Extended Policies for Automatic Service Composition in IMS

Cuiting HUANG, Noël CRESPI
Department of Wireless Networks and Multimedia Services
Institut Telecom, Telecom SudParis
Evry, France
{cuiting.huang, noel.crespi}@it-sudparis.eu

Abstract— **Service creation by composing existing services and/or network resources is considered by Telecom and Internet industries as the trend for service provisioning in Next Generation Networks. Related research work is being carried out for years, and various service composition approaches have been proposed by different standardization organizations and companies catering to different network requirements. However, the requirements of automaticity, flexibility and runtime adaptation are still open issues for operators and service providers in the actual service environment. This paper, relying on a SCIM (Service Capability Interaction Manager) based service composition model, attempts to extend the Policy concept and introduces two additional functional modules, Capability Policies Repository and Policies Comparator, for addressing some of the automatic service composition issues in service provisioning process. With these extended policy and functional modules, the SCIM based model is enhanced with attributes of intelligence, flexibility and runtime adaptability: it enables delivering competitive and revenue generating service at a more rapid pace; reducing service creation and maintenance costs; and optimizing service lifecycle.**

Keywords- IMS; S-CSCF; iFC; SCIM; Policy

I. INTRODUCTION

IP Multimedia Subsystem (IMS) is widely acknowledged by Telecom operators as the reference architecture for Next Generation Network (NGN). It was defined from 3rd Generation Partnership Project (3GPP) Release 5 specifications as an access independent, all IP based, and service control enhanced architecture [1]. It has gone towards making service development more flexible, such as the overlay architecture that enables the convergence of voice, data and multimedia traffic over an IP based infrastructure, the unified service control layer based on Session Initiation Protocol (SIP), the capabilities layer and the Initial Filter Criteria (iFC) based service invocation mechanism. However, regarding the rapid changes going on both inside and outside Telecom industry, especially for the emergence of Web 2.0 mashups from IT world, above mentioned evolutions in IMS are not enough to ensure its prominent role in next generation service delivery. Therefore, a more advanced service delivery mechanism relying on IMS infrastructure is considered as necessary.

Comparing with the initial definition of service delivering in IMS, 3GPP has proposed Service Capability Interaction Manager (SCIM) as a standalone entity located between S-CSCF and application servers for managing the interaction among the Capabilities and/or application services [2]. However, years later, the specification for this entity is still limited to a few sentences. The fact that nobody knows exactly what SCIM should be provides SCIM a great opportunity to answer the currently unresolved problems in service composition. Some so-called SCIMs have been proposed with different interpretations for its functionalities, e.g. SCIM defined by Ubiquity [3], WCS SCIM provided by Convergin [4], SCIM defined by Alcatel-Lucent. In our research work, we have proposed an enriched SCIM model with our own interpretation for its functionalities [5] for addressing today's service composition requirements within IMS environment. However, several well known issues such as automaticity, extensibilities, convergence, user centricity, still require further evolutions in this proposed model.

The rest of the paper is organized as follows: Section 2 provides a brief introduction of the traditional S-CSCF based service delivery mechanism and our SCIM based service composition model; then Section 3 describes the extended policy concept in the SCIM based model for resolving the automaticity issue in service composition process; Finally, in Section 4, we conclude by discussing this proposal regarding the actual service environment and network evolution trend.

II. SERVICE DELIVERY WITHIN IMS

A. Traditional S-CSCF based service delivery mechanism

Within IMS service architecture, a capabilities layer is defined which contains a set of secured and combinable application-independent building blocks that offer generic functionalities to support various multimedia applications. Such building blocks are called *Service Capabilities* by 3GPP, and *Enablers* by Open Mobile Alliance (OMA). Some well known capabilities can be cited as Presence, Group Management, Messaging, Location, Network Address Book, etc. The existence of such a layer brings a new perspective for service creation, i.e. a service capability provides a predefined generic functionality so that a new and more revenue generating service can be created by combining multiple capabilities.

