An Agent-based Model for the Provision of Advanced Telecommunications Services

Mike Rizzo

Dept of Computer Science & A.I.
University of Malta
Msida, Malta
Tel: ++356 333903
Fax: ++356 320539
Email: mriz@unimt.mt
(from October 1995)

Computing Laboratory
University of Kent at Canterbury
Canterbury, Kent CT2 7NF, U.K.
Tel: ++44 1227 764000
Fax: ++44 1227 762811
Email: M.Rizzo@ukc.ac.uk
(to September 1995)

Ian A. Utting
Computing Laboratory
University of Kent at Canterbury
Canterbury, Kent CT2 7NF, U.K.
Tel: ++44 1227 764000
Fax: ++44 1227 762811
Email: I.A.Utting@ukc.ac.uk

Keywords: intelligent networks, capability sets, feature interaction, agent

Abstract

Modern stored program control (SPC) digital exchanges represent an opportunity to enhance traditional telecommunications services by offering subscribers finer control over the way their calls are managed. Thus far, the predominant approach (Intelligent Networks) has involved making highly specialised, well-defined functions available as services. These tend to exhibit a low degree of configurability and often interact with each other in undesirable ways (a phenomenon commonly known as *feature interaction*). This approach does not exploit the potential offered by SPC exchanges to the full, and cannot adequately deal with specific customer requirements.

We propose an alternative approach whereby calls are managed by a number of co-operating agents acting on behalf of subscribers. In this model, subscribers have finer control over the functionality offered by the switch via their agents. The latter communicate amongst each other using a negotiation protocol which enables many kinds of feature interaction problems to be avoided. Our model considerably alters the traditional enterprise viewpoint and requires radically different approaches to marketing, deployment, evolution and tariffing of services.

1 Introduction

Modern stored program control (SPC) digital exchanges represent an opportunity to enhance existing telecommunications services by offering subscribers finer control over the way their calls are managed. In the Intelligent Network (IN) [Inta] approach, a number of services have been identified and grouped into capability sets for standardisation purposes. Each of these services performs a highly specialised function, and subscribers often have little control over the behaviour of a service. Subscriber control is generally limited to simple interactions such as turning features on and off, specifying alternative telephone numbers, keying in personal identification number codes, and toggling between calls. This rigid approach does not exploit the potential of SPC exchanges to the full and is often not flexible enough to satisfy subscribers' requirements. Additionally, no mechanisms are defined to control feature interaction.

In this paper we propose an alternative approach that can better exploit SPC exchanges in an open distributed environment. The underlying principle of this approach is to offer a finer granularity interface to the subscriber, allowing for more flexible control. We propose an agent-based model and compare and contrast this with the IN approach.

2 Limitations of the IN Capability Set approach

The principal objectives of the model we will be describing aim to overcome two drawbacks inherent in the IN capability set approach, namely (i) poor extensibility and flexibility characteristics, and (ii) the absence of a general-purpose mechanism to resolve certain kinds of feature interaction dynamically. In this section we demonstrate these problems using examples, and show that (ii) is in fact a consequence of (i).

2.1 Extensibility and Flexibility Limitations in IN

One of the simpler service features documented in IN CS-1 [Intb] is known as *Call Forwarding on Busy/Don't Answer* (CFC). This allows called parties to have their incoming calls addressed to another number if their line is busy, or if the call is not answered within a specified number of rings.

Consider a scenario in which a call originating from A to B is forwarded to C via CFC. In the case that that C also turns out to be busy, or doesn't answer, one of two things can happen: (i) the calling party A hears an engaged tone (busy) or a ringing tone (not answered), or (ii) a service feature, such as Call Waiting (CW) or even CFC again, is invoked on behalf of C, giving rise to a feature interaction.

Neither of these alternatives may be acceptable to the original called party B. In case (i), the call is effectively lost altogether. In case (ii), B relinquishes control over the call completely and, in a worst-case scenario, A might end up being forwarded to a number which B does not approve of.

