor networks (and wind farm management in particular).

B. Paper Outline

Section II discusses policy-based management, and explains the policy system and policy language the paper is based on. Section III discusses how policies are used in high-level management and configuration of sensor networks. Section IV introduces an approach to goal-based management of sensor networks, and explains how goals are defined and also refined into policies. Section V rounds off the paper with a summary and a discussion of future work.

II. Policy-Based Management

A policy is defined in a high-level language that specifies the syntax and semantics of policy constructs. Policy-based management techniques have typically been developed by others for purposes such as access control, quality of service, and security. However, the management of sensor networks is significantly different in a number of respects.

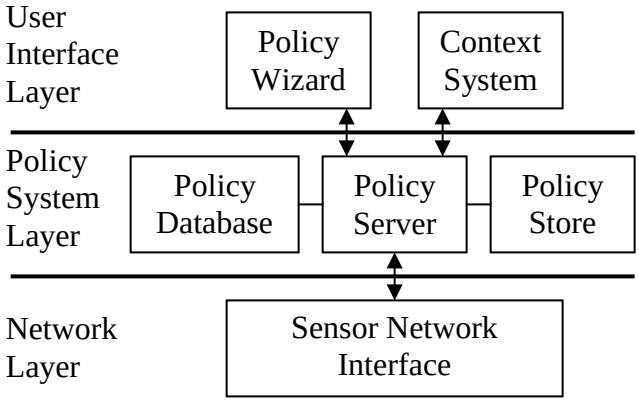

Fig. 1. ACCENT System Architecture

The work reported here makes use of the ACCENT policy system (Advanced Component Control Enhancing Network Technologies, *www.cs.stir.ac.uk/accent*). In turn, this supports the APPEL policy language (Adaptable and Programmable Policy Environment and Language, *www.cs.stir.ac.uk/appel*). The following sections describe ACCENT system architecture, the APPEL policy language, and how both have been adapted for sensor network management.

A. Accent Policy System

The ACCENT policy system was originally designed for the domain of (Internet) telephony, providing user-defined handling of call preferences. The system has three conceptual layers, as shown in figure 1. At the lowest level, the system layer connects the policy system to its external environment. The policy system layer is responsible for enforcing policies. It consists of a policy server, policy store (where policies reside) and a policy database (containing user login and server configuration data). At the top level, the user interface layer provides a mechanism for users to create policies and a means to obtain contextual information. The ACCENT architecture is described in [1].

APPEL is a policy language used to express policies within ACCENT. The language is extremely flexible, comprising a core structure which can be extended for any application domain. Originally, APPEL was tailored for telephony and policy conflict resolution, but has since been extended and specialised for a number of new areas, including sensor networks and home care. The APPEL language specification can be found in [8].

The core of APPEL defines the overall structure of a policy document, including regular policies, resolution policies and policy variables. A policy consists of one or more rules in ECA form (Event-Condition-Action). Each rule has a combination of optional triggers, optional conditions, and mandatory actions. These constructs are extended for new applications with domain-specific triggers, conditions and actions.

A policy is eligible for execution if its triggers occur simultaneously and its conditions apply. Additional conditions may be imposed, such as the time period during which the policy applies. When the policy system is informed of an event, the applicable policies are retrieved and applied if eligible. As multiple policies can be triggered, conflicts may arise among their actions. Such conflicts are handled at runtime using resolution policies. Resolution policies have a very similar structure to regular policies, except that their triggers are a conflicting pair of regular policy actions. Policy conflict and resolution in ACCENT is discussed in [9], while filtering for conflict-prone policies is described in [10].

B. Policy Wizard User Interface

APPEL policies are specified in XML. To make the language accessible to non-specialists, a variety of policy wizards are available. For example, a web-based policy wizard [11] allows policies to be defined using near-natural language. This wizard retrieves policies in XML, renders them in a user-friendly way, allows them to be edited easily, and finally stores them again in XML. The wizard supports multiple natural languages and may easily be extended for others. In general, the policy wizards allow domain experts to make effective use of the policy system.

C. Ontologies for Policy Language Modelling

An ontology is the set of terms used to describe and represent an area of knowledge, together with the logical relationships among these. OWL (Web Ontology Language [12]) is a standardised, XML-based ontology language that models a domain using classes (concepts) and properties (the relationships between concepts). Using OWL, a framework of ontologies was devised to support APPEL. The framework models the core language constructs (*GenPol*), how such constructs are categorised and displayed within the policy wizard (*WizPol*), and specialisations of the language for various domains (e.g. *SenPol* for sensor networks). OWL supports ontology inheritance, allowing ontologies to extend one another through importing. Consequently, the domain-specific aspects of APPEL can be easily modelled in a new ontology by importing the generic language model.

