Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 45 (2004) 585–603
COMPUTER NETWORKS
www.elsevier.com/locate/comnet

Policy-enabled mechanisms for feature interactions: reality, expectations, challenges

Petre Dini [a], Alexander Clemm [b], Tom Gray [c], Fuchun Joseph Lin [d], Luigi Logrippo [e,*], Stephan Reiff-Marganiec [f]

[a] *Cisco Systems, Inc., 725 Alder Drive, Bldg 20/3, Milpitas, CA 95035, USA*
[b] *Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706, USA*
[c] *PineTel, 184 Chemin lac de la truite, Fort-Coulonge, Quebec, Canada J0X 1V0*
[d] *Mobile Networking Research, Telcordia Technologies, RRC (Room 1T221), One Telcordia Drive, Piscataway, NJ 08854-4157, USA*
[e] *Département d'informatique et ingénierie, Université du Québec en Outaouais, Gatineau, Québec, Canada J8X 3X7*
[f] *Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH, UK*

Available online 22 March 2004

Abstract

This paper is based on the discussion during a panel that took place at the 7th Workshop on Feature Interactions in Telecommunications and Software Systems in Ottawa, Canada, June 2003. It presents a holistic picture on two paradigms, namely *feature* and *policy*, and their intertwining. The guest panelists brought examples from complementary areas and presented their experiences on using the concept of policy for defining features and for treating the feature interaction problem. The intrinsic interactions commonly called policy *conflicts* within policy-based systems were also discussed. The panelists considered methodological issues, such as the use of deontic logic and the representation of features through policies, as well as industrial applications, such as service provisioning and policy-based management applications for service bundling. They also brought out different views that reflect some disparity between the communities involved in this research.

Keywords: Telecommunications features; Policies; Feature interactions; Formalisms; Service provisioning; Service bundling

1. Introduction

Since the early 90s, the problem of feature interactions has been recognized as crucial in the design and implementation of telecommunications distributed systems. As distributed systems evolve in terms of features, technology, complexity, and size, the feature interaction problems expand, become distributed and more complex, and require sophisticated solutions. Fundamentally, feature interactions arise in the creation, maintenance, and evolution of new services (telephony, electronic commerce, Web services, multimedia, banking, etc.) and in the implementation of these services

* Corresponding author.
E-mail addresses: pdini@cisco.com (P. Dini), alex@cisco.com (A. Clemm), digroup_codec@direcway.com (T. Gray), fjlin@research.telcordia.com (F.J. Lin), luigi@uqo.ca (L. Logrippo), srm13@leicester.ac.uk (S. Reiff-Marganiec).

1389-1286/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2004.03.001

across distributed, sometimes heterogeneous, platforms. Feature interactions also relate to software maintainability, software verification and validation, software documentation, and the software development process.

While it is advocated that policy-based management systems become the pillar technology of self-managing and autonomic computing systems, problems with the paradigm itself are being debated in the policy community. There is much industrial and academic research on the theory and practice of policy-based systems. From a practical perspective, there is a tendency of oversimplifying the use of the policy paradigm and neglecting feature interaction problems. Several existing usage paradigms are particularly tailored to given areas (security, intrusion detection, monitoring, etc.). This approach makes policy conflicts easier to handle. Further research is needed for inter-applications, inter-networking, and inter-features applications of the policy paradigm. As multiple formalisms to express policies exist, conflicts between these formalisms must also be explored. It should be noted also that there are no common tools for detecting and solving policy conflicts, especially when policies are expressed in different languages or approaches.

Several observations led to this paper, which is a result of the debate in a panel held during the Feature Interaction Workshop in Ottawa, 2003 [2]. Logistically, the concepts and issues pertaining to the title of the paper are debated in two separate communities, namely the ''feature interactions'' one and the ''policy-based systems'' one, each of them having their own fora and dedicated workshops. While most of the target-problems are similar, solutions, approaches and directions are not yet synchronized. Hence, the main scope of the panel was to discuss the state of the art and identify future ways to leverage the technical achievements of these two paradigms.

It is important to note that this paper is a digest of different viewpoints expressed during the panel. The panelists agreed on the importance of the new work on policies, but emphasized their own personal views on the basis of their experiences and vision. Therefore, this paper should not be read as a research paper. The positions expressed in the paper are individual ones and do not attempt to provide a comprehensive synthesis on the state of the art.

This panel report is organized as follows. In Section 2, brief historical remarks and details of the concepts used in the areas of feature interactions and policy-based systems substantiate the relevance of the core questions. Section 3 presents some concepts used in the two areas, identifying several technical key points. In Section 4, two industrial applications are presented, where both feature interaction and policies play important roles, i.e., service provisioning and policy-based management applications. Sections 3 and 4 reflect the views brought forward by the panelists and thus show snapshots of the current state of the art and perspectives, from different angles. Section 5 concludes with the panelists' position on the two paradigms and identifies common research and application issues.

The panel presentations on which this paper is based are published at http://www.site.uottawa.ca/fiw03/.

2. Features versus policies, and the panel questions

This section presents some research issues and positions current achievements in the two areas of feature interactions and policy systems. Basic concepts and definitions of feature interactions and policies are given in Sections 2.3 and 2.4.

2.1. Duality of the problem space

The first question relates to the semantic understanding of the target-domain. Both features and feature interactions, on one hand, and policies, on the other hand, are related to the creation, maintenance, and evolution of new services (telephony, electronic commerce, Web services, multimedia, banking, etc.) and to the implementation of these services across distributed, sometimes heterogeneous, platforms. Most of the mechanisms proposed to support new service creation refer to conflict avoidance, detection, and resolution in specifying new features. Static and dynamic mechanisms have been proposed, and industrial tools exist.

Policies refer mainly to composite behavior, a policy being a mechanism to specify the composition of behavioral constraints; an example of this is setting a priority among two conflicting features provided by composed parts. Commonly, policies include references to triggering conditions and appropriate actions. By their nature, policies (when composed) may embed constraint conflicts and action conflicts (for one single policy with multiple actions, or between actions specified by different policies).

This latter aspect raises a second question related to the nature of policy conflicts. On one hand, policies are used to design, specify, and implement techniques to detect and fix feature interaction problems. On the other hand, feature interaction problems may originate from policies that may have inadvertent embedded conflicts, e.g., in the same system status, two policies may recommend either different or even conflicting actions.

In other words, policies have their own feature interactions: the so-called policy conflicts. These are inevitable, and so any policy system has to provide support for handling such conflicts [29]. For example, when a user attempts to upload a new policy, in principle this should be checked not only against other policies for the same user, but also against policies that the user might be subject to, e.g., due to her role in the enterprise, or due to contractual obligations. We can however only check for static interactions, i.e., those that are inherent to the policies, independent of changing contextual data. This suggests the use of offline detection methods and filtering techniques. Any inconsistency detected needs to be reported to the user. The resolution mechanism is typical of offline methods—a redesign of the policy base.

Some interactions can only be detected at run time, as might happen between policies of unrelated users, for example users in different domains that become involved in a transaction. The enforcement environment of policy architecture needs to be able to detect and resolve such conflicts.

The duality of the two paradigms of feature and policy is summarized in Fig. 1.

