EXTENDING TINA TO SUPPORT SERVICE CUSTOMIZATION

Linas Maknavičius[1], Gautier Koscielny[2] and Simon Znaty[1]

[1]ENST Bretagne, RSM Dpt.,
rue de la Châtaigneraie, 35512 Cesson Sévigné, FRANCE
[2]Valoria Lab., Université de Bretagne Sud
rue Yves Mainguy, 56000 Vannes, FRANCE

{ Linas.Maknavicius | Simon.Znaty } @enst-bretagne.fr, Gautier.Koscielny@univ-ubs.fr

Abstract: In the telecommunication and multimedia service market which is foreseen to greatly flourish in the coming years, an important issue will be the accommodation of 'standard' services to the user's requirements and preferences, i.e., the capabilities of customizing services. Research on this matter is still in its infancy. TINA provides basic but insufficient perception of service customization. In this paper, we analyze the TINA architecture in this sense and we strive to extend it by introducing several flexible models with ascending degrees of customization. These range from a simple service options offer and a tailored service instance creation to a total user participation in the service behavior. An intermediate level consists of a dynamic user-driven customization based on component groups and generic service types. We discuss the mechanisms for the proposed customization levels as well as related issues.

Keywords: Telecommunication services, TINA, customization, group of distributed components

1 INTRODUCTION

Today's telecommunications environment is changing dynamically: the market is being liberalized, the industry is repositioning itself into alliances and towards new partnership forms, and the actors segregate into distinct service providers, network operators, retailers, service traders, service designers and other stakeholders.

The advanced services are emerging. In this context, service providers' major concerns are the following: to contrive and introduce new services with the utmost rapidity, to guarantee an appropriate service level through service management and to

L. Kutvonen et al. (eds.), *Distributed Applications and Interoperable Systems II*
© Springer Science+Business Media New York 1999

meet particular user requirements. One sort of user requirement is a desire to tailor a service he/she is consuming. E.g., the user may wish to get a video in HDTV 16/9 format from a Video-on-Demand (VoD) server, instead of a default VHS-quality video. Usage preferences, service parameters and features, functional aspects of a service, specific attributes of the service, operating procedures, technological and institutional constraints etc. - all these items present the basis for telecommunication service customization. The customization is defined as a facility users are provided with to modify a service in order to accommodate it to user's individual needs or to the needs of a user group, or to their operating procedures.

The advanced and multimedia services present further potentialities to adjust them according to their multiple options and functional features; on the other hand, the advent of 'smart users', who actively participate in service provisioning, also fosters service customization [9].

Various network and service architectures - Internet, IN, TINA, DANSE, UMTS - offer several concepts for service customization [9]. For example, IN (Intelligent Network) technology implements a specific bloc which manipulates user service parameters. It is believed that IN will be gradually evolving to TINA, an architecture for telecommunication services. Thus they both deserve to be surveyed. Whereas IN customization, although confined to its service field, is rather widespread and proven, TINA needs a closer consideration.

The main TINA customization principles are presented and evaluated in Section 2. Then, Section 3 describes the proposed service customization levels and gives some details on them. The levels differ in their degree of dynamics and reflect a growing user's involvement in a service. Section 4 discusses related research and open questions, and closes the paper with conclusions.

2 TINA SERVICE CUSTOMIZATION

2.1 Architecture

TINA (Telecommunication Information Networking Architecture) [13, 14] is an open software architecture for the creation, deployment, provisioning and management of advanced telecommunication services in a global (up to world-wide) scale. It dissociates service functions from the complexities of an underlying transport network infrastructure. This dissociation allows the designers and developers to focus on service specification alone. A TINA service is seen as a set of distributed object-oriented software components interacting through their operational or stream interfaces. The components distribution over different computing nodes is supported by the *Distributed Processing Environment*, or DPE (basically, an extension of a CORBA platform to suit telecommunication applications) which provides the execution medium for applications and actually ensures transparent interactions among service components.