A called party might want to specify a second forwarding number to be used in the event that the first forwarding attempt fails. This functionality is not available in either of CS-1 or CS-2. In the vein of the IN approach, it could be introduced as a new service feature[1], say *Double Call Forwarding on Busy/Don't Answer* (DCFC). But then again, another called party might want to specify more than two forwarding numbers in which case DCFC could be extended to *Multiple Call Forwarding on Busy/Don't Answer* (MCFC).

This new feature may still be insufficient. A called party might require that, in the event of its line having been found to be busy and all forwarding attempts having failed,

[1] A good example of this phenomenon is Selective Call Forwarding on Busy/Don't Answer (SCF). This essentially combines a generalisation of CFC with a variant of Originating Call Screening (OCS).

CW should be invoked. Another user might wish to specify similar behaviour, but with Call Queueing (QUE) in place of CW. Yet another user might require Automatic Call Back (ACB) instead.

Clearly, to carry on with the IN philosophy of adding new features whenever new variations, generalisations or combinations of existing features are identified would rapidly lead to an explosion of new capabilities. Additionally, it is difficult to anticipate what users' requirements will be in advance, and it is therefore not possible to deploy a finite set of specialised features that will meet each and every subscriber's requirements to the full.

2.2 Feature Interaction in IN

Another problem inherent in the IN approach is that the multitude of specialised features give rise to several kinds of feature interaction[2].

Sometimes, resolution of a feature interaction involves making a choice between two alternatives. A default choice might be hard-wired into the intelligent network, but allowing subscribers to make the choice is often desirable. Unfortunately, this would necessitate the introduction of yet another new feature. For example, if Originating Call Screening (OCS) is introduced in a network already supporting Call Forwarding (CF), another feature will have to be introduced in order to give a subscriber control over whether to apply OCS to forwarding numbers or not. This is rather clumsy but there appears to be no other way around it if the subscriber is to be given control over the interaction.

Another kind of feature interaction problem is due to the lack of support for a negotiation process to prevent two clashing features from being activated at either end of a call. One example of this is exhibited by Call Waiting (CW) at the called party's end and Completion of Calls to Busy Subscriber (CCBS) at the calling party's end. Clearly it only makes sense to activate one of these and some form of negotiation between the two parties is needed in order to decide which one.

The lack of such a negotiation process can also give rise to race conditions as exhibited by CCBS at the calling party's end and ACB at the called party, where both sides can end up repeatedly trying to call each other simultaneously, resulting in livelock.

The feature interaction problem actually encompasses a variety of different kinds of problems. The kind we have described so far are problems of a run-time nature that require some choice to be made dynamically. These are arguably the hardest interactions to deal with from a computational viewpoint, and they are precisely the class of problems that our model addresses.

Problems of a static nature can frequently be solved by applying sound software engineering principles and are not addressed specifically by our model. A case in point is the problem that arises when two different but simultaneously active features use the same signals to mean different things. For example, in some implementations of multi-way calling (MWC) a flashhook signal issued by a busy party is interpreted as a request to add a third party to the call. The same flashhook signal is also used in some implementations of Call Waiting (CW) to put the current connection on hold and switch to a new caller. It would therefore not be possible to subscribe to both these services. From a computer scientist's point of view, this would not be considered a feature interaction problem but a user interface problem. It arises from the failure of IN to distinguish user interfacing concerns from feature

[2] See [C+93] for a comprehensive overview and examples of feature interaction.

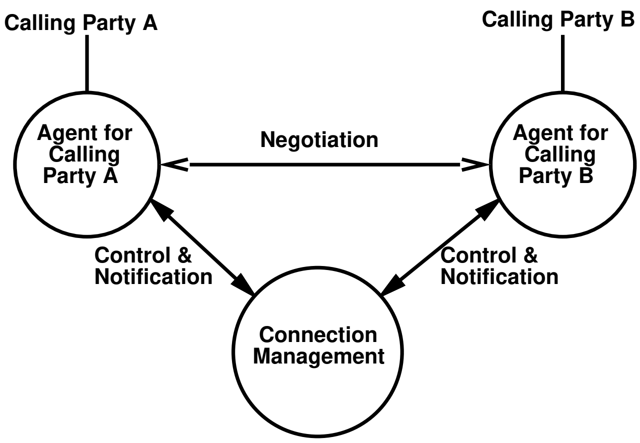

Figure 1: Agent-based model for Call Management

functionality. A clear separation of the user interface would enable subscribers to customise their interface as they deem suitable, and would facilitate the gradual introduction of more sophisticated terminal devices with better user interface characteristics[3].