The web-based policy wizard was originally hard coded with options specific to call control. In previous work [13], [14], the policy wizard was re-engineered and generalised for use with any domain specialisation of APPEL using the ontology framework just described. Hard-coded call control options (including specific triggers, conditions, actions and variables) were removed and specified in an ontology. When accessed, the wizard dynamically generates policy options based on information drawn from the appropriate ontology.

To integrate ontology information within the policy wizard, previous work developed a stand-alone server named POPPET (Policy Ontology Parser Program Extensible Translation), as shown in figure 2.

POPPET parses an ontology document at a given URL, reasons about its contents, and builds a model of its structure. Any client application (such as the policy wizard) may then interrogate the stored ontology model using a variety

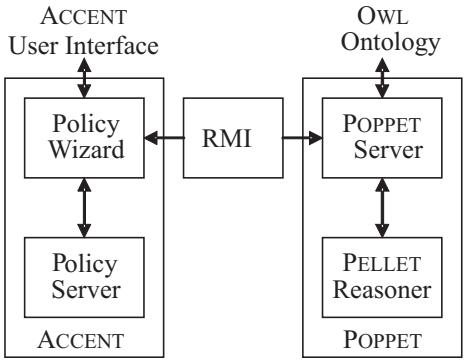

Fig. 2. Ontology Integration using POPPET

of generic methods. Communication with a policy wizard is achieved using Java RMI (Remote Method Invocation). Although the ontologies were originally designed to drive the policy wizards, a domain-specific ontology may contain unlimited information about the domain area. This additional information is used for goal modelling and goal refinement as described later.

III. POLICIES FOR SENSOR NETWORKS

The policy-based approach described here is fully implemented and being tested with a sensor network in a real wind farm site as part of the PROSEN project. Sensor network policies are defined using a combination of generic and domain-specific triggers, conditions and actions.

Generic constructs are handled internally by the ACCENT system. Generic triggers include timer expiry, general conditions include time-based restrictions, and general actions deal with logging, timers and variables.

Specific constructs for sensor networks are used to interface to the managed system. The approach is designed to be simple but highly flexible. Sensor network constructs include the following:

- A *device_in* trigger reports incoming events from any external component (whether hardware, software or service). The first trigger parameter is mandatory to specify the event type. Four optional trigger parameters deal with the entity, entity instance, when the event occurred, and a list of message values.
- Conditions can use the trigger parameters.
- A *device_out* action is used to configure and control external components (again of any type). The action parameters are similar to those for triggers, the first being the action type, with the others specifying the entity, entity instance, when to execute the action, and a list of message values.

Using these language constructs, sensor network policies can be triggered by events such as sensor measurements, warning alerts, network status information, and commands issued by an operator (say, via a console). Policy conditions can be based on any trigger parameter such as the event type (e.g. wind speed, gearbox oil temperature, sensor battery alert) or the entity reporting the event (e.g. the operator console, a particular sensor node, a software agent). Policies can also depend on the particular message values.

Examples are provided in this paper of policies and goals. These are mostly given in their native XML form. However, in practice a policy wizard is used to allow user-friendly definition and editing of policies.

The following policy applies to domain *farm3.wind.com*, i.e. wind farm 3. It alerts the operator if the average speed from any anemometer is reported to be above 20m/s for at least one hour. No entity name or instance is given in the action because operator messages are for the default console. In general there may be several trigger parameters, but just a single value (the wind speed) is used here. Note that this policy deals with a single wind speed report. To deal with multiple reports, the trigger history would need to be checked.

```
<policy owner="admin" applies_to="@farm3.wind.com"
 id="Gusts" enabled="true" changed="2008-04-15T10:20:59">
  <policy_rule>
    <trigger arg1="average_speed" arg2="anemometer">
      device_in(arg1,arg2)
    </trigger>
    <conditions>
      <and/>
      <condition>
        <parameter>message_period</parameter>
        <operator>ge</operator>
        <value>60</value>
      </condition>
      <condition>
        <parameter>parameter_values</parameter>
        <operator>gt</operator>
        <value>20</value>
      </condition>
    </conditions>
    <action arg1="operator_output" arg5="Repeated gusts">
      device_out(arg1,,,,arg5)
    </action>
  </policy_rule>
</policy>
```