The TINA Service Architecture (TSA) provides a framework for service development by specifying operations among service related software components. The main components are illustrated in Figure 1. A provider is represented by a service component within the user's domain - the Provider Agent (PA). It is designed to set up

a trusted relationship between the user and the provider, to convey all kind of user's requests and to receive invitations. Likewise, a user is represented within the provider's domain - the User Agent (UA) which responsibilities are session creation, suspension, resumption and deletion. When a request to create a new service session is issued, the UA invokes an operation at the Service Factory (SF) component which instanti-

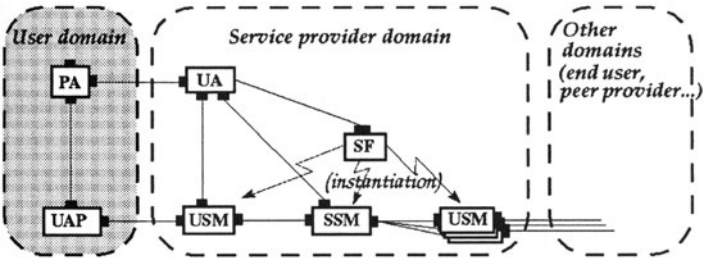

Figure 1. TINA service architecture, TSA (simplified).

ates specific service components: Service Session Manager and User Service Session Manager (SSM/USM). In fact, the SSM and USM decompose a service session into a provider part and a user part: the former represents the core service behavior common to all users, while the latter contains the information and service capabilities which are local to a particular user. The User Application (UAP) component models a service application in the user domain and enables him/her to make use of service capabilities. Thus, the described service components hold well-defined roles and capabilities that are felt to be applicable to the most telecommunication/multimedia services.

2.2 General requirements to support service customization

The necessity to customize services is underlined in the very objectives and requirements of the TSA [14]. A TINA service must be based on a flexible and granular model and thus be customizable in order to satisfy specific requirements of various customers. This is achieved by offering the subscribers and the end-users some direct control in managing their services:

→ customization of pre-choices/pre-conditions on access to other stakeholders (we call this type *customization a priori*);

→ customization of service usage enabling the users to tailor a layout and the functions of service components in the provider domain (this is a *genuine service customization*);

→ customization of configuration of user-system related resources (this type may be regarded as *physical customization*).

2.3 Types of customization and associated objects

To enable different participants to tailor services to their requirements, TINA advocates three types of customization: by a service provider, by a subscriber, and by an end-user. These actors accommodate service characteristics by modifying corresponding

service profiles (*cf.* Figure 2). The profiles are the informational objects describing customizing attributes for the given participant, i.e. identifying particular preferences and requirements set by the participant. They model a desired service behavior. The service profiles are in essence of three types - Provider Service Profile, Subscriber Service Profile, End-User Service Profile -, and embody three distinct "vertical" aspects (Figure 2): service settings, usage constraints, configuration requirements. These

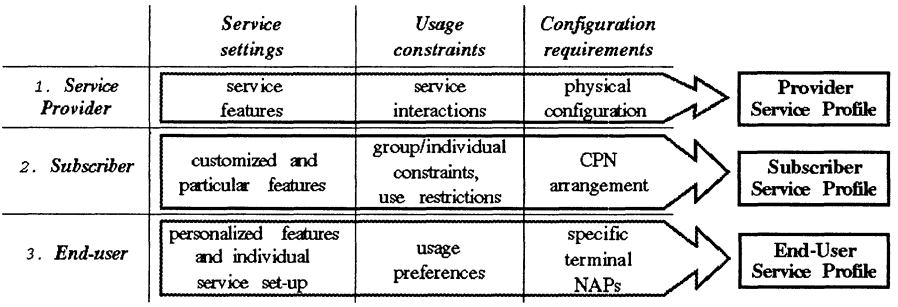

Figure 2. TINA constituents for customization and their meanings.

service profiles are taken into account when service is instantiated (a service session is established) and by that mean affect service characteristics. The service profiles' relationships with other informational objects are laid out in Figure 3. (for this, the OMT notation is used). The three service profiles make up the Customization Infor-