In order to aggregate the different capabilities into a unified service, IMS always uses the S-CSCF based service chaining mechanism in which all the relevant capabilities are chained into the SIP signaling path through the predefined Initial Filter Criteria (iFC) [6]. In order to facilitate the description of S-CSCF based service chaining process, Fig. 1 depicts the simplified IMS architecture.

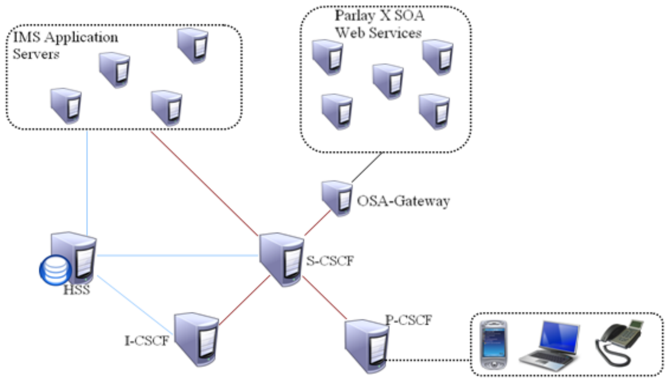

Figure 1. Basic IMS functional architecture

That's to say, once S-CSCF receives a request from user, a User Profile which contains an ordered list of iFCs is retrieved from HSS. S-CSCF then verifies each iFC following the priority order. Once a Trigger Point in the iFC is matched, S-CSCF dispatches the SIP message to the corresponding Application Server (AS). After processing the request, AS sends back the response to S-CSCF, and S-CSCF continues to analyze the unevaluated iFCs and determines whether to route the SIP request to next AS, or terminate the service chaining and send back the response to end user [7].

This iFC based mechanism is the original concept for service delivery in IMS environment and is too simple facing with the complex NGN world requirements. Some known issues in this regard are as follows: (1) Within this iFC based service chaining, all the services are invoked sequentially, that will incur the efficiency limitation in a case when some capabilities have no interdependence among them and can be executed in parallel. (2) It also brings some issues for S-CSCF itself, e.g. it may cause a heavy processing by S-CSCF in addition to other tasks it already has. (3) Multiple interactions between S-CSCF and ASs may lead to increased end-to-end latency for service delivery. (4) The limitations on flexibility and extensibility (e.g. convergence with other types of services outside the IMS scope) are some unresolved problems faced by this mechanism.

B. Alternative SCIM based service composition model

Addressing above challenges, a more advanced mechanism for complementing the S-CSCF based solution is considered necessary for future service delivery. Different standardization organizations (e.g. 3GPP, OMA, Parlay) or projects (e.g. SPICE, EXOTICUS, SERVERY, etc.) initiated to propose solutions for resolving some of the specified issues for service composition. According to their proposals, adding a new entity for handling the service composition issues becomes an implicit consensus for the service composition evolution. One of such well known entities is SCIM (Service Capability Interaction Management) proposed by 3GPP.

Considering above actualities, we adopt the SCIM as the service composition management component in our current approach and try to enrich this concept for addressing some of unresolved service composition issues in IMS environment. To achieve this goal, a SCIM based service composition model is proposed in [5]. In this model, SCIM acts as an intermediate between the network control layer and the service layer, and enables the reuse of service capabilities as shown in Fig. 2.

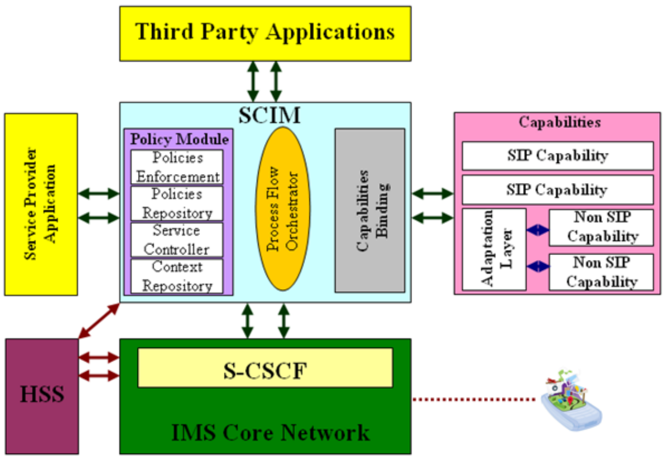

Figure 2. SCIM in IMS environment

When S-CSCF receives a request from end user, it transfers this request to SCIM instead of routing it to the application server. The information extracted from these requests triggers the execution of the appropriate SCIM application logic. Therefore, the incoming message is either re-routed, or transformed, or mapped onto multiple messages, or rejected by SCIM.