Let us assume a PAS (Policy-Aware System), a system that allows policies to monitor and manage its behavior, including a set of features {Fi}. A PS (Policy System) is intended to monitor and manage a PAS through a set of policies {Pi}. One of the following interaction loops may execute when a new policy or a new feature is specified:

(i) When a new policy P is specified in a PS, the feature interaction analysis with the existing {Pi} is performed, to detect a potential conflict;
(ii) When a new feature F is specified in a PAS, a policy or set of policies is executed to detect, and eventually solve [undesired] feature interactions;
(iii) In both cases, a special loop is performed to synchronize the PS and PAS with the new specified features; a new feature in PAS may

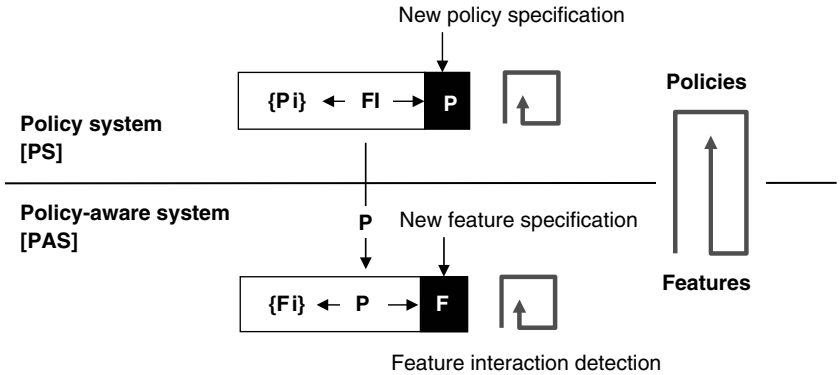

Fig. 1. Dual semantic of feature interactions and policy-aware systems.

require new policies in PS or the inhibition of existing ones, while a new policy in PS may require new feature interaction resolution methods.

2.2. Panel questions

It is important to mention that both paradigms are concerned with static and dynamic considerations in terms of conflicts. Some of the interactions can be captured at the design or definition phase, while others can be captured only at run time. Policies depend on much contextual data that may not be available at design time, and testing all possible combinations is not feasible. Also, at design time, one might not be aware of all the policies that could exist in a system; after all, no one single individual/company is in charge of defining all policies. These observations led to the following core questions for the debate:

- Is a policy a feature, or a solution for feature interactions?
- What formal tools/languages can be leveraged?
- Are all policy conflicts feature interactions?
- Can all feature interactions be solved through policies?
- Are there particular technologies where the marriage of policy and feature interactions is beneficial?
- What is the bridge between the two paradigms, currently debated in two distinct communities?

During the panel, the list of questions became even longer, with the following issues:

- Are agents a bridge paradigm between features and policies?
- What will be the relationship between the world of agent policies and the world of human law?
- What concepts and methods can be exchanged between the worlds of policy systems and legal systems?

Of course, the discussion of these topics could be only partial and fragmentary, and our account below includes some after-the-fact reflections of the panelists.

2.3. Feature interactions

The basic reference on feature interactions in telephony systems is often considered to be [8], and we assume that the reader is familiar with the topic.

A generally agreed definition of feature interaction does not exist, although there are several definitions which depend on the hypothesis and formalisms on which they are based. Informally, feature interaction is considered to be any new behavior resulting when combining two or more feature behaviors. Some interactions can be desirable; in fact they are required in order to obtain certain services. Others may be undesirable in the sense that they can reduce the usefulness of the service or prevent its proper operation. Some authors refer to these as positive and negative feature interactions. Another view, developed in [14] is that a negative feature interaction is essentially a logical contradiction among the requirements of various features, together with the requirements of the system. Although they might appear quite different, it turns out that these two definitions are equivalent in many cases.

Several methods have been proposed and published in the literature for specifying features and detecting feature interactions. We enumerate a few and focus on a more recent one discussed in the panel. For a more complete overview, the reader is referred to [6]. The ''online detection with automatic resolution'' and ''offline detection with manual resolution'' solutions have been presented in [24]. Kimbler presented a methodology for addressing the interaction problem at the enterprise level [22]. Dini and Bochmann proposed a run-time feature interaction resolution in an object-oriented environment, tailored for automatic reconfiguration [13]. Capellmann et al. proposed a comprehensive approach in service creation for consistent interaction detection [9]. An observer-based mechanism for detection and solving service interactions has been suggested by Aggoun and Combes [1]. Gray et al. implemented a policy-oriented system using context and intent in call processing, a feature that supports natural human collaboration within a human environment. Observers monitoring the behavior of users and

call processing might intervene to provide for proper behavior and prevent undesirable behavior and interactions [16].

Methods for capturing undesirable interactions have been tailored to specific domains. Recently, Zave proposed formalisms for ideal address translation and its applications, using a classification of feature interactions caused by address translation, along with principles (similar to policies) for evaluating these interactions as desirable or undesirable [31]. Special temporal dependencies for interaction detection are considered, although not explicitly, in [7,20].

2.4. Policy-driven systems

The complexity of today's distributed systems imposes a design discipline to capture behavioral constraints of interacting components according to the business logic. Many communities (IETF, DMTF, OMG, OASIS, TMF, etc.), academia (Imperial College, etc.) and industry (Evidian, ILOG, etc.) are focusing on general frameworks, languages, or easy-to-use products, respectively, for using policy-driven mechanisms to cope with this complexity.

As in the case of feature interaction, a general definition of policy does not exist. It is generally accepted that there are several layers of policies from the high-level business logic to the configuration of network devices. For the sake of simplicity, we refer hereafter to the concepts of business layer, design layer, and implementation layer:

- The *business layer* specifies the high-level goal (objective) in general terms, either in a plain natural language text form, or in a semi-structured specification form. In policy parlance, these are business policies, or high-level policies. Policy-based management applications implement this logic. As an example of goals, let us consider this example: "The following security degrees must be supported by a managed system when handling the management data: noAuthNoPriv, authNoPriv, and authPriv." Sometimes, the goals are less deterministic, where statements like "when possible..." or "it is recommended that..." require special design considerations.
- The *design layer* introduces a formal discipline, where decisions are captured in policies. Policies may be represented as special groupings of rules, with a particular dependency, tying the objective to targets and refining the objective in concrete actions, under certain conditions. For example, at the design level, different SNMP versions and authorization and authentication methods may be made more precise, e.g., SNMPv2c, noAuth, noPriv.
- Finally, the *implementation layer* is the bottom layer, where the decisions are implemented in terms of rules (conditions-actions) by the so-called "rule-engines". The rules derived from the policy specifications may use other status data provided to resources, entity access rights, conditions, security rules, or specify certain types of execution engines supporting the actions prescribed. Following the above example, SNMPGet, SNMPSet, DenyAuth, AcceptAuth, and so forth are concrete actions a policy may specify.

Fig. 2 presents an example from the security domain, where goals, policies and rules are identified. As a detail, translation between layers may be needed. In this figure, only one translation is represented, as an example.

We mention that the policy approach is commonly used for constraining the systems' behavior according to a series of objective functions. In a general sense, a policy is considered triggerable when all its pre-conditions are satisfied. Let us consider a simple form of a policy, e.g.,

$$\text{IF } \{\langle \text{condition\_1}\rangle \cdots \langle \text{condition\_}n\rangle\}$$
$$\text{THEN } \{\langle \text{action\_1}\rangle \cdots \langle \text{action\_}k\rangle\}.$$

The number of conditions may be unbounded and the conditions may be complex. Actions generally target distributed system entities and may have implicit or explicit dependencies among them.