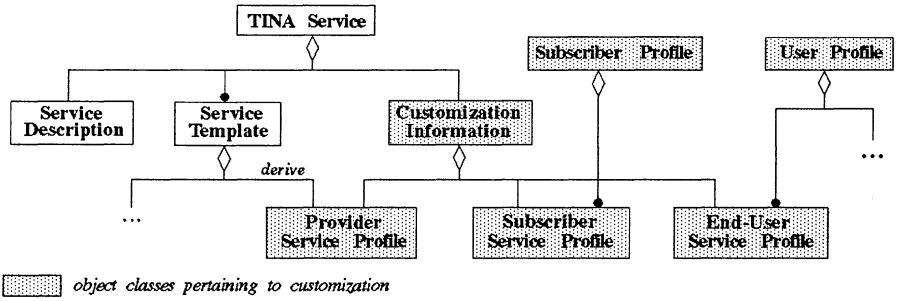

Figure 3. TINA service informational model with customizing objects.

mation object included in TINA Service. This latter object class also encompasses the Service Description, an object giving a textual statement about the service, and the Service Template, representing informational and behavioral characteristics of a specific service type (instance name, id, required services, alternative service parameters) [14, 15]. The Subscriber Service Profile and the End-User Service Profile are parts of more generic objects called Subscriber Profile and User Profile. They contain all relevant information regarding a subscriber and user respectively, such as usage contexts, active session descriptions and invitation handling policies.

2.4 Assessment and current limits

TINA provides sound and generic basis for service customization. The lining up of the customization by participants - service provider, subscriber, and user - is of particular value. This approach enables each group of participants to bias service characteristics accordingly to their preferences and implied constraints, by modifying corresponding service profiles. Besides, considering the "Service settings" aspect throughout the service profiles (*cf.* Figure 2), it appears that the provider-enforced service features constituent is a superset of the subscriber-customized features constituent. This latter in its turn is a superset of end-user individual features. Consequently, if only the "Service settings" customization aspect is taken into account (i.e., there are no configuration nor usage restrictions), a tailoring made by a provider restricts the customization alternatives for the subscribers and subsequently for the end-users. Indeed a subscriber, when modifying its Subscriber Service Profile, imposes or limits certain options for its subordinate users.

On the other hand, the TINA customization approach is not flexible enough. The main drawback is the rule stating that the modification in a service profile does not affect to the service already instantiated: the new information in the profile is assumed solely at the time of the next instantiation of the service. Therefore, if a participant aims to substantially tailor a service he is perceiving, he ought to leave a current session and to initiate an opening of a new one. E.g., consider a road traffic information service: a truck-driver using this service in an audio-only form is informed of a highly congestioned zone nearby; to avoid a tailback, he decides to deviate from his initial itinerary. As he is aware of the possible difficulties in a new area because of a heavy traffic there, he wishes to receive the information covering this new area in a graphical form on a GPS terminal. He has no other solution as to stop the current service session, to convey his wishes in order to update his service profile and to initiate a new, modified service session. This is a hard inconvenience. Ergo, we categorize TINA customization as *"piecemeal"*.

Admittedly TINA introduces an interesting concept of dynamic customization, but it is rudimentary. It is considered as being effective after the service instantiation and achieved by modifying the so-called customizable data which is supposed to be bundled directly into the service instance. Nonetheless, no details about the nature of these data nor any guidelines for implementing the dynamic customization are given.

TINA misses computational model for customization, i.e. does not provide a service instantiation procedure according to service profiles. It is only said that the profiles are checked for modifications when instantiating service components. It is still not clear how these profiles can influence a new service session in concrete terms. In consequence, the customization aspects are not fully integrated into the whole service architecture.

Moreover, the scope of the profiles is sometimes interpreted inconsistently, that is, the profiles are also seen within the user lifecycle and subscription management model.

To summarize, TINA offers a promising and conceptual framework, but its object structure is strict and too static to rapidly assume frequent changes in the service

environment and to satisfy the requirements for adaptability; the customization aspects, even static, are not sufficiently detailed.