As an example of a resolution policy, conflicts can arise where actions cannot be performed simultaneously due to underlying sensor limitations (e.g. on processing, memory, electrical power or bandwidth). The following checks for conflicts between the actions *get_log* and *sensor_check*. The actual parameters of these actions (*arg2* for entity name, *arg3* for entity instance) are bound to resolution variables. If these pairs are identical, the action with the stronger preference is applied. (A variable is substituted into an expression by preceding its name with ':'.)

```
<resolution owner="admin" applies_to="@farm3.wind.com"
 id="Resource conflict" enabled="true"
 changed="2008-04-13T20:40:00">
  <policy_rule>
    <triggers>
      <and/>
      <trigger arg2="variable2" arg3="variable4">
        device_out(get_log,arg2,arg3)
      </trigger>
      <trigger arg2="variable3" arg3="variable5">
        device_out(sensor_check,arg2,arg3)
```

```
      </trigger>
    </triggers>
    <conditions>
      <and/>
      <condition>
        <value>:variable2</value>
        <operator>eq</operator>
        <value>:variable3</value>
      </condition>
      <condition>
        <value>:variable4</value>
        <operator>eq</operator>
        <value>:variable5</value>
      </condition>
    </conditions>
    <action>apply_ stronger</action>
  </policy_rule>
</resolution>
```

IV. Goals for Sensor Networks

A goal is a high-level principle governing how a system should behave. Goals are a way of controlling system behaviour, and can be used for autonomous system management based on high-level operational and strategic aims. Achieving a goal means reaching a desired state (or set of acceptable states). In an Artificial Intelligence (AI) approach, this is typically accomplished by constructing a plan that uses the procedural actions available to the system. In a policy-based approach, goals are achieved by determining which policies will ensure the aims are met. This section outlines an approach for goal-directed management of sensor networks for wind farm monitoring.

A. Example Goals

For sensor network management, two aspects can be managed via goals and policies. One is the sensor network itself, including configuration of data sampling rates, data logging, data transmission frequency and other sensor settings. The other is management of the monitored system. For the work of this paper, the monitored system encompasses the wind farm as a whole, including individual turbines and other monitored components within its general environment – physical or otherwise.

Goals may therefore focus on managing the sensor network or the wind farm. For example, a sensor network goal might be to conserve battery life in sensor nodes, while a wind farm goal might be to minimise turbine downtime. Refinement of goals relating to the monitored system may result in policies for managing the sensor network. For example, minimising turbine downtime might require maximising sensor data on turbine gearboxes so as to anticipate problems. Goals typically fall into the following categories:

- for many kinds of systems: maximise revenue, minimise component degradation, reduce unplanned downtime, etc.
- for sensor networks: maximise battery life, make best use of bandwidth, deal with intermittent communications, etc.
- for condition monitoring: maximise value of measurement data, minimise spurious alerts, optimise data quality and timeliness, etc.
- for wind farms: reduce blade vibration, limit turbine rotational speed, avoid damage in gusty conditions, etc.

Broadly speaking, there are two types of goals the system might want to achieve. The first kind of goal typically involves maximisation (e.g. of sensor battery life) or minimisation (e.g. of unplanned maintenance), possibly subject to constraints. There is usually a space of possible solutions that need to be optimised in some fashion. The second kind of goal aims to alter system behaviour to achieve a desirable state (e.g. stable power output from a wind farm despite varying wind conditions).

B. Goal System Requirements

A goal-directed approach requires the following:

- a language to express goals, including the means to express constraints on goals
- a method of goal refinement to select and define policies that realise the goals
- domain information and system state information to help in the refinement.

The following subsections describe how APPEL is used to express goals for the work described here. System state can be captured with the aid of domain ontologies and goal constraints, while refinement of goals to policies can be achieved using optimisation approaches combined with a domain ontology.

C. Goal Definition

Following a study of formal techniques and AI-based frameworks used in existing goal-based systems, no approach was found to suit the needs of goals for the sensor network domain. Consequently, APPEL was adapted for goal definition. Defining goals based on this has several advantages. Primarily, it allows sharing and reuse of the existing ACCENT framework and tools:

- a policy wizard: to define and edit goals and their associated policies
- the conflict analyser: for offline determination of conflicts among goals and policies
- the policy server: for run-time policy execution and conflict resolution
- the policy store: for storage and retrieval of goals and policies.