As conditions and actions may be specific for different technology areas, many policy languages have been proposed, each covering one or several areas. The relevant work is published in the Policy Workshop series. The following few references are

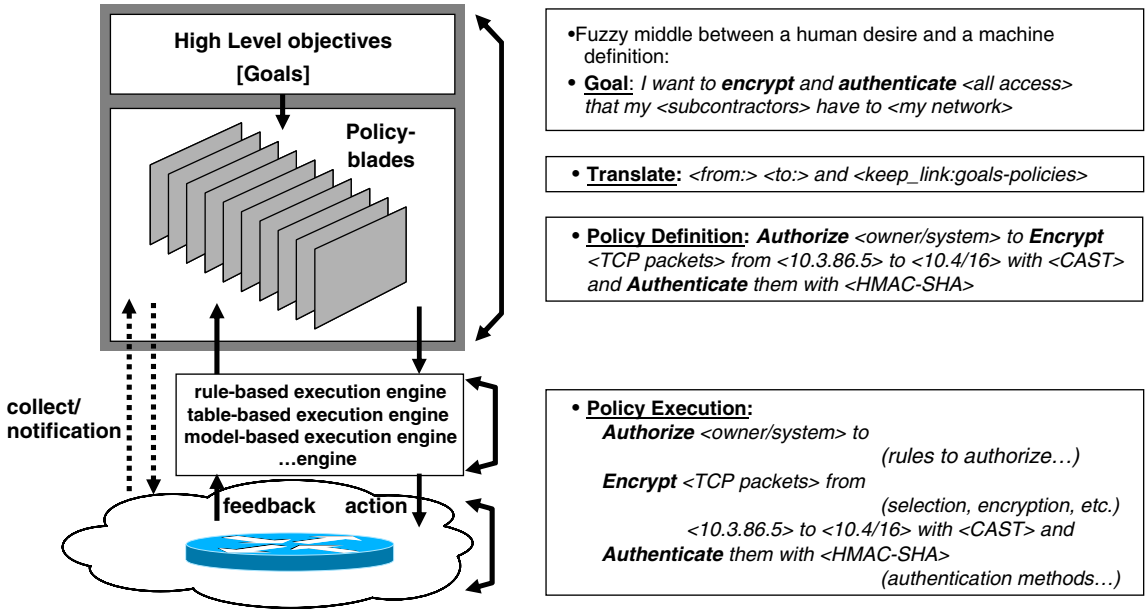

Fig. 2. A simple representation of policy layers.

intended to show the spectrum of the problem coverage. The basic reference on policy is the work done at Imperial College London by Sloman and his team [23]. Most policy languages have roots in the Ponder language [12]. An extension of the Ponder language, called Alloy, deals with delegation of obligations, where the main issues are the balance between authorization and obligation policies, the source of obligations and the reasons for delegation, and meta-policies for controlling the delegation of obligations [30]. The Chisel policy language deals with mobile-aware dynamic changes in the behavior of various services of the middleware and allows unanticipated behaviors at run time using behaviors as meta-types [21].

Policy combination and grouping may introduce policy conflicts. Several languages dealing with policy combination and policy groups have been proposed. Policy combination methods for many mutually dependent policies on specific purposes are formalized in [19]. A meta-policy language for controlling changes of a defined policy or set of policies under pre-defined constraints has been proposed for role-based access control [5]. An interesting concept called Policy of Policies has been proposed by Grandville et al.: the concept is used to orchestrate the deployment of dependent policies following a Policy FSM-based lifecycle, especially designed for time-sensitive configuration policies [15].

Temporal constraints may cause policy conflicts due to (i) the procedures of measurement and constraints evaluation and (ii) the dependencies among actions. For time-sensitive systems, the temporal relationships between policies themselves are important and must be captured. A suitable policy logic coping with the latter aspects was proposed in [26]. For the former issues, very few and partial results have been published, e.g., temporal connectives of action constraints have been proposed in the PDL language [10].

2.5. Techniques for policies and feature interactions

The current industrial trend is towards merging different existing communications technologies (such as videoconferencing, email, Voice over IP), with new technologies (such as home automation), and device control, together with a move to greater mobility (e.g., wireless communications, mobile

telephony, and ad hoc networking). As a result, the end user will be an always connected entity. However, users might not always wish to be disturbed, or at least not by everyone or for any type of inquiry. Future services need to provide support for users to control their *availability*, and availability is highly dependent on the *context* of the user.

We believe that services then will enable communication by allowing users to achieve particular goals. This leads to a situation where it is desirable for end-users to define their own services by writing policies for achieving a maximum level of customizability. A suitable paradigm for users writing their own features is that they write policies, using an elementary set of very basic built-in features. In this sense, the role that was played by features in the past will be partially played by policies in the future (therefore, the systems we know today as feature-based will evolve towards the policy-based structure) [28].

Policies may be used for all features that are set up to handle calls before the user participates, or in absence of the user. Defining policies should be easy for users so that it is worth their effort. Clearly this requires good, usable interfaces that reduce the special skills required.

Large-scale customizability by policies has implications for the types of feature interaction handling that will be used. Traditional systems provide a common centralized definition for features to all users, and feature interaction detection techniques function to make the definitions of all features compatible. In systems providing policy-based customization this is no longer possible. With IETF's Session Initiation Protocol (SIP) and Call Processing Language (CPL) for example, users can describe their own preferences in the making and receiving of calls. Preferences will be defined at run time and thousands of scripts will be operating simultaneously. It is unreasonable to expect that all of these policies can be made compatible. Additionally, the worth of making compatible the policies of users who will never call each other is dubious. Instead, feature interaction techniques must act at run time to allow the system to act understandably and beneficially to users in the face of incompatible preferences. Resolution techniques must be designed to this end.

As a result, important questions to be asked and answered are

- *How can users define policies and how can users define new features?* Users define policies with a policy language. The language will have to be presented with a number of interfaces to suit different kinds of users: e.g., a low-level interface for system administrators, voice interfaces for users on mobile phones or Web interfaces for general use. A policy definition language for call control has been presented in [27]; other languages exist for other domains. Existing software development technologies, such as formal methods, can be adapted to feature design and development. However, policy-based systems have not reached the stage to possess their own validation tools, as is the case for feature interactions.
- *What should users be allowed to customize?* This depends very much on the extent to which a user is trusted. It also depends on the underlying system and its fragility and robustness. A complete answer requires further work. With the advent of service-on-request and network-on-demand, it appears that feature interactions will be present in user actions. A role-base policy authorization, authentication, and access seem to be the key mechanisms to detect conflicts or interactions in user's requirements (at the request phase and during the demand fulfilment phase).
- *How can providers charge for policies?* This issue has not been considered at all. However, one could imagine libraries of policies (much like clip art libraries) being sold, as well as a charged for on a per use basis. The question is not trivial; there are well-defined cost models for new features that leverage existing features (cost split). If policies are embedded within policy-based applications, it is almost impossible to identify a cost model. With the advent of policy-frameworks, where policies are individual entities that can be imported, traded, and applied, charges for policies (as policy blades, see Fig. 2) are foreseeable.

3. Formalism convergences

3.1. Feature interactions formalisms for policies

Policies may express constraints on user actions. Policies can tune the operation of user features to current user context and needs. Policies can function as observers monitoring the behavior of users and call processing. They can intervene to provide for proper behavior and prevent undesirable actions and interactions. Large numbers of individual policies can be created that can cooperate to create useful systems.

3.1.1. Formalisms for professional designers

It is very difficult for designers to comprehend the behavior generated by large systems of independent policies. An aspect of this is their inexperience with this type of system [16]. A major issue is the lack of formalism and tools that would allow them to conceive, configure and operate such policies. To this end, Obligation–Permission–Interdiction (OPI) deontic task trees (see Fig. 3) were developed at Mitel [4].

OPI trees allow trained personnel to effectively reason about cooperating (or antagonistic) policies.