In the remainder of this paper, we attempt to surpass these limitations and disadvantages by proposing a syncretic and ascending view for service customization including several distinct levels. We adopt the concept of service profiles, but apply it for even more powerful customization.

3 CUSTOMIZATION SUPPORT AND LEVELS

3.1 Enterprise viewpoint

The purpose of service customization is to ease, improve, and promote the access to and the usage of a service by providing the means to adjust the service traits dynamically and at low cost. To meet this objective, two communities are formed: a customizing and a customizable system. To model interactions between these two communities, five main roles are involved: service consumer, retailer, third-party service provider, peer service provider, and content provider (an anonymous user, an identified user and a subscriber are the possible stakeholders for the consumer role). A customizing system must be awarded with the tools to change/update the used (subcontracted) service features. A service portion supplied by another (peer or third party) service provider has also to be tailorable. The customization process is governed and limited by the service contract established between two roles.

3.2 Generic computational model

We clearly need an extensive structure and mechanisms to enable any type of service accommodation. A proposed generic computational view is depicted in Figure 4[1]. It extends the TSA component UA (User Agent) by appending supplementary components.

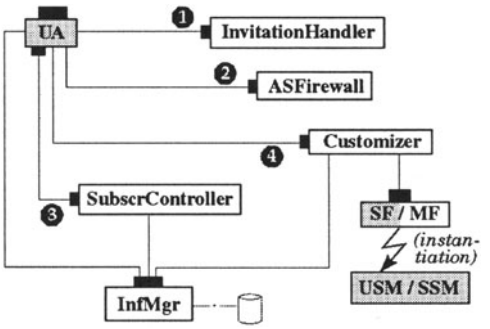

Figure 4. Customization computational model (simplified).

One of them, Invitation Handler, is intended to perform the appropriate actions when

[1] "Standard" TSA components are outlined in grey.

an invitation to join a session or a call for a scheduled session is received (1). These actions will depend on the activated invitation handling policy which is included in the User Profile informational object (recall Section 2.3). The possible policy values are "accept/reject/forward/follow-up" invitation. Once a session call accepted, the component ASFirewall plays a role of a "firewall" in an access session (2): it identifies, filters and authenticates the users willing to access a service, as well as grants the access rights to them. Then, a subscription process is managed by another specific component called SubscrController (3) which additionnally calls the information manager (InfMgr) in order to extract user subscription information and available service lists. Finally, a service is customized (4) according to a specific user service profile; the Customizer calls the appropriate service or service module factories (SF/MF) which create tailor-made service components. The concept of module factory is explained in the subsequent sections.

3.3 Distinct customization levels

In order to set a clear view of service customization, we choose the extent of user involvement in service provisionning as a criterion to "measure" the customization level. Therefore, we define several customization degrees with a growing user participation when he/she configures or modifies the service:

level 1) at the time of service session instantiation, a user conveys his requirements and preferences to a provider; this level is separated in two ways:

1a) provider-oriented service offer when a provider presents a set of service options or alternatives to a user;

1b) user-oriented service instantiation on demand when a service is made-to-order according to the user profile;

level 2) during the service provisioning, and using the open management interfaces set on the service components, so that user could activate specific service features;

level 3) active services that allows a user to inject his/her own added value into running services, through scripts or mobile agents.

3.4 First level: personalized service instantiation

3.4.1 Option-based switchboard. A step surpassing the provision of standard mass services is the offering of static service options to the user. So the user would be able to "switch on" certain options, either the core features or the details of the provided service. The suggested alternatives take the form of a pull-down menu or "on-off" choices at the user interface.

For example, a user subscribes to a "Call Forwarding" telephone service, which allows to program a number to which he/she wants the calls to be forwarded from his/her personal phone. The user may choose between the following options: "forward on no answer", when calls are forwarded after, say, 5 rings, and/or "forward on busy"

option for calls that come in when the line is busy. Additionally, the user can possibly make a choice between forwarding to a national/international number.