From the functional view, the main role of SCIM is to enable the reuse of the existing capabilities [8], which are relevant to the actions like discovery, access, and communication among capabilities. That implies that an SCIM will act as an authorization and authentication controller for providing and controlling the accesses to the capabilities for application servers. It will also be in charge of manipulating the collaboration among different capabilities, which means it should ensure all the relevant capabilities are invoked in proper order and with precise event data, which is related to the term of "*Policy*" in this model. Moreover, in order to facilitate the service creation process, providing an infrastructure to expose the capabilities towards higher layer also plays an important part in this service creation model.

The advantages of this proposed SCIM model are: it simplifies the iFC execution logic, releases the extra processing responsibilities from S-CSCF by shifting the capabilities interaction handling task from S-CSCF to a standalone entity; it provides a unified capabilities access interface and completely isolates the application layer from IMS core network; it does not need a great modification to the already defined substantial core network architecture, since SCIM is based on SIP and can be seen by S-CSCF as a normal SIP application server.

In above discussed approach, the service composition management model mainly focuses on dealing with the simple SIP based service composition processes. Some well known issues for service composition, such as automatic services aggregation, extensibilities, convergence, user centricity, still have not been taken into account. Addressing some of these challenges, we attempt to refine and enrich our previous service composition model by extending the "Policy" concept which is used in this SCIM model.

III. EXTENDED POLICY IN SCIM BASED SERVICE COMPOSITION MODEL

A. *Policy analysis for previous SCIM based model*

As introduced in [5], this model use the term "*Policy*" to represent the service composition information, and relies on the *Policy Module* to apply these policies to the passing requests. Fig. 3 depicts the basic components contained in SCIM.

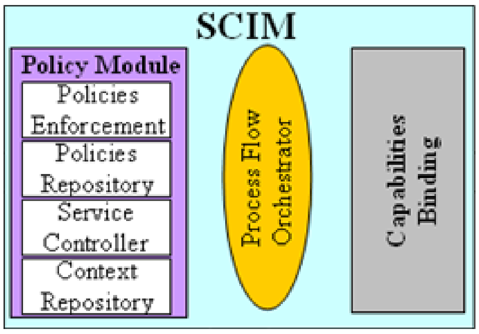

Figure 3. Basic components in SCIM

Such polices can be considered as formalisms that are used to express business, engineering or process criteria. Generally, a policy can be represented by a combination of a condition and action to be performed if such condition is true. The "action" in policy can be an invocation of a function, script, code, or workflow. And for the "condition", it can be Boolean predicate that yields true or false, or it also may be extended to some other machine readable data [9].

For the service composition management process, a policy should go through the following steps: (1) Evaluate the policies using received messages and other contextual information; (2) Execute the policy action resulting from the positive evaluation of the policy conditions; (3) Return a policy decision to the requestor.

In our SCIM based service composition model, all of above steps mainly rely on the *Service Controller* component. When SCIM receives a request originated by an end user and transferred by S-CSCF, a corresponding policy is retrieved from *Policies Repository*. Then, *Service Controller* evaluates it using information extracted from the incoming message or other contextual information retrieved from *Context Repository*. According to the evaluation result, *Service Controller* executes an "action" as a consequence of the evaluation result. Such an action can be an invocation of one capability or multiple capabilities in the predefined orders, or termination of the message routing and send the final decision to end user, or even the rejection of the incoming message.

The above mentioned policy mainly relates to the composite service execution process. Its intrinsic variety makes it easy to be extended for addressing some specific requirements in service provisioning process, such as the automatic service composition.