In these trees, each node is a task to either act in the world or observe it. An action may be to place an assertion in a shared memory space (so-called *tuple space*) that can be observed by observers in other trees. In call-processing applications, large numbers of trees may be configured to create useful features by monitoring user actions, triggering actions by the system and placing constraints on these actions.

Deontic logic is the logic of obligations. An action or an observation may be Obliged (O(Tree)), Forbidden or Interdicted (I(Tree)) or Permitted (P(Tree)). Each node is also supplied with an operator that instructs it in the temporal modality in which its offspring nodes are to be executed (*in sequence*, indicated by an arrow across the nodes, *by choice*, indicated by a curved line across the nodes, or *in parallel*, indicated by absence of operator symbols).

The deontic aspect of the trees allows the various nodes to be semantically composed to create the concept of success of the overall goal of the tree. With these trees, designers are able to relate cooperating groups of policies to the various goals and sub-goals required to operate a feature to the expectations of multiple users in multiple contexts. Designers are able to relate the success or failure of these goals to the overall intent of the features and the preferences of the various users. They have a formalism that allows them to relate feature and user purpose to the behavior of policies and to the relationships between them.

3.1.2. Formalisms for naïve users

Tools and formalisms will also be required for the naive users who will be given the ability to personalize call processing in IP telephony sys-

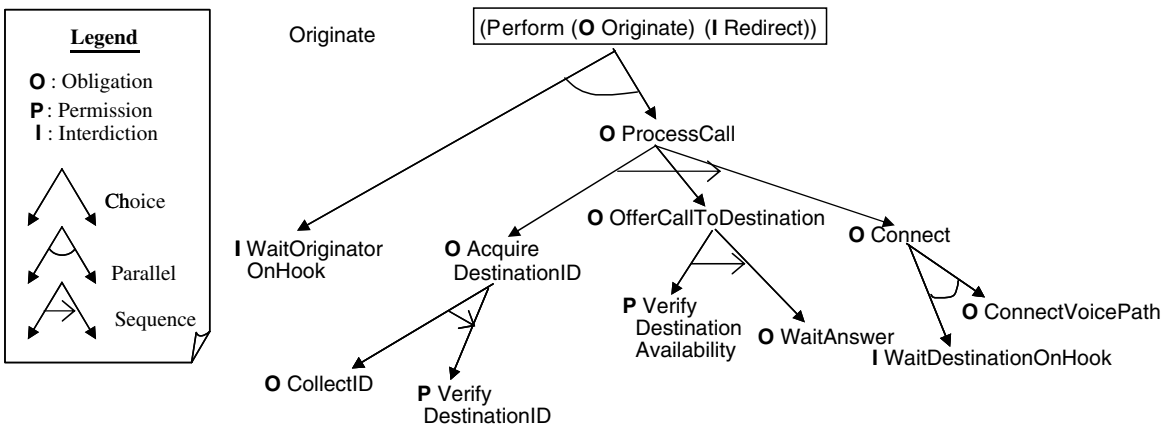

Fig. 3. Example of tree composition.

tems. Users will create their own sets of policies to constrain the behavior of their systems. As naïve programmers, they will use their own form of human reasoning to specify how the system should operate and will expect that the system understands them. For example, a user will comfortably tell his secretary that he does not want to be disturbed for the next two hours but if a certain important customer calls then she is to be put through immediately. This is a form of indeterminism that a logic-based analyzer would report. To the human secretary, following human rules, this is of course not so. He would know exactly what the boss wants and behave accordingly.

It can be expected that typical users will have many policies in order to cover the range of their contacts and the range of contexts in which they will interact with them. It is unreasonable to expect users to understand without aid the behavior that will emerge from large numbers of interacting policies. As shown in the secretary example above, users should be given the means to relate the expected operation of their policies to situations which they currently understand. Static detection tools to help users in this direction are being developed [3,25]. To accomplish this, policy systems for both professional and naïve feature designers must be informed by knowledge of the human world. Features will be expected to act properly in most contexts. Because of this and the fact users will be programming their own policies, the usefulness of offline detection techniques will be limited. Online resolution techniques must be able to function as users would expect in multiple human contexts. They must act with knowledge of human principles that can be found in sociology, psychology and other social sciences. Resolution methods for feature interaction must be informed of social facts. For example, they must be aware of the relevant organizational power of the participants in a call in order to select the most appropriate behavior given the preferences expressed by the calling and called parties in their policies. An application of this would be the rule that the preferences of a user's boss would be given precedence over those of a user's subordinates. In other words, to function properly, policies must be situated in the human context of the user. For example many features are triggered if the user is busy. In traditional systems, busy is a state of the user device. A device can only accept one call at a time and a user has only one device. However, in converged IP systems, devices will routinely be able to accept and process multiple calls and users will have multiple devices. A user engaged in a call may be interrupted by a more important call. The determination of the busy state moves then to the human level and depends on the human social situation. A user may be very busy to external salesmen but very available to his boss. Busy is a person's state and also depends on who is asking.

3.2. Policy-related languages

Many policy languages exist. The descriptive nature of policies lends itself to the use of a Policy Description Language (PDL). Such a language or logic must be able to express a large number of concepts to be usable for call-processing policies. On the other hand, limiting the complexity of the language is an important goal as it must be decidable under realistic time constraints and must be simple for an average user. Proposals to formulate simple policies and then compose them to obtain more complex cases have been made by many authors [17,19,28].

A simple policy can be defined by a policy rule (see Section 2.4). Each policy rule is composed of three parts: a trigger block, an action block and a condition block. Note that not every policy rule will have all three parts, in fact only the action block must be there. One or more rules are combined to form policies. The composition of the policy rules is achieved by defining sequences of them, or by providing choices. Each policy also can have a preference and will usually state to which target it applies.

Many system parameters are applicable in a typical call control system. A policy language suitable for this context must be able to express conditions based on the following parameters: caller (a user or agent on behalf of a user), callee (as caller), devices (mobile phone, PDA), call content (email, video, language), media (fixed, mobile, high speed), call type (emergency, long distance, intra-company, local), cost, quality, or

topics (wines, project x, weekend plans). The language must also consider the users' context, e.g., location, role, and interests [27].

We summarize a minimum set of requirements that a policy language must support, with the intent of expressing various conditions and actions:

- be able to specify policies at the instance and type level of any component or event;
- allow any kind of event naming schema and component naming schema;
- provide support for typed data;
- provide arithmetic and logical operators;
- allow negation;
- allow causal operators (e.g., implies);
- provide projection operators (e.g., mapping_to, etc.);
- provide quantitative (e.g., before, etc.) and qualitative (e.g., a_5_seconds_after_b, etc.) temporal operators;
- provide support for importing naming schemas;
- provide support for importing inventory;
- provide event processing operators (e.g., filter, count, etc.);
- provide location mobility (e.g., becomes, etc.);
- provide definition reuse (e.g., symptoms, dialects, etc.);
- provide existential operators, i.e., "for all", and "there_exists";
- provide "at_least", "at_most" operators.

It is important to mention that similar requirements hold for formalisms intended to specify features or deal with feature interactions.

During the panel discussion, it appeared that the agent paradigm leveraging the principles governing the legal world may be a unifying conceptual model. This approach could be valid for feature interactions and policy conflicts, as presented in the next section.

3.3. Policy systems, normative systems and legal systems

In the context of the foreseen convergence of features and policies, this section intends to show that policy systems have the potential to develop into normative systems, and that these normative systems will have a relationship with legal systems. This relationship has the potential to motivate exchanges of concepts and methods between policy systems and legal systems.