Applied to the TINA context, the options positioning on the "switchboard" pertains to multiple predefined types of SSM/USM components, being instantiated according to the selected option(s). Each component matches one option or a set of related options. When instantiating, no subscriber/user service profiles would be examined. The service characteristics would be merely extracted from the Service Template informational object which is distinctive to the selected service type. In other words, the explicit customization by the user/subscriber shall not take place in this case, because the Service Template object alone (*cf.* Figure 3) will be assumed. Therefore, this customization scheme is provider-centric. At this level, the service construction is rather simple as it depends only on the option choices (that is, predefined component types).

3.4.2 Tailored service composition. The service options scheme may prove to be restrictive for the service provider considering that there should be as many service component implementations as possible options (or groups of congruous options). Moreover, in order to create these components, the provider should keep a large number of distinct Service Factories active. Their number is directly proportional to the quantity of options. On the other hand, the apprehension, choice and positionning of a large number of options may be troublesome or weary for the user.

That's why we introduce a new, transparent, "on the fly" service construction technique. It is based on the User Profile usage: the profile determines not only global service parameters and specific features, but also (indirectly) an appropriate service logic.

A possibility to create a service by combining existing components is raised in [16]. For example, a user will be able to watch a VoD film in black and white with stereo hi-fi sound simply by combining a B&W video object with a stereo hi-fi sound object together into a new user-specific object.

We identify several stages for this type of construction. In the first place, an abstract request, based on the user profile, is step by step transformed into an internal service structure. Secondly, we identify the components, or modules, to form up a service. They are supposed to be distributed, heterogeneous and communicating. Finally, these modules will be assembled by "connecting" their interfaces. Following this methodology, the proposed steps to made-to-order a service are illustrated in Figure 5. The user profile is mapped (1) to an internal arrangement consisting of a set of configurable modules. Their dependencies are modeled by a graph or a tree, where the modules of one layer can only interact with the modules from adjacent layers. These modules are the elementary customization units. They are combined together afterwards, in order to build a "conventional" service component. The modules have to be carefully selected (2), in accordance with the information given in the user profile. The next step is dedicated to the creation of the selected types of modules. We introduce a concept of Module Factory (MF) whose role is to instantiate (3), initialize and activate the

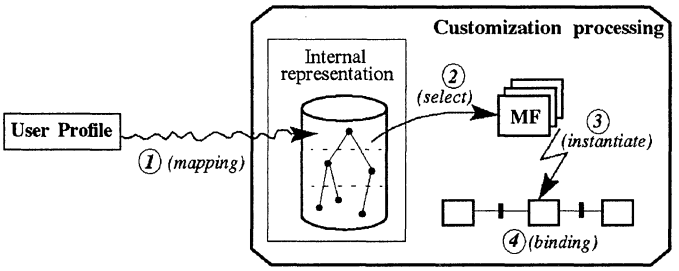

Figure 5. Model to compose a tailored service "on the fly".

modules. Finally, they are bound together (4) to make up a whole service component which is custom-designed and suites the given profile.

In order to specify the possible module dependencies, a directed dependency graph is built. This configuration graph is subsequently used to generate all possible module combinations. Here we adopt a hierarchically layered graph, although there are several other approaches: slotted, class hierarchy based, object message oriented graphs [5].

Our service composition method is transparent to the user since it makes a translation of the User Profile to the internal service structure. Other important advantage is that it allows to reduce processing in the provider domain as measured by a number of "factories" to be executed permanently. Indeed, a large number of customized services is potentially obtained with a fixed number of modules by combining them. In normal operation, a separate Service Factory (SF) would be needed for each such customized service. In our case, a Module Factory (MF) is needed for each module type, and the number of MFs is clearly lower than that of SFs. As a result, we gain in terms of processing.

3.5 Second level: customization interfaces

Instead of customizing a service once and for all, a user may want to activate or deactivate service properties during the session.

For instance, a groupware service like multimedia conferencing permits not only a textual information but also audio, images, graphics and video media use along communication channels. Each media corresponds to a particular feature of this service and a user may choose between different media mixes (e.g., audio+video, chat+audio, whiteboard+audio etc.) throughout the session. Each mix matches a specific facet of the same service. It would be useful to group those components together and let the users to manipulate these services through customization interfaces.