B. *Enriched policy for automatic service composition*

Regarding the automaticity requirements in service composition, it always relates to two cases: (1) When the capabilities have been modified, the relevant composite service logic and the corresponding configuration should be automatically adjusted; (2) When user creates a new service by defining the functional part, the system should automatically create the appropriate composite service logic and select the corresponding capabilities with little or no direct human intervention. In the following part, we try to illustrate how to use the extended policy and the newly introduced components, Capability Policies Repository and Policies Comparator, to manage both of the automatic cases relying on the SCIM based service composition model.

1) *Case 1: Automatic update for composite service*

For most of the capabilities, it is normal that they will be updated regularly. Once such modification occurs, it will be necessary to perform the corresponding adjustment in the service invocation logic and the corresponding service binding. Today, most of the updates are handled manually, which requires an enormous amount of work for both Capabilities Providers and the Application Service Providers. At the service capabilities side, each modification requires a cautious consideration regarding for the potential influence on all the relevant Application Services, which may be impossible if such capabilities has been involved in a large number of services for different purposes, and it may be possible that some of the services get neglected causing a crash for certain parts or being incompliant for the whole system. For the application services side, they also have to pay a great deal of attention for noticing the possible modifications on the relevant capabilities which they use. They have to update their services frequently for ensuring that all the capabilities are involved in a correct way and are used optimally for their systems. From this point of view, how to facilitate this service logic update process and how to automatically select the optimal capabilities according to the real-time changing network environment is a big challenge for improving the service creation environment.

Referring to the successful experiences in Web service composition, Policies can also be used to represent the capabilities and requirements of an existing reusable service as specified in WS-Policy [10] and WS-PolicyAttachment [11]. Traditionally, if a developer wants to use a certain service or capability offered by a service provider into his composite service, he has to read the related documentations, contact the service provider to understand the service metadata, or look at sample SOAP messages and infer the service requirements or its behaviors. But if the service provider could present the requirements and capabilities as policies and in machine-readable format, end user can use a policy-aware tool, which can access and understand the policy, to engage the corresponding requirements and generate the client code automatically [12]. This successful experience inspires us propose a solution for the automatic update problem within the SCIM based service composition model: that is the introduction of Capability Policy into the service composition model.

In this enriched model, policies are not only generated by SCIM for expressing the service capabilities invocation logic which are named as *Service Policies*, but also can be generated by the existing service capabilities or reusable services provided by other service providers which are named as *Capability Policies*. Once a new capability is introduced into the network or an existing capability is updated, it will advertise its capabilities and requirements to SCIM as a policy.

That is to say, the newly introduced or updated capability should initiate a connection between itself and SCIM, and then it embeds a policy which contains its capabilities and requirements information into the exchanged messages that may rely on Extensible Markup Language (XML). When SCIM receive these messages, it abstracts the Capability Policy from them and stores it at a policy repository. In order to avoid too many messages exchange, this policies advertisement can be based on the traditional stateless HTTP protocol since this process has no direct relationship with the service execution process which is generally stateful or based on SIP for accessing the capabilities located in the IMS environment, that also means we have to add a small module within the SIP based SCIM for receiving the incoming HTTP requests, that is a HTTP interface which intercepts the incoming HTTP requests advertised by capabilities for their policies and then triggers the corresponding policies repository for storing the incoming capability policies on it.

A usage example of the capability policy advertisement process can be cited as in Fig. 4: Service provider A provides a presence capability to end-users. In order to facilitate end user to discover and use this capability, they add an additional module in the Presence server with the description of its functionalities and usage requirements (i.e. Capability ID, functional description, Address, Location, Price, Security, QoS, etc.). And this additional module embeds these information into the exchanged messages and send them to SCIM as indicated in upper part. As shown in step 1, at the beginning, in order to encourage end-user to use this presence capability, provider A configures the usage of this capability as free, but with low security and medium QoS guarantee. After a few months, the security and QoS are ameliorated, and service provider begins to charge this presence service, thus the price, security and QoS portions are re-configured as shown in step 2. Thus an updated capability policy is re-sent to SCIM.