We are going towards an information society where humans (henceforth to be called persons in this section) and autonomous agents (henceforth to be called agents) will have interchangeable roles. Agents will enter into contracts with each other, or with persons, will acquire obligations and will be capable of committing torts. Hence, policy systems will develop into normative systems. However, persons will be ultimately responsible for their behavior, since in a legal sense an agent is no more capable of committing a tort than a gun is capable of committing a crime (although an agent can be 'punished' in various ways, for example by disconnecting it). Therefore, there is a need for seamless integration of the world of automatic agents, which is regulated by policies in the form of computer scripts, with the world of human agents, which is regulated by policies in the form of laws, statutes, contracts, etc. The worlds of law and agent policies will have to be mutually consistent, with priority given to law. The conflict resolution mechanisms will have to act consistently in the two worlds, because responsibility for an action in one world will have to match responsibility for the corresponding action in the other world.

The following example (due to Waël Hassan) illustrates the similarity of thinking in the two areas of policies and conventional telephony features. Let us consider a situation where a Client and a MerchantA enter into a contract, for example a purchase. The Client's policy is that a merchant should not sell his personal information, and MerchantA has a consistent policy. However, MerchantA subcontracts (delegates) to MerchantB, who delivers the goods, but does not have a policy on selling client information, and does sell it.

This situation has a logical structure that reminds one of the well-known Originating Call Screening–Call Forwarding feature interaction in telephony: suppose that calls from PhoneA to PhoneC are forbidden, and that at the same time PhoneB is in Call Forward to PhoneC. Then, un-

less particular mechanisms are put in place, a call from PhoneA to PhoneB will lead to a forbidden call between PhoneA and PhoneC. Again, we witness here a form of delegation from PhoneB to PhoneC, leading to an unwanted interaction. Delegation is pervasive in telecommunications, so other similar examples can be found.

The agent world, including the electronic commerce world with all its applications, will present us with numerous such situations, for which we must be prepared, to avoid equally numerous possibilities of incertitude, abuse, and litigiousness.

Now, feature interaction is the subject of human law, and there is much we can learn from feature interaction resolution mechanisms in law. In fact, we claim that, in this merger of the technological world and the legal world, lessons must be taken from both sides.

In the legal world, lawyers and judges have been doing feature interaction detection and resolution for thousands of years. There are methods for dealing with contradictions in normative systems, and methods for dealing with conflicts among agent policies. For example, if persons have a conflict on the use of an object, they will find guidance in law, where concepts exist to determine who has the right to use what (an ownership concept) and has mechanisms to detect and solve conflicts (a judicial system). However, if agents deadlock on a resource, little can be said beyond the fact that perhaps one of them can be suspended, usually at random or depending on technological considerations, simplistic solutions that are unacceptable in legal terms. New concepts and principles will have to be developed to cover many situations that can occur between agents, and possibly not between humans. These concepts will have to be consistent with the law (although in many cases they may be indifferent to the law), and eventually some of them may percolate back into the legal world.

Legal systems provide some important architectural lessons also, such as the necessity of trusted third parties (judges, notaries, and the government) and the possibility of distributing them.

We have seen that formalisms to express policies are required, but such formalisms need to be integrated into systems that detect and resolve conflicts that depend on context. Again, this is not unlike a court system, where judges are required to consider disputes in the context where they have occurred.

This discussion leads to our next point which is the need for uniform underlying conceptual models in this integrated world, as well as for similar interaction resolution mechanisms, wherever similarity is useful. In the technological world, automatic theorem proving and other logical methods have been developed to detect and resolve conflict situations based on precise rules. Some papers in the Feature Interaction Workshops (for example [14]) discuss the application of these methods, mainly in the area of telephony systems. Related methods are finding application in the world of law [18], although research at this point is concentrated on using Artificial Intelligence methods to aid or reproduce traditional legal reasoning. It is of course possible that such uniform logical models will not be developed, and that policy interactions and legal cases will continue to be solved with ad hoc or traditional methods. If this happens, an opportunity of gaining clarity and consistency in these two worlds will have been missed.

4. Industrial applications for service specification and management

In the following section, we will survey the potential benefit of applying feature interactions and policy-enabled mechanisms to concrete industrial applications.

4.1. Feature interaction between service provisioning systems

One area that stands to benefit from work on both feature interactions and policy-based management is service provisioning. Service provisioning systems are concerned with fulfilling requests for (communications) service, for example, requests for data and voice service by a residential subscriber, or requests to add a remote site to a business VPN. To fulfill a request, they translate the request for service into a sequence of

operations and configuration updates that are then applied to the affected devices in a communications network. Those devices are commonly also referred to as *network elements*.

To provision separate services over a converged network, in many cases multiple independent service provisioning systems are used, one for each category of services. Examples of combinations of services that are concurrently supported by converged networks are voice and data, voice and video, or cellular phone standards such as GSM and UMTS. Many service provisioning systems, however, expect to be provisioning "their" service on what amounts to a dedicated network, not a converged network, carrying out their provisioning operations without regard to the need of other services that are being provisioned on the same network. Accordingly, what amounts to negative feature interaction between the different provisioning processes can occur when the provisioning operations of one system negatively affect services provisioned by another system.

The effects of these negative interactions can be very undesirable:

- They can be function-impacting, resulting in one (or all) of the provisioned services becoming dysfunctional and needing to be reprovisioned. It is the case where the same network resource is reassigned for different purposes, e.g., a port that is used to carry GSM traffic unknowingly is reprovisioned for UMTS.
- They can be performance impacting. An example is the case where a connection admission control (CAC) policy is being provisioned without taking into account the network resource consumption by another service that is subsequently provisioned. This will result in too many connections being admitted that compete for resources, impacting Quality of Service for each of them.

The end result in all cases is poor service, violations of service level agreements, operational churn, and ultimately lost revenue and low profitability.

Therefore, the expectation is that service provisioning systems will provision services in a way that they do not impact each other. Specifically the following constraints need to be observed:

- Once a service instance is provisioned, it will keep functioning properly irrespective of other services provisioned over the same network.
- When a service is provisioned for a certain service level, it will continue performing within the service level bounds that it was provisioned for, irrespective of other services provisioned over the same network.

This corresponds to an isolation property, known by analogy with the ACID properties in database management systems (Atomicity, Consistency, Isolation, and Durability). In environments where isolation cannot be guaranteed, a weaker expectation will be that, at a minimum, at least one of the following conditions will be met:

- An application will be warned when attempting to carry out operations that may negatively impact a service previously provisioned by another system.
- The "owning" service management application will be warned when configuration changes that affect integrity of service instances that were provisioned earlier occur in the networks.

There are a number of practical ways to deal with these challenges and minimize infractions on these constraints and conditions. Many simply involve good engineering practices that minimize the risk of provisioning operations accidentally impacting each others, e.g., (i) the use of *configlets* as opposed to entire configuration files, or (ii) performing *rollback* of provisioning operations by undoing their effects, as opposed to simply reverting to earlier configuration files which may unintentionally undo valid operations that were in the meantime performed by other systems.

However, for the purpose of this paper, we want to examine how the disciplines of policy-based systems and feature interactions can be leveraged. One observation is that the mentioned constraints and conditions can be expressed and enforced as policies. The concept of policy, then,

becomes a technique to resolve particular feature interaction problems.