3 An Agent-based Model for Call Management

We propose an alternative approach to exploiting SPC digital exchanges whereby subscribers employ agents [4] to handle calls on their behalf (see figure 1). This approach differs from IN in that (i) the network exports a finer-grain interface for use by subscribers' agents, and (ii) subscribers have more flexible control over the behaviour of agents. Consequently, the same functionality that is packaged into services and features in IN is still possible, but is no longer explicitly compartmentalised into well-defined units.

Agents exercise control over the network via a *connection management interface*. This interface minimally allows the setting up of connections between subscribers' terminal equipment. It may also offer operations to transfer and dissolve connections, and to generate audio signals on an already connected line (such as a Call Waiting indicator tone).

An agent's behaviour is partially determined by a subscriber's *call management policy*. Typically, this is expressed in terms of a *policy specification language* and captures a subscriber's requirements with respect to various management aspects of both incoming and outbound calls. The range of capabilities available to a subscriber is limited by the policy specification language which is in turn limited by the connection management interface.

In the agent model, the calling party places a call request via its agent. This request is viewed as a temporary extension to the calling agent's policy; it determines how the call in question is to be managed. Before attempting to set up the call, the calling agent engages in a negotiation process with the primary target agent. If it is determined that the call can go ahead then a connection is established. Otherwise the negotiation process might conclude with an alternative course of action to follow. This might involve agents

[3] Such devices are already emerging, as demonstrated by British Telecom's latest payphone and PC-based videophone card.

[4] One of the earliest references to the notion of an agent can be found in [Kay84].

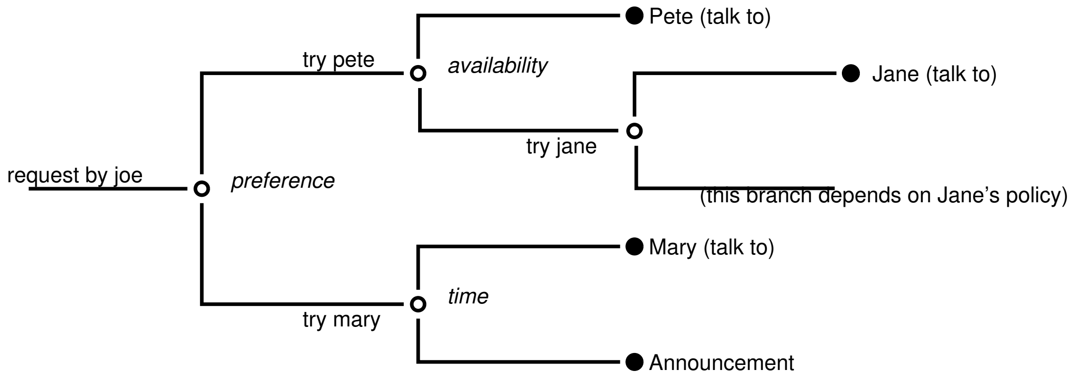

Figure 2: Pictorial representation of possible outcomes

negotiating to set up a different call immediately (Call Forwarding being one such example) or possibly re-attempting to make the call at some later time (as in Completion of Calls to Busy Subscriber). Such a course of action is determined by the policies at either end.

Once a connection has been established, an agent can still exercise control over it. For example, a subscriber may interact with an an agent in the course of a call in order to effect a transfer.

The negotiation process works as follows: the calling agent makes a proposal to the target agent. This either accepts the proposal, or refuses, in which case it may suggest an alternative action (known as a fallback). If such a fallback is returned, the calling agent must then decide whether to go ahead with it, or whether to try some other alternative action of its own. In the latter case, the calling agent is allowed to store the fallback offer so that it can be taken up later if all else fails.