The concept of service group as proposed in [11] allows the encapsulation of a set of distributed objects sharing common features, a common behavior and whose interfaces may be disjoint. An immediate consequence of this model is to ensure the availability of the common services provided. Nevertheless it differs from traditional object groups whose main purpose is to provide fault-tolerance and load-sharing properties by passive or active replication of services [4]. Contrarily to a group of replicas where each object

offers the same interface, a service group may offer multiple external interfaces because of its heterogeneous contents. These interfaces reflect partially those offered by the group members who joined the group. Note that the set of external interfaces may be augmented by appending new components into the group.

The customizable service component group is based on the model described above. A component belonging to such a service group will match one or several service properties the user wants to enjoy during the session. Every property designates a *facet* of the service. A facet is represented by an external interface and will be activated dynamically. A set of properties (facets) composes a customizable service (*cf.* Figure 6). For example, several conferencing components with different properties like chat facet, <4:2:2 format MPEG video+stereo audio> or <4:2:2 video+mono> facet, and whiteboard facet will form the customizable conferencing service (e.g., facets 1 through 4 respectively in Figure 6, 2nd and 3rd being mutually exclusive).

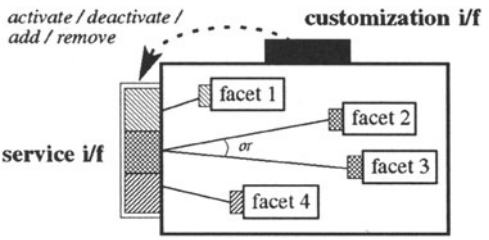

Figure 6. Customized service group as a set of service facets.

A service group interface is configured through an additionnal customization interface. This latter, made available to authorized users, enables them to adjust the service dynamically. The customization interface provides operations to activate/deactivate properties (i.e., to include/withdraw individual facets' interfaces to/from external service interface), as well as to add, remove, replace components within a group.

Multiple components may provide the same service while having different types definition. Generic service allow the substitution of a component by one of its variant while keeping the same external service interface. This is achieved thanks to the definition of a type conformance relation between similar services called the coercive compatibility relation [6]. Type conformance is generally defined by subtyping, an instance of inclusion polymorphism which specifies a substitution rule between types [2]. A type τ must replace a type σ in each context where σ is expected. For instance, interface subtyping relation as found in software architecture like CORBA[2] and RM-ODP is defined by interface extension, i.e., the addition of new operation signatures in subtypes. Note that contrary to CORBA, RM-ODP operation interface subtyping rules support parameter subtyping applying respectively the contravariance and covariance principle on argument and result types of an operation in the subtype [1].

[2]In CORBA, subtyping is achieved through multiple interface inheritance whereas generally inheritance doesn't imply subtyping.

The coercive compatibility relation is more general than the subtyping relation. It defines a mapping between a group interface and a set of different component interfaces representing the same service. For each operation of an external interface, there exists a coercion from this operation to the equivalent operation in component interfaces. Note that the correspondence may be incomplete (e.g., the service operation may have less or possibly more parameters than the facet operation). However, contravariance and covariance rules are respected and the parameter types of a facet operation corresponding to the parameter types of the external operation are generalized whereas the result types are specialized. Hence, to the service consumer point of view, the external interface remains the same even though a new component version is included. As an example, a whiteboard service may itself contain several components to draw curves. A user may replace a basic component previously active with a new version enabling B-splines drawing.

The proposed model of service facets group extends TINA Service Group (SGP) concept [14, 10] in that it allows a dynamic selection and activation of components at any moment during the service session. Other two main features of SGP are preserved in a facet group: it may span multiple DPEs, and is dependent on the prior installation and initialization of the group constituents.

3.6 Third level: active services

The previous level may be perceived as having a drawback: a user must be aware of the facets composing the service group, where facets are a pre-defined set of components. To avoid this restriction, a much more powerful model may be applied. That is, the active service model. It allows to dynamically extend or adapt a service functionality by injecting a supplementary, custom-made piece of code (object, script, or agent) into a service component at run-time. This approach ensures an optimum consideration of user customization requirements and continuous, smooth and straightforward adaptation of a service during the service session. The adaptation/extension capabilities are almost unlimited.