Beyond that, it is possible to map the process of service provisioning itself onto a set of policies. For example, formal descriptions of services and their mapping into the provisioning process can be interpreted as special-purpose policies [11]. Policies for the provisioning of multiple services can result in conflicts between policies. As already mentioned, there are obvious analogies between feature interactions and policy conflicts, which allow techniques for dealing with feature interaction to be applied to the resolution of policy conflicts. By mapping the resolved policy conflicts back onto the problem of resolving negative service provisioning interactions, we have applied a solution from the policy and feature interaction domain to the practical problem of service provisioning. As depicted in Fig. 4, instead of solving the problem of service provisioning interactions directly, the service provisioning problem is mapped into the policy domain. The transformed problem is subjected to policy resolution, possibly borrowing from feature interaction concepts. The resolution is then applied back into the service provisioning domain, solving the original problem.

Of course, as mentioned earlier, practical ways to deal with service provisioning interactions directly do exist. By applying a reverse mapping from the service provisioning domain onto the policy and feature interaction domains, it may be possible to also let the fields of policy and feature interaction benefit from concepts and solution approaches developed in an application field, in this case service provisioning. This is indicated by the reverse arrows.

4.2. Policy-based service management

Policy-based network management is a hot technology for the management of the emerging carrier grade IP networks. Due to the rapid advancement and development of the IP network, configuring the IP network is an overly complex, manually intensive, and constantly changing task. This has also made the task of managing these networks more challenging than ever. Consequently, more than 45% of the IP network operation cost comes from configuration and reconfiguration of the networks.

With policy-based network management, the industry is hoping to solve this problem. The idea is to unambiguously define management policies and let these policies drive the configuration and reconfiguration of the networks automatically. Nevertheless, most current commercial systems only allow limited QoS and access control policies to be specified by users as the policies themselves are difficult to capture and specify in formal languages unambiguously and without potential

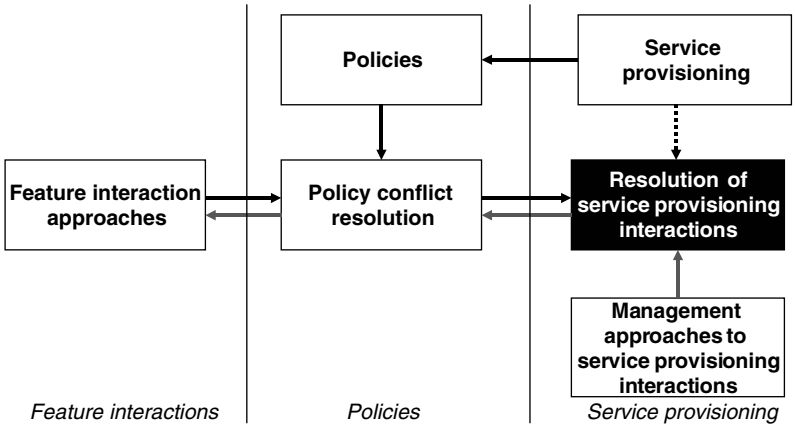

Fig. 4. Service provisioning using feature interaction and policy paradigms.

conflicts. Moreover, scalability is a problem when dealing with large-scale networks with many policy specifications.

At present, policies in network management are often expressed as IF–THEN clauses with a condition triggering some actions for network configuration or reconfiguration. However, this form cannot express many of today's network conditions. The issue is how much expressive power is needed to fully capture a network manager's intent. Should the language be the first order predicate logic or even temporal logic? Will there be issues of decidability with expressive policy languages? Moreover, if there are conflicts among policies, can we detect them before run time? Should policy rules be structured flat or hierarchically if there are many of them? All these issues are yet to be addressed.

In addition to the application of policies in network management and to the manifestation of policy conflicts as a form of feature interactions, we also see a new emerging area that may potentially tie the notion of policy even closer to the notion of feature interaction. We call this area *Policy-Based Service Management*. The premise is to move one layer higher above the network management and address the problem of better managing and leveraging a multitude of services enabled by PSTN, mobile, and IP networks. Currently, there exists no methodology that can support the rapid integration of cross-domain, cross-provider services to achieve this leverage. Nevertheless, the industry is moving in this direction and has started to bundle and aggregate services from various domains. For example, T-Mobile works with AOL to bundle its Short Message Services (SMS) with the popular AOL Instant Messaging (IM). Also, NTT DoCoMo works with the content providers to bring very rich content to its 2G network (i-mode). Though successful, the techniques used for service aggregation are rather ad hoc and cannot be applied to other services.

At the center of the issue is how to capture the end user's desired service behavior and then bundle the component services accordingly and automatically. Just as policies can be used to drive network management automatically, the concept of policy also holds the answer to an effective service management system in which a multitude of services enabled by various service providers can be bundled via a well-defined process. Policies here will be defined as the rules that regulate the behavior of the overall service when it is composed from its component services.

There is a strong business motivation in pursuing such a policy-based service management. Telecommunication service providers have searched for killer applications for many years since the emerging of Internet and mobile networks in the 90s. But in the past 10 yeas very few new voice services beyond familiar Caller ID, Call Forwarding, Call Waiting, Three Way Calling, and Voice Mail have been invented. The same observation can be applied to data services. Just ask, "What's new after Telnet, FTP, E-mail, Web, and IM?" An observation is that a different approach should be adopted by service providers for them to identify more revenue-generating applications, that is, to look for new ways of bundling and aggregating the already existing and successful services. Service providers, such as AOL, Voicestream, and NTT DoCoMo have taken this route and achieved a certain degree of success.

We can take advantages of policies to create new services via bundling and composition. The current practice shows that when a set of services is bundled, there may exist zero, one, or many potential composite service behaviors called *behavioral options* (see Fig. 5). In the case of zero, it means that the services cannot work together and thus cannot be bundled as an offer. In the case of

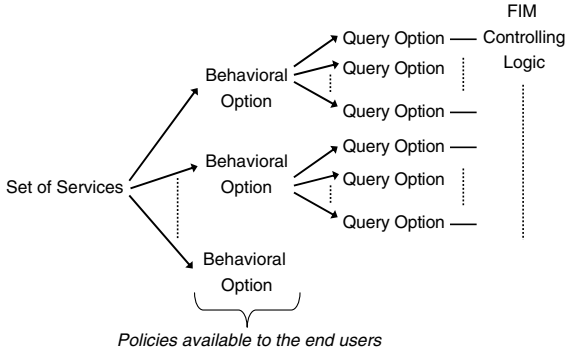

Fig. 5. Desired bundled service behavior as policy.

many, it means that the services can work together in many different ways and thus have the potential of generating many new composite services (each as a new behavioral option).

Furthermore, we have realized that many different invoking sequences (called *query options*) for implementing one composite service might exist. This invoking sequence actually is governed by the Feature Interaction Management (FIM) control logic for these component services. Fig. 5 depicts this relation among behavioral option, query option, and FIM control logic. Policy-based service management, as a new term, defines a policy as a desired bundled service behavior when a set of services is put together as a new service offer. These policies will dictate how component services have to work together to serve the needs of an end user. Thus which policy should be applied to the service management will be the end user's choice.

An example will help explain the above notion of interplay between policy and feature interaction. Let us suppose that a person subscribes to three services (*Originating Screening*, *Number Translation*, and *Originating Call Logging*) from three different service providers. There exist at least four behavioral options for these three services when they are bundled together. We can

1. screen on the dialed number and log every call attempt;
2. screen on the dialed number and log only the successful call;
3. screen on the translated number and log every call attempt, or
4. screen on the translated number and log only the successful call.

These behavioral options will become the management policies of these three distinct services and a user will need to choose among these four options the one that corresponds to the way he desires his services to work. Once the choice is made, this will be the user's service policy that will govern the behavior of his three bundled services.

Research problems exist here on how many behavior options, i.e., service policies, are possible given a set of services. Telcordia Technologies have developed a methodology for AIN 0.1 based on the notion of *behavioral relations* that, when applied, can algorithmically compute all the potential behavior options from the component services.