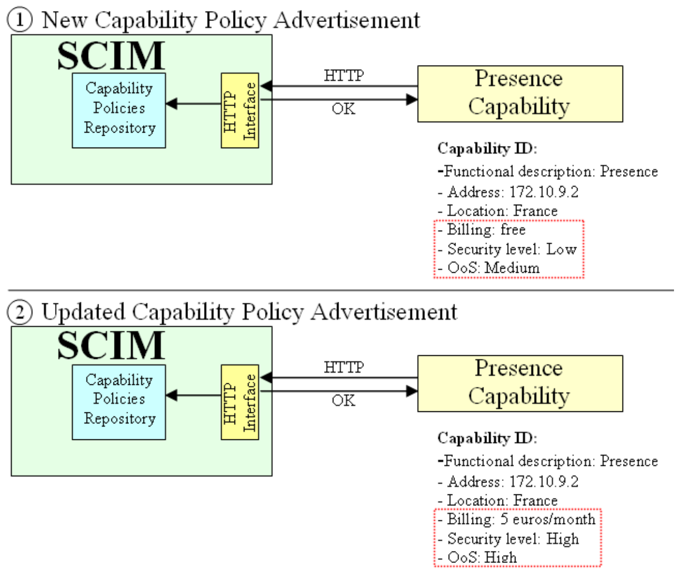

Figure 4. A usage example for capability policy advertisement

In order to well adapt the new introduced capability policies into our model, two new components – *Policy Comparator* and *Capability Policies Repository* – are introduced into SCIM as additional functional entities for handling these Capabilities Policies as shown in Fig. 5.

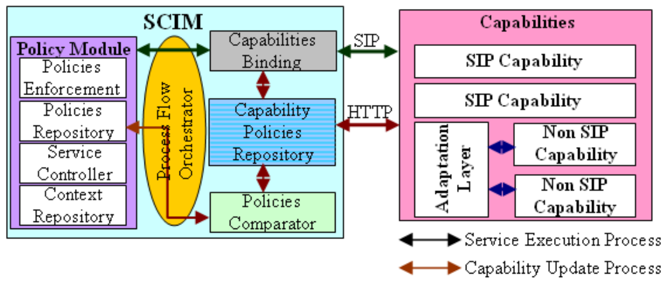

Figure 5. New introduced capability policies repository and policy comparator in SCIM

When SCIM receives a policy from capability, the *Capability Policies Repository* tries to find out if a previously advertised policy from the same capability has been stored by matching the unified capability ID indicated in the received policy and in already stored policy, and then it makes a decision for this incoming policy: *Create* if no previously stored policy has been found or *Update* if yes.

(1) If the decision is *Create*, a new item is created in the Capability Policy Repository, and a relevant capability binding item is generated and stored in Capabilities Binding component according to the necessary information extracted from this incoming policy for facilitating the access to the newly introduced capability.

(2) If the decision is *Update*, a previous existing policy is selected and replaced by the new incoming policy. The Capabilities Binding component updates the corresponding binding item according to the capability modification information extracted from the received policy. At the same time, if the incoming policy is marked as *Update*, it also will be

routed to the *Policy Comparator*. According to the capability ID extracted from this incoming policy, *Policy Comparator* contacts *Policies Repository* for retrieving all service policies relevant to the same capability. By analyzing the relevant service policy/policies with the incoming capability policy, Policy Comparator makes a decision of whether the existing service policy should be adjusted according to the requirements indicated in the incoming capability policy. If the decision is positive, the corresponding service policy will be updated and then re-stored in the Policies Repository if it has been configured as automatic update, or an alarm will be displayed to service maintainer or user for demanding an update manually.

Within this automatic service update process, there may exist two different solutions: (1) *Policy Comparator* keeps the previous selected capability in the service policy and just modifies some corresponding configurations according to the new requirements extracted from the received capability policy, e.g. add the new capability access requirements into the *Condition* portion in previous service policy; (2) *Policy Comparator* removes the previous selected capability from the service policy, and then re-connects with *Capability Policies Repository* for finding an optimal capability, which has the same function as previous one but with the different usage requirements, to replace the previous one and re-configures the necessary portions in the service policy according to the information extracted from the newly selected capability policy.