Evidently, this model requires the establishment of a common negotiation protocol. However, there is no requirement for a standard policy specification language. Any pair of agents can always participate in a negotiation, even if they are guided by policies written using different languages. This property is extremely important because it allows different agent service providers to follow their own evolution path independently.

Additionally, in order to make the system more accessible to subscribers, requests and policies may well be specified using a graphical language as opposed to a text-based one. If desired, a mixture of graphical and textual policy specification tools could be provided, each suited to some particular kind of policy and varying in the amount of granularity and flexibility provided.

3.1 Illustration

Consider a subscriber Joe placing a call to another person Pete at 17:30. Pete's line is busy and his policy dictates that a fallback to connect the call to Jane should be offered. However, Joe has specified that failing to call Pete, he would rather try talking to Mary before trying anything else. Mary doesn't accept calls after 17:00 and her policy returns a fallback specifying that all incoming calls after this time should be connected to an announcement service. Joe's policy specifies that failing communication with Mary, he would like to follow the fallback offered by Pete. Jane's agent is able to accept the call and so Joe is finally connected to Jane.

Figure 2 is a pictorial representation of all the possible outcomes arising from the interaction of these policies. The root node represents Joe's request, intermediate nodes represent decision-making points, and leaf nodes represent actual connections. Note that the tree is incomplete as we have not described Jane's policy.

Joe's call request might look something like this:

```
res_pete = try(talk_pete);
if (res_pete.result == Failure) {
    res_mary = try(talk_mary);
    if (res_mary.result == Failure) {
        res_fallback = try(res_pete.fallback);
    }
}
```

Pseudo-code for relevant parts of the policies of Pete and Mary are shown below:

```
/* Pete's policy */
switch (line_status) {
   ...
case BUSY:
    offer(talk_jane);
    break;
   ...
}

/* Mary's policy */
if (curtime > 17:00 || curtime < 08:00) {
    offer(announcement_mary);
}
else {
    accept();
}
```

This is a very simple example, but it illustrates the concepts of requests, policies, and negotiation. It is difficult to see how the kind of functionality exhibited in this scenario could be achieved in the existing IN framework.

Another point worth noting is that no notion of feature interaction exists. Essentially, the calling agent is always in control, and invokes fallbacks only if it so desires.

3.2 Universal Personal Telecommunications

Thus far, we have avoided defining exactly what we mean by a subscriber. More specifically, we have not distinguished between users and the end-points via which they communicate, but have assumed a one-to-one correspondence between the two. In traditional telephony, end-points are the only directly addressable entities and the term subscriber refers to the person or organisation responsible for paying the charges associated with a particular end-point.

More recently, however, the notion of Universal Personal Telecommunications (UPT) has emerged, whereby mobile users are also addressable entities that can be reached via any of several end-points, depending on that user's location. Here, UPT subscribers are assigned a Personal Telecommunication Number (PTN) and are required to provide an appropriate telephone number for the location in which they are currently situated. Calls directed to a PTN are then automatically routed to the user's current location.

More sophisticated location mechanisms are possible. For example, an Active Badge system [HH94] combined with a location database could be used to determine the nearest telephone to a user. Other sources of location information include computer system login records and on-line electronic diaries. All these sources may be used collectively, so that where one mechanism fails, another may succeed [RLU94]. The specific location mechanisms available can vary considerably amongst different users, depending on their working habits as well as the facilities available within their local organisations. A good model for UPT should not constrain the choice of location mechanisms but should provide appropriate hooks for any available mechanisms to be used under the guidance of some user-defined policy.

It will often be the case that the end-point at which a user may be contacted is associated with an altogether different subscriber who may wish to enforce some policy with respect to the use of that end-point. For example, in order to maximise availability to local users, it might be desirable to restrict calls to 'foreign' users to a maximum length of three minutes. A model for UPT should allow such end-point policy constraints to be respected.

We can refine our model by providing distinct agents for both users and end-points. In this model, calls can still orginate from and be directed to end-point agents, but they can also originate from and be directed to user agents. As before, agents are guided by policies and agree on a course of action by means of a negotiation process. Because agents exist at both the user and end-point level, it is possible to impose policy constraints for both kinds of entities independently.