The active service concept naturally draws on the recent programmable/active network paradigm. The main advantage of these networks is the following: their nodes are not only capable of passive data transfer, but are also enabled to perform computations on the data, or to manage these data. The transferred data are used to make special decisions. Applications and users customize the processes in the network (such processes as routing, signaling, multicast, congestion control, firewalls etc.) by providing their own methods. The usual network functionality is therefore augmented.

To extend active/programmable networks paradigm to a service level, a common support is to be chosen, e.g. a Java virtual machine or any other, preferably platform-independent, support shared both by the customizable environment (servers in the provider domain) and the customizing system. This would allow the execution of the portable code submitted by the user. This portable code can be implemented as:

- a script to be uploaded to the service component (USM in TINA) and interpreted at the appropriate moment, or

- a nomadic module (e.g. mobile agent) which will be executed by the receiving component (in TINA, operational interfaces of TSA components are to be extended as to take into account the submitted modules).

A somewhat similar concept, which surely fits the active service paradigm, is proposed in [8]: a service is customized by intelligent agents sent to the server machines where they are perceived as front-ends of plain services. Residing in the protected agent domain, they provide a customized service interface to users, by using one or more existing "base" services.

4 DISCUSSIONS

4.1 Related work

Our proposed mechanisms for service customization differ somehow from the software adaptation method described in [7] which is a universal approach. This method makes inter-object relationships more flexible through a high level programming. This kind of metaprogramming allows instantiation of the abstract software to a particular graph of associations among object classes. The suitability of the adaptive software method to service components is to be reviewed.

Aster, a system for customization of a run-time environment [12], allows to build adapted software buses automatically. This system thus may be applied as a way to adapt the underlying infrastructure, i.e. the DPE (Distributed Processing Environment) and the NCCE (Native Computing and Communication Environment) of TINA.

4.2 Open issues

As we have seen, the customization is mainly based on the user profile or another representation of user requirements. Therefore, a customization made by/for one user may affect other participants in the case of cooperative service. The possible requirements interactions (conflicts) have to be managed.

Furthermore, a security problem arises, especially for the active services. Indeed, mobile agents or script fragments may present a threat towards the provider's computing environment. Likewise, customization interface of a service group is subject to security checks. These security procedures are scheduled to be performed by ASFirewall computational object (recall Section 3.2).

4.3 Conclusion

In this paper, we analyzed the means to customize telecommunication services and developed an ascending customization approach. This approach is conveyed in different levels, each of them corresponding to a rising degree of user involvement in a service. At the first level, a set of service options is offered at the provider-client interface. The next, more elaborated stage is supported by a conjunction of configurable elementary modules designed to form a custom-made service instance (these modules are categorized hierarchically in order to satisfy their inner constraints and to obtain

a correct service combination). The second customization level brings a possibility of active user intervention: he/she is enabled to choose a needed version of a service, using customization interfaces over a service facet group. Finally, the active service paradigm is introduced and lets a user to deploy his own pieces of software as to implement customized services.

In sum, our approach is based on several integrity levels allowing to choose a desired involvement of the user in the service customization process. Our proposal is fine-grained and flexible, as opposed to the static and piecemeal TINA approach. Moreover, we specify enterprise and computational viewpoints for our customization proposal.

Work is currently under way at ENST Bretagne to implement and demonstrate the discussed customization concepts. The two identified customization levels - these of service composition and customization interfaces - will be based on a TINA service architecture prototype currently being developed over the VisiBroker CORBA platform at our site. In addition, we consider the use of new scripting language [3] intended to control CORBA objects (the service modules and the facets, in our case).

Acknowledgements

The authors would like to thank anonymous reviewers for points of clarification.
A part of the work described in this paper is partially supported by the Britanny General Council and the European Union.

References

[1] CARDELLI, L. *A Semantics of Multiple Inheritance.* Springer Verlag, 1984, pp. 51–68.