2) *Case 2: Automatic creation for composite service*

With above mentioned extended policy and additional components, *Policy Comparator* and *Capability Policies Repository*, the automaticity concept can be extended to the service creation process. That is to say, when a service is created by a professional or non professional user, user just needs to define the abstract functional part for the service which means defining the functions that are to be engaged in this created service, and he dose not need to know or define which concretely existing capabilities are invoked and how such concrete capabilities can be accessed and to be configured. It will greatly facilitate the service creation process and would encourage non professional end user to create more personal services in a more friendly way, since user does not need to spend much time to get to know about the concrete capabilities and network environment which is always very difficult or even impossible for non professional user.

The complete process for utilizing extended policy in automatically service creation process is as follows: when a non professional end-user wants to create a personal service through some friendly service creation environment, such as some Natural Language Enabled Service Creation Environment or some Friendly Graphical Service Creation Environment, he can just enter his requirements of this service by natural language or in some predefined fields or boxes, and such environment will transfer user's requirements as a policy and send it to SCIM. According to the received policy, Policy Comparator retrieves the corresponding capabilities policies, and then analyzes the user's policies with all the corresponding capabilities' policies. After contacting with the predefined capabilities selection rules and the contextual information, Policy Comparator makes a decision for creating a new service policy which represents the capabilities invocation logic and stores it in the policy repository. Fig. 6 depicts the simplified automatic composite service creation process.

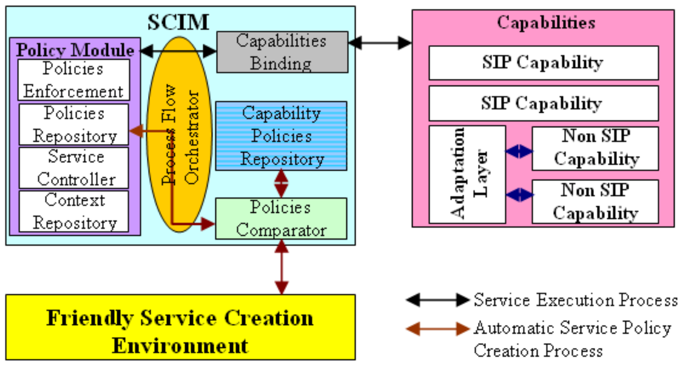

Figure 6. Simplified process of automatic service creation

If a user has created a service and just wants to update his/her previously created service, after receiving the incoming policy, Policy Comparator retrieves the corresponding existing policy stored in Policy Repository, and analyzes all three types of policies: new incoming user policy, previously existing service policy, corresponding capabilities policies, and then make a decision for updating the existing policies according to the results of the analysis.

C. *Additional consideration for the extended policy and system analysis*

For both of above two cases using extended policy for automatic service composition, the policy advertised by capability or by end user can consist of the "*Capabilities*" portion and the "*Requirements*" portion. In the case of capability advertised policy, the "*Capabilities*" portion contains some functionalities descriptions of this Capability; the "*Requirements*" portion contains the information about access authorization requirement, use conditions, the necessary input data, etc. In the case of user advertised policy, the "*Capabilities*" portion contains the information about the terminal device's capabilities (e.g. media capabilities, access capabilities), end-user's authorization (which types of services end-user can use, which services user has not subscribed to), user's location, etc. For the "*Requirements*" portion, it includes all the information about user's service design idea (which functions user hopes to invoke in this new service, the order for the function invocation, etc.). All of such information will be abstracted into a policy by the upper service creation environment such as Friendly Graphical Service Creation environment, Widget Based Service Creation environment, Natural Language Enabled Service Creation environment, and so on.

This extended policy and architecture modifications are consistent with our previous SCIM based service composition management model and it has very limited impact on the underlying architecture, especially has no influence on the basic architecture of IMS since all of the modifications are inside the SCIM which is a standalone entity above the IMS control layer. It provides the possibilities of resolving the automatic service composition issue, which is one of the main trends of evolving service composition technologies.