Figure 3 illustrates a typical user-to-user setup in which two user agents are each engaged in two negotiation processes: one with the other, and one with an end-point agent. One possible sequence of events leading up to this situation is as follows:

1. User A instructs their agent to set up a call with user B.

2. The agent for user A (hereafter referred to as agent A) initiates a negotiation process with agent B, passing a service request.

3. Agent B evaluates the request, and decides that it will be ready to provide service if and when agent A has ascertained that user A is actually reachable; i.e. that agent A has obtained an end-point for user A. At this stage, B is expressing a willingness to provide service under certain conditions, but this does not necessarily guarantee that it will be able to actually deliver service if and when the conditions are satisfied.

4. Agent A responds to agent B's reply by attempting to obtain an end-point for user A. This involves locating user A via some appropriate location mechanism, resulting in the identification of a suitable end-point agent X[5].

[5] Of course, if user A has initiated this process through end-point X, this step will be trivial!

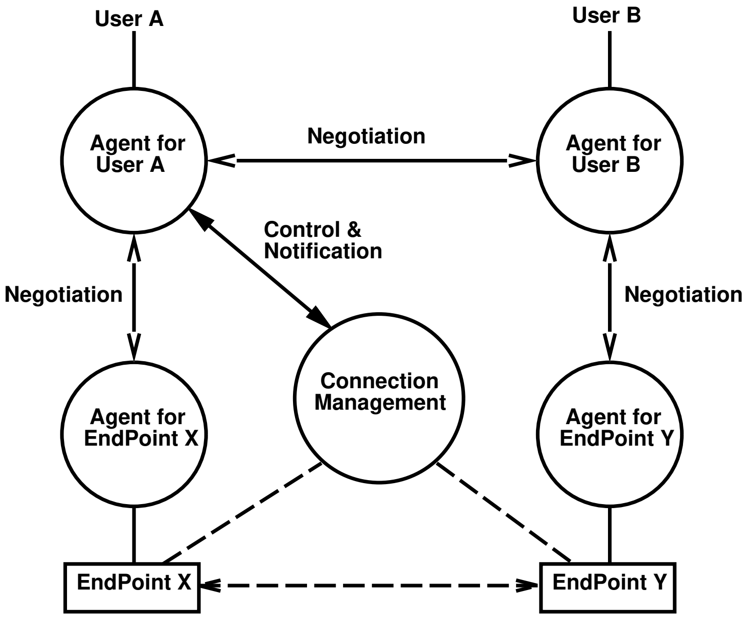

Figure 3: Distinguishing between user agents and end-point agents.

5. If the location process succeeds, agent A initiates a negotiation process with the returned end-point agent X. From agent A's point of view, the ultimate objective of the negotiation process is to obtain an end-point reference. From the point of view of the end-point agent X, the objective of the negotiation process is to ensure that use of its associated end-point is in line with the policy of the party responsible for that end-point.

6. Agent X returns an end-point reference to agent A.

7. Agent A informs agent B that it now has an end-point for user A.

8. Agent B invokes a location process which returns a reference to end-point agent Y.

9. Agent B initiates a negotiation with agent Y.

10. Agent Y returns an end-point reference to agent B.

11. Agent B passes this reference back to agent A.

12. Agent A now has references for two end-points, one for each user, and proceeds to establish a connection between them via the ConnectionManagement interface.

The negotiation processes remain active during the course of the call so that any attempts to manipulate the call in some way or other can be subjected to the approval of the relevant agents. For example, user B might instruct agent B to forward the call to agent C. This request is passed to agent A, which might seek further information about agent C and will also have to get approval from agent X before going ahead with the transfer.