[2] CARDELLI, L., AND WEGNER, P. On Understanding Types, Data Abstraction, and Polymorphism. *ACM Computing Surveys*, Vol. 17, No. 4, Dec. 1985, 471–522.

[3] CorbaScript language. http://corbaweb.lifl.fr/CorbaScript/.

[4] GUERRAOUI, R., FELBER, P., GARBINATO, B., AND MAZOUNI, K. System Support for Object Groups. *OOPSLA'98*, Vancouver, Canada, Oct. 18-22, 1998.

[5] HILTUNEN, M. A. Configuration Management for Highly-Customizable Services. *4th Int. Conference on Configurable Distributed Systems (ICCDS'98)*, Annapolis, USA, May 4-6, 1998.

[6] KOSCIELNY, G., AND SADOU, S. Type de service générique pour la réutilisation de composants. *Langages et Modèles à Objets (LMO'99)*, Villefranche-sur-Mer, France, Jan. 27-29, 1999.

[7] LIEBERHERR, K. J. *Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns.* PWS Publishing Company, 1996.

[8] MAGEDANZ, T., ROTHERMEL, K., AND KRAUSE, S. Intelligent Agents: An Emerging Technology for Next Generation Telecommunications? *INFOCOM'96*, San Francisco, USA, Mar. 24-28, 1996.

[9] MAKNAVIČIUS, L., KOSCIELNY, G., ZNATY, S. Customizing Telecommunication Services: Patterns, Issues and Models. *6th International Conference on Intelligence in Services and Networks (IS&N'99)*, Barcelona, Spain, Apr. 27-29, 1999.

[10] PARHAR, A., AND HANDEGARD, T. TINA Object Groups: Patterns in Chaos. *TINA'96*, Heldelberg, Germany, Sep. 3-5, 1996.

[11] SADOU, S., AND KOSCIELNY, G. Competence Pool Abstraction and Dynamic Re-use. *ECOOP'97 Workshop Reader*, Jyväskylä, Finland, Jun. 1997, Lecture Notes in Computer Science, No. 1357, pp. 221–255. Workshop #6: Models, Formalisms and Methods for Distributed Object Oriented Computing.

[12] SARIDAKIS, T., BIDAN, C., AND ISSARNY, V. A programming System for the Development of TINA Services. *Joint International Conference on Open Distributed Processing and Distributed Platforms (ICODP'97)*, Toronto, Canada, May 26-30, 1997.

[13] TINA CONSORTIUM. *Overall Concepts and Principles of TINA v1.0.* Feb.1995.

[14] TINA CONSORTIUM. *Service Architecture v5.0.* Jun. 1997.

[15] TINA CONSORTIUM. *Service Component Specification v1.0, Part B.* Jan. 1998.

[16] ZAHARIADIS, T., ROSA, C., PELLEGRINATO, M., LUND, A. B., AND STASSINOPOULOS, G. Interactive Multimedia Services to Residential Users. *IEEE Communications Magazine*, Vol. 35, No. 6, Jun. 1997.

Biographies

Linas Maknavičius holds a BSc degree from Vytautas Magnus University, Lithuania, a French MSc degree from University of Rennes 1, and is preparing his PhD at ENST Bretagne (Graduate School of Telecommunications Engineering), Rennes, France, in the telecommunication services management field.

Gautier Koscielny is working toward the PhD degree at University of Bretagne-Sud, Vannes, France. His research interests include object-oriented programming, software architectures and distributed systems. His thesis topic is coordination patterns for distributed application.

Simon Znaty is a Professor, Hab. Dr. in the Networks and Multimedia Services departement of ENST Bretagne, Rennes, France. He obtained a PhD degree from ENST Paris, France, in 1993. During 1993-94, he worked with the Network Architecture Laboratory at NTT, Japan, and during 1994-96, he was a senior researcher with the Telecommunications Laboratory at EPFL, Lausanne, Switzerland. His current research interests span telecommunication services engineering, intelligent networks, mobile networking, services management, active networks and services, and distributed computing.