In addition, defining goals in a similar format to the target policies that they refine into allows a completely consistent approach.

Using APPEL, a goal is expressed as a policy without a trigger. A goal is therefore always available for execution, subject to meeting any constraints that appear in conditions. Like a policy, a goal has a mandatory action – the goal statement itself. Optional conditions are constraints which must be satisfied for the goal to be achieved. A constraint typically specifies some value or range for one or more system variables. This effectively defines the set of states the system must be in for the goal to be achieved.

As an example, the goal for operating a wind farm at night may be to operate it cautiously (reducing the risk of breakdown), and to reduce power drain on the sensor network (which cannot use solar cells during this period). The constraints on the goal are that it should apply between 8PM and 8AM, and that operational turbines should generate power in the range 200kW to 600kW.

```
<goal owner="admin" applies_to="@farm3.wind.com"
 id="Night time" enabled="true"
 changed="2008-04-12T21:43:15">
  <policy_rule>
    <conditions>
      <and/>
      <condition>
        <parameter>time</parameter>
        <operator>in</operator>
        <value>20:00..08:00</value>
      </condition>
      <condition>
        <parameter>turbine_output</parameter>
        <operator>in</operator>
        <value>200..600</value>
      </condition>
    </conditions>
    <actions>
      <and/>
      <action>maximise(turbine_life)</action>
      <action>minimise(sensor_drain)</action>
    </actions>
  </policy_rule>
</goal>
```

D. Goal to Policy Refinement

Goal refinement is the process of incrementally breaking down a goal into one or more executable policies that achieve it. These policies are expressed in the sensor network dialect of APPEL described earlier. Note that goals do not directly invoke actions. Only during policy execution do events lead to actions that manage the monitored system and the sensor network.

Policies derived from goals have the same structure as ordinary (user-defined) policies defined by users. However, they need to be kept distinct since goal-derived policies are related only to the goals they achieve. If a goal is changed or removed, the policies generated from it are also affected. From the viewpoint of the policy server, however, ordinary policies and those derived from goals are stored and executed identically.

Goals for sensor networks are not straightforward aims of achieving some state. Instead, they specify aspects of the monitored system that must be optimised. There is no single course of actions for the system to perform. Refinement leads to a set of policies that best achieve the goal.

Optimisation selects a set of suitable policies that when instantiated achieve the desired goal. Besides simply selecting policies, refinement may also parameterise them. For example, thresholds may be set for alarms, or limits may be set for acceptable operation of the wind turbines.

The conditions of goal execution are specified as constraints. These specify a range of values for system variables. If all constraints are deemed satisfiable, the goal is achievable. System variables in this sense represent state that may be influenced by goals. Examples include operational cost, system downtime, component lifetime, and equipment noise levels.

Goals and policies have effects on system variables. This kind of information is domain-dependent, and is formulated in the ontology for sensor networks. This contains information about:

- the factors that contribute towards goals
- how constraints are affected by these factors
- the contribution (qualitative or quantitative) of policies towards these factors.

Goal refinement is treated as an optimisation problem. It is necessary to select and parameterise a collection of policies that optimally achieve some goal. This is measured by an objective function that evaluates a potential solution in the form of a collection of policies. The domain ontology allows the fitness of this solution to be judged in terms of the contributions by policies to the goal characteristics that they aim to realise. The optimisation therefore requires selecting and combining predefined policies (some of which may also be parameterised). Goal constraints impose boundaries on acceptable regions of the solution space.

Consider the goal for night-time operation of a wind farm given in the previous subsection. This can be refined into a number of policies such as the following (given here in English for clarity and brevity):

- when wind speeds exceed 25m/s, take higher-up turbines out of service
- if the amplitude of turbine blade vibration exceeds 1mm, decrease blade pitch by 10°
- if a turbine's output falls below 200kW, take the turbine out of service
- if a turbine's output exceeds 600kW, decrease blade pitch by 20°
- if solar power for a sensor node falls to 20% of normal, send measurements in blocks every 30 minutes
- if battery power for a sensor node falls below 75% of capacity, reduce sensor measurements to once every 10 minutes.

V. CONCLUSION

The paper has presented new research into policies and goals for proactive management of sensor networks in general, and for wind farms in particular. The ACCENT policy system and its associated APPEL policy language have been adapted and specialised for sensor network management. Policies support automated, but user-defined, management and configuration of systems. Goals are high-level system objectives that are refined into policies that realise them. Similar to policies, goals are defined in APPEL and share the existing mechanisms of the ACCENT policy system. For example, the existing wizards are easily extended to support goal definition in a manner similar to that for policies.