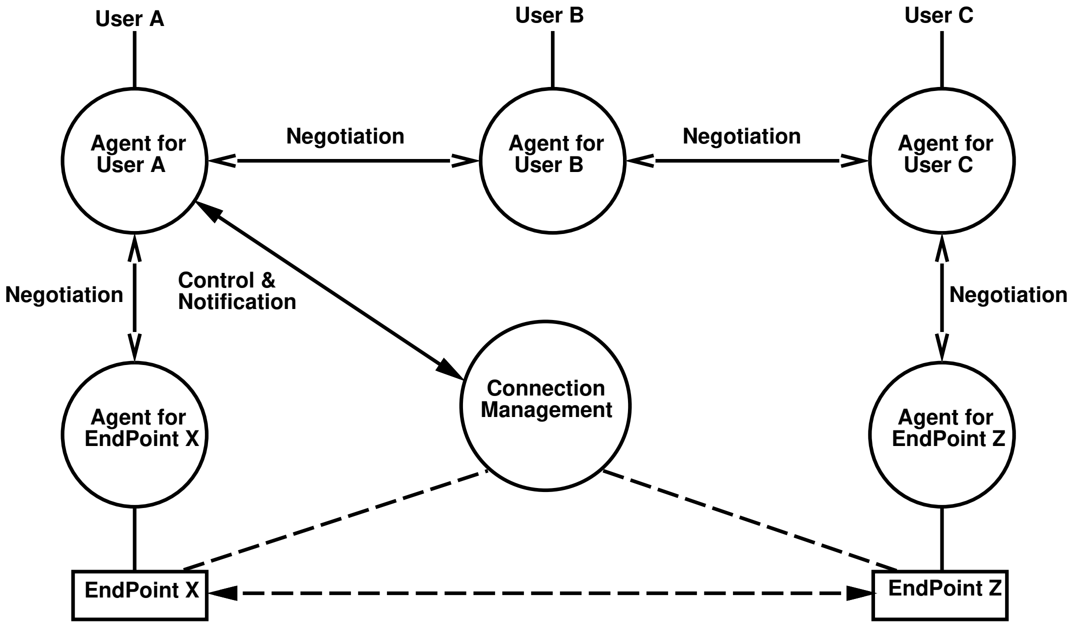

Figure 4: Transferring a call.

This model allows for two kinds of transfer: agent B can withdraw from the call completely so that agent A ends up negotiating directly with agent C, or agent B can retain some degree of control by acting as an intermediary between agent A and agent C (see figure 4). In the latter case, any action initiated by either agent A or agent C must be approved by agent B. This can be very useful: it means, for example, that once agent B has forwarded the call to agent C, it can prevent agent C from forwarding the call to some other agent which B does not approve of.

3.3 Interactive services

Agents can also be used to represent interactive, automated services such as tele-banking and voice messaging services. Such services would typically provide end-points that play back pre-recorded speech messages and process incoming speech and keypad signals.

An agent representing such a service would be responsible for ensuring that any conditions of use associated with the service are observed. For example, it might be required to ensure that any calling agent provides identification.

A service agent might also be responsible for generating a forwarding request to the calling agent, providing a new end-point to connect to. For example, a user calling an information service might first be connected to an end-point which reads out a menu offering a choice of different actions. Upon selection, the service would pass back a second end-point to its agent, which then passes this back to the caller agent so that this may effect the transfer via the connection management interface.

3.4 Agent Negotiation Protocol

The choice of negotiation protocol is clearly a crucial one as this is the only element in the model that is required to be common across all administrative boundaries. It is unlikely that any negotiation protocol would be able to cope with all requirements and situations in its first incarnation. Consequently, it is important that the protocol be designed with backward-compatible extensibility in mind. Backward compatibility is important because as the negotiation protocol evolves, different administrative domains will not upgrade to the latest version at the same time.

The negotiation protocol is primarily concerned with information exchange. One agent might ask another for a piece of information, such as a reference to an end-point or identification of a user. In order to supply an item of information, an agent may consult other agents, or might require that the requester itself provide some information before it complies. It is important to distinguish between the information exchange mechanisms and the nature of the information items themselves. It is our hypothesis that if designed properly, the information exchange mechanisms will not need to evolve frequently, if at all.

On the other hand, it is difficult to predict exactly what information item types will be required in a negotiation protocol as these are more likely to evolve over time. Whereas certain items, such as agent references, end-point references, connection references, user identifiers, and end-point identifiers are of immediate fundamental importance, the need for other less obvious items, such as maximum duration of a call, are likely to be recognised long after deployment of the first negotiation protocol.