With application to the AIN 0.1 services, the following behavioral relations were defined:

- A is independent of B ($A \mid B$),
- B uses information generated by A ($A \rightarrow B$),
- A can disconnect the call and prevent B from affecting call processing ($A ! B$),
- A is incompatible with B ($A \# B$).

For other application domains, new behavioral relations will need to be invented and defined but the same methodology can still be applied.

Under this scheme of policy-based service management, the behavioral options, in fact, are the service policy specifications; the associated query options are, in fact, the FIM rules that would enforce a service policy.

5. Conclusions

The panelists brought examples from complementary areas and presented a wide range of views on the discussion topic. They presented their experiences on using policies for feature interaction problems, together with their position on the potential of using the policy paradigm for challenging feature interactions problems in telecommunications and software systems.

The opinions presented are controversial as is to be expected in panel presentations. They reflect the panelists' perceived reality and their short-, mid-, and long-term expectations. It is simple to overstate or underestimate the obstacles in combining the two paradigms of features and policies for achieving practical solutions for feature interaction problems and policy conflicts. Hopefully this panel report brings forward new ideas for research in the academic lane, as well as on the advanced industrial innovation lane. Maybe a merger or at least a much closer collaboration between the two research communities is on the horizon?

In conclusion, several trends can be emphasized as potential areas of intense research:

(i) The increasing importance of the concept of feature interaction in the policy world.
(ii) The development of (unified) formalisms to express features and policies.
(iii) System architectures that
  (a) in traditional feature-oriented systems: add support for policies, and
  (b) in policy-managed systems: add mechanisms for feature interaction handling.
(iv) The penetration of legal concepts in this world; cross-fertilization between the two areas with the birth of new concepts and new ways of thinking.
(v) The use of common logic concepts and application of theorem proving in both worlds of law and of agents.

It seems that these trends are inevitable, except for (v), which is only a matter of good thinking.

Trend (iii) requires further attention. It appears to be of great relevance to investigate the following:

(a) Services as policies
  - The use of the concept of policy for service definition and provisioning.
(b) Policies for the resolution of service provisioning conflicts
  - The use of the concept of policy for feature interaction resolution.

Policies and feature interactions are two very closely intertwined paradigms: when a new feature is created, appropriate policies must be invoked to resolve potential (undesired) interactions. This is usually done through additional policies or through particular mechanisms that are similar to policies, but are embedded into the feature itself. Thus, a policy is a feature for a policy-based system and a possible resolution mechanism for feature interactions.

Conversely, when a policy is added to an existing policy-system, feature interaction techniques must be used to detect and fix potential policy conflicts. Therefore, it is expected that results in feature interaction detection and resolution will be leveraged in the area of policy conflict interactions. Currently, policy conflict resolution is usually done through meta-policies. These might not be sufficiently general, since there could be conflicts between them. It is worth to add that meta-policies require a certain degree of agreement from the parties owning the policies under dispute, while this is not usually the case for features, since these are commonly uniquely owned. However for distributed feature interactions, where features may be provided by different suppliers, the problems become much more similar.

In the analyzed applications (service provisioning, service bundling through policy-based management systems, address translation), there are common aspects valid for both feature interactions and policies. First, there are static and dynamic considerations that determine the degree of problem resolution. Due to the distribution, it appears that run-time policy conflict resolution is quite a sophisticated operation.

Acknowledgements

Petre Dini thanks the panelists for accepting the invitation and for providing written statements on their positions. Tom Gray wants to thank his colleagues in the Mitel Networks Strategic Technology group and their university partners for their contribution to define and implement some ideas expressed in the panel. Luigi Logrippo's work was funded in part from grants of the Natural Sciences and Engineering Research Council of Canada, and he wants to thank his students and colleagues for many interesting ideas. Alex Clemm wants to thank Zbigniew Blaszczyk and Dave McNamee for their support. Stephan Reiff-Marganiec expresses thanks to Kenneth J. Turner and other colleagues at the University of Stirling for contributions to the policy work. Joe Lin thanks the Telcordia Technologies Labs's research and development teams for contributing to the ideas presented in the panel. The authors are also indebted to the referees for valuable comments and to Daniel Amyot for editorial help as well as for having contributed to these ideas through discussion over the years.

References

[1] I. Aggoun, P. Combes, Observers in the SCE and the SEE to detect and resolve service interactions, in: P. Dini, L. Logrippo, R. Boutaba (Eds.), Feature Interactions in Telecommunication Networks IV, IOS Press, Amsterdam, 1997, pp. 198–212.

[2] D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003.

[3] D. Amyot, T. Gray, R. Liscano, L. Logrippo, J. Sincennes, Interactive conflict detection and resolution for personalized features, submitted.

[4] M. Barbuceanu, T. Gray, S. Mankowski, How to make your agents fulfill their obligations, in: Third Conference on the Practical Applications of Intelligent Agents and Multi-Agents (PAAM'98), London, UK, 1998.

[5] A. Belokosztolszki, K. Moody, Meta-policies for distributed role-based access control systems, in: Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy 2002), Monterey, USA, 2002, pp. 106–116.

[6] M. Calder, M. Kolberg, E.H. Magill, S. Reiff-Marganiec, Feature interaction: a critical review and considered forecast, Comput. Networks 41 (1) (2003) 115–141.

[7] M. Calder, M. Kolberg, E. Magill, D. Marples, S. Reiff-Maganiec, Hybrid solutions to the feature interaction problem, in: D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003, pp. 295–312.

[8] E.J. Cameron, N. Griffeth, Y. Lin, M.E. Nilson, W.K. Schnure, H. Velthuijsen, A feature interaction benchmark for IN and beyond, IEEE Commun. 31 (1993) 64–69; also in: L.G. Bouma, H. Velthuijsen (Eds.), Workshop on Feature Interactions in Telecommunications Systems, IOS Press, Amsterdam, 1994, pp. 1–23 (revised and reprinted).

[9] C. Capellmann, P. Combes, J. Pettersson, B. Renard, J.L. Ruiz, Consistent interaction detection—a comprehensive approach integrated with service creation, in: P. Dini, L. Logrippo, R. Boutaba (Eds.), Feature Interactions in Telecommunication Networks IV, IOS Press, Amsterdam, 1997, pp. 183–197.

[10] J. Chomicki, J. Lobo, Monitors for history-based policies, in: Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy 2001), HP Labs Bristol, UK, 2001, pp. 58–72.

[11] A. Clemm, F. Shen, V. Lee, Generic provisioning of heterogeneous services—a close encounter with service profiles, Comput. Networks 43 (1) (2003) 43–57.

[12] N. Damianou et al. The Ponder specification language, in: Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy 2001), HP Labs Bristol, UK, 2001, pp. 18–38.

[13] P. Dini, G.v. Bochmann, Automatic reconfiguration for runtime feature interaction resolution in an object oriented environment, in: K.E. Cheng, T. Ohta (Eds.), Feature Interactions in Telecommunications III, IOS Press, Amsterdam, 1995, pp. 115–126.

[14] A. Felty, K. Namjoshi, Feature specification and automatic conflict detection, in: M. Calder, E.H. Magill (Eds.), Feature Interactions in Telecommunications and Software Systems VI, IOS Press, Amsterdam, 2000, pp. 179–192.

[15] L.Z. Granville, G.A. Faraco de Sa Coelho, M.J.B. Almeida, L.M.R. Tarouco, OoP: an automated policy replacement architecture from PBNM, in: Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy 2002), Monterey, USA, 2002, pp. 140–146.