IV. Conclusion and Discussion

In this paper, we explored the service composition mechanism for innovative service creation relying on the IMS infrastructure. After presenting current S-CSCF based service delivery mechanism within IMS environment and introducing an alternative solution for service composition which uses SCIM as a relevant functional component to manage the service composition in the IMS platform, we enriched this SCIM based service composition model by extending the policy concept for addressing the automatic service composition issue.

The advantages of such extended policies are explicit: it greatly facilitates the service maintenance both for application service providers and the capabilities service providers, since each modification at service capabilities side can be observed by SCIM through the capabilities advertised policies and thus the influence of the corresponding composed service can be handled automatically by the corresponding components. Such extended policies also can facilitate the service creation process for non professional end user by enabling the automatic services selection, composition and monitoring. End user's requirements, personal configuration information, as well as his device's capabilities can be extracted into a machine readable policy and sent to SCIM, thus the information encapsulated in the policy can be engaged into the service creation process automatically according to the predefined composition rules. It would allow the creation of novel, compound service in a more user centric way. Moreover, such extensibility does not require any substantial changes to the already defined architecture and service execution methods. Thus, the usage of such extended policy will not only be limited to IMS but also can be adopted easily by other non-IMS based networks.

However, as presented in this paper, our current study mainly focuses on how to use the policy for transferring the necessary interoperable information from capabilities side and the end user side to the service composition manipulator – SCIM, and how these received information interact with other existing information. Some challenges for dealing with the automatic service composition still need further consideration. These issues can be cited as follows: the definition of an effective and intelligent service selection mechanism is an indispensable feature for this service composition model, especially the reality that more and more reusable services are introduced into the network, and as the demands from end users become more and more complex, how to ensure that the selected services are the optimal ones for end user at runtime is one of the key success factors for future service provisioning; Meanwhile, receiving a great amount of information from capabilities and end user, SCIM should provide an appropriate mechanism for balancing these information and avoiding the possible increased end-to-end latency.

For the future research, we will continue our work in the service composition by refining current SCIM based service composition management model. Not only above unresolved problems for automaticity, but also the convergence between IMS and web 2.0 will be taken into account for future service composition system design.

Acknowledgment

The authors would like to thank Raheel BALOCH for his thoughtful and detailed review.

References

[1] 3GPP TS 23.228, "IP Multimedia Subsystem (IMS) – Stage 2"

[2] 3GPP TS 23.002, "Network architecture -Release 4"

[3] C. Buliton, "An introduction to the Service Capability Interaction Manager (SCIM)", Avaya white paper, Sept. 2007. https://udn.devconnectprogram.com/products/whitepapers/final-wp_scim-092007-lb3700.pdf

[4] T. Zoller, F. Deza, "SCIM for IMS and Legacy Networks", http://convergence1.netcentrex.net/index.php?option=com_content&task=view&id=186&Itemid=20

[5] C. Huang, N. Crespi, "Enriched SCIM for Service Composition within IMS Environment", International Conference on Engineering Management and Service Sciences EMS 2009, Sept. 2009

[6] A. Gouya, "Service Interaction Management in Next Generation Network Service Control Overlay: IMS", PhD thesis, Jun. 2008

[7] A. Maaradji, C. Huang, N. Crespi, "Towards Personalized Service Composition on IMS: A Basic Approach", International Conference on Advanced Infocomm Technology 2009, Jul. 2009

[8] A. Gouya and N. Crespi, "SCIM (Service Capability Interaction Manager) Implementation Issues in IMS Service Architecture", IEEE International Conference, 4:1748-1753, Jun 2006

[9] OMA, "Policy Evaluation Enforcement Management" Approved Version 1.0, 05 Aug. 2008, OMA-AD-Policy_Evaluation_Enforcement_Management-V1_0-20080805-C

[10] W3C, "Web Services Policy 1.2 – Framework (WS-Policy)", http://www.w3.org/Submission/WS-Policy/

[11] W3C, "Web Service Policy 1.2 – Attachment (WS-PolicyAttachment)" http://www.w3.org/Submission/WS-PolicyAttachment/

[12] A. S. Vedamuthu, D. Roth, "Understanding Web Service Policy", Version 1.0, Jul. 2006, http://msdn.microsoft.com/en-us/library/ms996497.asp