Fortunately it is easier and less disruptive to get agents to recognise new item types than it is to change information exchange mechanisms. A well designed agent will be able to support many new item types through re-configuration as opposed to modification. Where an item of information cannot be obtained from an agent, it might be possible to obtain the information directly via interaction with a user. For example, an agent for an interactive service might ask the caller's agent for a preferred language. If the caller's agent cannot supply this information, the interactive service will ask for it. The service's agent might then advise the caller's agent of the language selected. It would be up to the caller's agent to decide whether to take heed of that advice or not, possibly after consulting with the user it represents.

3.5 Policy specifications

Earlier we stated that an agent acts under the influence of a user-defined policy. In practice, this policy is likely to be a composite one, consisting minimally of components provided by a subscriber and at least one higher-level organisation to which that subscriber belongs. For example, a company might choose to bar call forwarding to international numbers, and policies for individuals working within the company must respect this higher-level constraint. On a public network, a national telcommunications authority might impose the constraint that all end-point agents should be able to connect to emergency numbers so that it would not be possible for individual end-point policies to block calls to such numbers.

We also mentioned that agent policies could be specified using a variety of specification languages, and that there was no requirement for any pair of negotiating agents to use the same language. Indeed, it is not even a requirement for such a language to exist: policy might well be hard-wired into an agent, or an agent might be endowed with 'intelligence' so that it takes decisions based on a user's past behaviour instead of being guided by a precisely-specified policy [Mae94, M+94]. Other possibile approaches include visual programming environments and programming by demonstration [CS+94].

Wherever an agent does offer a policy specification language it must ensure that attempts to overrule a higher-level policy constraint are detected and trapped. In some cases it might be able to do this statically by applying validation tools to a submitted policy. However, dynamic detection is also required to cope, for example, with situations where a forwarding number is made known to an agent in the course of a negotiation.

4 Enterprise viewpoint considerations

Our model is not compatible with the traditional enterprise viewpoint for provision of advanced telecommunications services. In this section we discuss some important enterprise considerations, exploring some possible approaches that are better suited to the agent model.

4.1 Writing Policy Specifications

As always, there is a price to pay for additional flexibility: it is much harder to express what is required of an agent than it is to 'program' IN services. Intrinsically, this is due to the possibility of resolving in policy specifications those issues of feature interaction that are left open in IN approaches. The level of difficulty is of course related to the choice of policy specification language, which involves a trade-off between ease-of-use and expressive power. Agents may support a range of specification approaches[6], giving subscribers the option of expressing their requirements in whichever way they find to be satisfactory.

Within an organisation, one could perhaps foresee a relatively easy to use language which worked within the constraints of the organisation's policy. Should members of the organisation require some additional functionality, they could ask the relevant authority within the organisation for this to be provided to them. This authority could then arrange for the language to be extended accordingly.

Another possibility is the notion of a *policy consultant* responsible for writing policies on subscribers' behalf. Subscribers would describe their requirements informally to such a consultant, who would then translate these into a formal specification suitable for submission to an agent.

4.2 Agent Provision

In the same way that the notion of a service provider denotes an entity responsible for supplying services in IN, our model uses the notion of an *agent provider* to supply agents, be they for end-points, users, or interactive services. The same provider may supply a variety of agents offering varying degrees of flexibility and supporting a range of policy specification approaches. In addition to policy constraints imposed by a national telecommunications authority, an agent provider might impose further constraints on some or all of its agents above any other constraints specified by an individual subscriber. Such provider-level constraints should be taken into account by subscribers in the process of choosing an agent provider.

[6] Including possibly an emulation of IN capability sets.

Some agent providers might allow connectivity with other systems in order to allow policies to make use of external facilities such as location services. Alternatively, an organisation may wish to act as an agent provider for its own members, providing agents that are customised specifically to meet that organisation's requirements.

4.3 Marketing and Tariffing

One of the fundamental differences between our model and traditional approaches is that the increased functionality is no longer packaged into convenient service units as is the case with IN capability sets. This makes it more difficult to market this functionality as well as to charge for it.