[16] T. Gray, R. Liscano, B. Wellman, A. Quan-Haase, T. Radhakrishnan, Y. Choi, Context and intent in call processing, in: D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003, pp. 177–184.

[17] R. Hull, B. Kumar, D. Lieuwen, Towards federated policy management, in: Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy 2003), Lake Como, Italy, 2003, pp. 173–182.

[18] ICAIL 2003, Proceedings of the 9th International Conference on Artificial Intelligence and Law, June 24–28, Edinburgh, Scotland, UK, ACM, New York, 2003.

[19] Y. Kanada, Taxonomy and description of policy combination methods, in: Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy 2001), HP Labs Bristol, UK, 2001, pp. 171–184.

[20] S. Kawauchi, T. Ohta, Mechanisms for 3-way feature interactions occurrence and a detection system based on the mechanism, in: D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003, pp. 313–327.

[21] J. Keeny, V. Cahill, Chisel: a policy-driven, context-aware, dynamic adaptation framework, in: Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy 2003), Lake Como, Italy, 2003, pp. 3–14.

[22] K. Kimbler, Addressing the interaction problem at the enterprise level, in: P. Dini, L. Logrippo, R. Boutaba (Eds.), Feature Interactions in Telecommunication Networks IV, IOS Press, Amsterdam, 1997, pp. 13–22.

[23] E.C. Lupu, M. Sloman, Conflict analysis for management policies, in: R. Stadler, A. Lazar, R. Saracco (Eds.), Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Management, 1997, pp. 430–443.

[24] E.H. Magill, S. Tsang, B. Kelly, The feature interaction problem in networked multimedia services: past, present and future—4th deliverable (final report), Common Project of University of Strathclyde and British Telecom Research Laboratories, 1996.

[25] M. Nakamura, P. Leelaprute, K. Matsumoto, T. Kikuno, On detecting feature interactions in programmable service environment of Internet telephony, Comput. Networks 45 (5) (2004) 605–624.

[26] M. Popescu, Temporal constraints in policy-driven management systems, in: International Conference on Telecommunications (ICT2001), Bucharest, 2001.

[27] S. Reiff-Marganiec, K.J. Turner, Use of logic to describe enhanced communications services, in: D. Peled, M. Vardi (Eds.), Formal Techniques for Networked and Distributed Systems (FORTE 2002), Springer, Berlin, 2002, pp. 130–145.
[28] S. Reiff-Marganiec, K.J. Turner, A policy architecture for enhancing and controlling features, in: D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003, pp. 239–246.
[29] S. Reiff-Marganiec, K.J. Turner, Feature interactions in policies, Comput. Networks 45 (5) (2004) 569–584.
[30] A. Schaad, J. Moffet, Delegation of obligations, in: Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy 2002), Monterey, USA, 2002, pp. 25–35.
[31] P. Zave, Ideal address translation: principles, properties, and applications, in: D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003, pp. 257–274.

Petre Dini is senior technical leader and a principal architect at Cisco Systems for policy-based strategic systems and protocols for networks management, QoS/SLA, programmable networks and services, and consistent embedded manageability. His applied industrial research interests include instrumentation software agents, performance, scalability, autonomic computing, adaptive networks, and policy-related issues in adaptive networks. He is actively involved in standard groups (IEEE, ITU, and TMF) especially on topics pertaining to policies, component interactions, and system architectures. He has been an invited speaker to many international conferences, a tutorial lecturer, chaired several international conferences, and published dozens of technical papers. He received his M.Eng. from Polytechnic Institute of Timisoara, Romania, in Computer Engineering, and a Ph.D. in Computer Science from University of Montreal, Canada. He is currently an Adjunct Professor at Concordia University, Montreal, Canada, a Senior IEEE member, and an ACM member.

Alexander Clemm is a principal architect and a senior manager of engineering at Cisco Systems, where he has assumed a variety of architectural responsibilities that include embedded device management, network manageability with regards to service assurance, convergent management application architectures, and turnkey solutions for the management of large scale packet telephony networks. He is on the technical program committee of several network management-related workshops and conferences, such as IM, NOMS, DSOM, and MMNS. He holds Master's and Ph.D. degrees in Computer Science from Stanford University and the University of Munich, respectively.

Tom Gray has had a 30-year career in the telecommunications industry. He has worked for Bell Northern Research, Mitel Corporation, and Mitel Networks. He has over 40 patents issued and pending, and over 30 papers published in international conferences and archival journals. At Mitel, he was instrumental in developing a consortium of university projects that explored the types of services that will be useful and profitable in new converged multimedia networks and how they would be developed, operated, and maintained. Over 20 university research groups participated in this project in Canada and the United Kingdom. This effort resulted in useful new technology, many papers and patents, and the education of numerous students. Prior to this, he was the prime mover in the group that created the vision and technology behind the development of the Mitel Light family of PBXs. The group realized that the assumptions that had been seen as fundamental to the design of telephone switches were no longer valid and based a new architecture on the capabilities of new technology. The success of this product changed the industry-wide model of PBX architecture. The basis of this architecture has guided the development of PBXs and other telecommunication switches up to the present day.

Fuchun Joseph Lin is a Chief Scientist in Applied Research of Telcordia Technologies (Formerly Bellcore). He has 15 years of experience in both Bell Labs and Telcordia Technologies. He received his Ph.D. in Computer Science from the Ohio State University in Columbus, Ohio and his BS and MS in Computer Science from National Chiao-Tung University in Hsinchu, Taiwan. He joined Applied Research of Telcordia Technologies in 1992 after four years of work experience at Bell Labs in the 5ESS Switching Division. At Telcordia, his initial focus was on Intelligent Networks, especially on the research of service creation and feature interactions. He then moves into the areas of integrated services and network convergence for next generation networks. He currently serves as a project manager and technical director at Telcordia Applied Research, managing Telcordia External Research Programs on VoIP and Next Generation Networks. He is a member of the IEEE Communications and Computer Societies, and also a member of the ACM. He serves in the program committees of several international conferences and has won two patents and published two dozens of technical papers in professional conferences and journals.

Luigi Logrippo received a degree in law from the University of Rome (Italy) in 1961, and in the same year he started a career in computing. He worked for several computer companies and in 1969 he obtained an MSc in Computer Science from the University of Manitoba, followed by a PhD in Computer Science from the University of Waterloo in 1974. He was with the University of Ottawa for 29 years, where he was Chair of the Computer Science Department for 7 years. In 2002 he moved to the Université du Québec en Outaouais, Département d'informatique et ingénierie, while

remaining associated with the University of Ottawa as an Adjunct Professor. His interest area is formal and logic-based methods and their applications in the design of communications systems. For a number of years he worked on the development of tools and methods for the language LOTOS. His current research deals with the formal analysis of advanced communications services made possible by internet telephony, of the policies that govern them, and of their interactions, in application areas such as presence features and e-commerce contracts.

Stephan Reiff-Marganiec was working in the computer industry in Germany and Luxembourg for several years. He obtained a BSc (hons) degree in Computing Science from the University of Wales, Swansea, in 1998. From 1998 to 2001, he was working as a Research Assistant on the EPSRC HFIG project at Glasgow University, while at the same time reading for a PhD in Computing Science. The work performed at Glasgow investigated hybrid approaches to the feature interaction problem, and the thesis presents one such approach. From 2001 to 2003 he worked as a Research Fellow on the ACCENT project at the University of Stirling, investigating policies, emerging features and associated conflict resolution techniques. Since 2003 he is a lecturer at the University of Leicester, pursuing research on telecommunication and web services, particularly considering component based and reconfigurable systems in the context of rapid market changes and complex legacy products.