Goals for sensor network management commonly define aspects of the system to be optimised. These goals can be

achieved using optimisation techniques in conjunction with ontology-defined state information. The benefit of using goals and policies in this way is to give domain experts the means of managing a system based on high-level operational and strategic aims.

Conflicts are detected and resolved only at the level of policies, not goals. Further work might extend this to detect and resolve goal conflicts at an earlier stage in the refinement process. It would also be useful to have automated explanation for a human operator of why certain policies were chosen as a result of goals, or of why certain actions were performed as a result of policies. The goal refinement strategy presented in this paper will be evaluated on two wind farm test sites.

ACKNOWLEDGEMENTS

The authors thank their colleagues on the PROSEN project for discussions that helped shape work presented in this paper. They also thank Stephan Reiff-Marganiec (now at the University of Leicester) and Lynne Blair (who was on leave from Lancaster University during development of ACCENT), for their substantial contribution to the original policy system. Gavin Campbell was supported in this work by grant C014804 from the UK Engineering and Physical Sciences Research Council.

REFERENCES

[1] K. J. Turner, S. Reiff-Marganiec, L. Blair, J. Pang, T. Gray, P. Perry, and J. Ireland, "Policy support for call control," *Computer Standards and Interfaces*, vol. 28, no. 6, pp. 635–649, June 2006.

[2] N. Damianou, N. Dulay, E. C. Lupu, and M. Sloman, "Ponder: A language specifying security and management policies for distributed systems," Imperial College, London, UK, Tech. Rep. 2000/1, 2000.

[3] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok, "Semantic web languages for policy representations and reasoning: A comparison of KAoS, Rei, and Ponder," in *Proc. 2nd Int. Conf. on The Semantic Web*. Berlin, Germany: Springer, 2003, pp. 419–437.

[4] S. Keoh, K. Twidle, N. Pryce, E. Lupu, A. S. Filho, N. Dulay, M. Sloman, S. Heeps, S. Strowes, and J. Sventek, "Policy-based management for body-sensor networks," in *Proc. 4th Int. Workshop on Wearable and Implantable Body Sensor Networks*, Aachen, Germany, Mar. 2007, pp. 92–98.

[5] D. Alrajeh, A. Russo, and S. Uchitel, "Inferring operational requirements from scenarios and goal models using inductive learning," Shanghai, China, May 2006.

[6] E. Letier and A. van Lamsweerde, "Deriving operational software specifications from system goals," in *Proc.10th ACM SIGSOFT Symp. on the Foundations of Software Engineering*, Charleston, South Carolina, Nov. 2002.

[7] A. Bandara, "A formal approach to analysis and refinement of policies," Ph.D. dissertation, Imperial College, London, July 2005.

[8] K. J. Turner, S. Reiff-Marganiec, L. Blair, G. A. Campbell, and F. Wang, "APPEL: The ACCENT project policy environment/language," Department of Computing Science and Mathematics, University of Stirling, UK, Tech. Rep. CSM-161, Sept. 2007.

[9] K. J. Turner and L. Blair, "Policies and conflicts in call control," *Computer Networks*, vol. 51, no. 2, pp. 496–514, Feb. 2007.

[10] G. A. Campbell and K. J. Turner, "Policy conflict filtering for call control," in *Proc. 9th Int. Conf. on Feature Interactions in Software and Communications Systems*, L. du Bousquet and J.-L. Richier, Eds. Amsterdam, Netherlands: IOS Press, May 2008, pp. 83–98.

[11] K. J. Turner, "The ACCENT policy wizard," Department of Computing Science and Mathematics, University of Stirling, UK, Tech. Rep. CSM-166, Dec. 2005.

[12] World Wide Web Consortium, *Web Ontology Language (OWL) – Reference*, ser. Version 1.0. Geneva, Switzerland: World Wide Web Consortium, Feb. 2004.

[13] G. A. Campbell, "Overview of policy-based management using POPPET," Department of Computing Science and Mathematics, University of Stirling, UK, Tech. Rep. CSM-168, June 2006.

[14] G. A. Campbell and K. J. Turner, "Policy conflict filtering for call control," in *Proc. 9th Int. Conf. on Feature Interactions in Software and Communications Systems*, L. du Bousquet and J.-L. Richier, Eds. France: IMAG Laboratory, University of Grenoble, Sept. 2007, pp. 93–108.