With capability sets, the tendency has been to market and tariff services individually. It is quite common for a service provider to advertise introduction of a new service using brochures that explain what the service does, how it will be used and how it will be charged for.

In our model, marketing and tariffing can no longer work on this unit-by-unit basis. There is no notion of introduction of a new service, but only the notion of refinement of a specification language, or the availability of a new one.

Tariffing of agent provision could perhaps be done using a flat, annual rate on the basis of the expressive power of the policy specification language used. To charge on a per call basis might prove impractical because it will often not be possible to predict how much a call will cost in advance (depending on what fallbacks are followed, if any).

Charging for services provided over the network, such as credit card hot-lines (which traditionally has been done using the notion of a *premium rate* service), could be incorporated into the negotiation process. Consider, for example, the possible undesirable behaviour resulting from the use of ACB with premium rate numbers. By allowing charging constraints to be stated explicitly in call management policies, such behaviour can be prevented.

Establishment of charges for usage of the *network* (as opposed to services) could also be done as part of the negotiation process. For example, some implementations of call forwarding always charge the forwarder of the call for network usage. While this scheme might be acceptable in certain circumstances, it is not difficult to think of cases where it would be entirely inappropriate. By incorporating network charging into the negotiation process, agents would be able to agree on who should be responsible for network charges before processing the forwarding request.

Marketing is perhaps harder to tackle, especially at the level of public networks. Instead of focussing on services and features, it will have to focus on examples of the kinds of things that can be done with a language. In the case of languages favouring ease-of-use over expressive power, this should not prove too difficult. Such languages should prove to be more popular with the average public network subscriber in any case. The increased flexibility benefits of our model are perhaps more advantageous to large organisations within which one would expect to find one or more technically competent people who would be capable of digesting marketing material of a more technical nature.

5 Discussion

Although IN capability sets provide a useful framework for deployment of basic services in the short term, we believe that a radically different approach is needed if SPC digital exchanges are to be exploited to the full. In our view, the underlying shortcoming with IN capability sets is that, from an enterprise viewpoint, they have been designed around a number of highly specialised functions geared towards the simplification of marketing and tariffing concerns. Consequently, subscribers are given very limited control over services, meaning that their requirements often cannot be met, and it is difficult to deal with run-time feature interactions.

In this paper we have outlined an alternative model which gives subscribers more flexibility and control, and effectively eliminates run-time feature interactions. We have also shown how the model can be extended to support flexible UPT services and automated, interactive services.

Although our model eliminates some of the problems inherent in IN, it also introduces new ones. Policy specifications may be quite complex and it is certainly more difficult for the average user to write a policy than to set up IN features. Additionally, the available functionality is not packaged into convenient units as it is for IN. Consequently, there are a number of enterprise considerations which have to be addressed differently.

We have built prototypes to demonstrate some of the concepts outlined in this paper. Currently, we are continuing to experiment with the agent model by constructing scenarios and designing a negotiation protocol and policy specification languages around them.

References

[C+93] E. Jane Cameron et al. A feature-interaction benchmark for IN and beyond. *IEEE Communications Magazine*, 11(3):64–69, March 1993.

[CS+94] David Canfield Smith et al. KIDSIM: Programming agents without a programming language. *Communications of the ACM*, 37(7):55–67, July 1994.

[HH94] Andy Harter and Andy Hopper. A distributed location system for the active office. *IEEE Network*, 8(1):62–70, January 1994.

[Inta] International Telecommunication Union - Telecommunication Standardization Sector. *ITU-T Rec Q.1201 – Principles of intelligent network architecture.*

[Intb] International Telecommunication Union - Telecommunication Standardization Sector. *ITU-T Rec Q.1211 – Introduction to intelligent network capability set 1.*

[Kay84] Alan Kay. Computer software. *Scientific American*, 251(3):41–47, September 1984.

[M+94] Tom Mitchell et al. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):81–91, July 1994.

[Mae94] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40, July 1994.

[RLU94] Mike Rizzo, Peter F. Linington, and Ian Utting. Integration of location services in the Open Distributed Office. Technical Report 14-94, Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent CT2 7NF, United Kingdom, August 1994.