Towards Unanticipated Dynamic Service Adaptation

Marcel Cremene[1], Michel Riveill[2], and Christian Martel[3]

[1] Technical University of Cluj-Napoca, Romania, cremene.marcel@com.utcluj.ro
[2] University of Nice, France,riveill@unice.fr
[3] University of Savoie, France, christian.martel@univ-savoie.fr

Abstract. Most service adaptation solutions follow an anticipated approach: the adaptation control is based on predefined service-specific rules and strategies. These solutions will not work correctly in a context that was not taken into account by predefined rules and strategies even if a large context diversity was considered. This paper presents a solution based on a context-service common representation that enables us to discover the adaptation rules and strategies rather than to fixe them a priori.

1 Introduction

Component based software represents an important trend lately. A service is provided by a software component architecture.

Service dynamic adaptation is necessary because the context (user profile, physical resources and other elements) may change while the service is running. We have chosen a forum service example in order to get a better image of this problem. Let us suppose that the forum service was initially designed and built for a specific context: the users speak the same language (English) and are able to use a graphic interface (standard users), the terminal is a desktop PC (14-inch screen at least), a pre-installed web browser and a stable Internet connection are also available.

If the context does not fit in the initial hypotheses, the forum service will not work correctly or it will become completely unusable. The following contexts are such examples and assume dynamic changes: the user may have difficulties to write messages in English and for the long phrases he may prefer to use his native language, the user may be unable to watch the screen all the time because his view might be busy sometimes, the terminal may be changed while using the service, the user is traveling and his geographical/social changes as his interest too.

The current service adaptation approaches require the prediction of all these contexts and specify rules for each possible situation, and the service cannot work outside these predictions. We call these approaches "anticipated".

Current approaches and their limitations. After analyzing several adaptation platforms such as: "Rainbow" [GCH+04], "MobiPADS" [CC03], "Gaia" [Rom03], "Odyssey" [NSN+97], "Xmiddle" [CEM01], "Molene" [SA00], "NAC" [Lem04], "DACIA" [LP00], "CESURE" [MR00], "Chisel" [JV03], "K-component" [DC01] and others the conclusions were:

- Existent adaptive architectures are in general specialized but all present three main parts: a)the adapted system including reconfigurable elements, b)an observation part that monitors the context and the service state and c)control part based on rules and strategies. This architecture model is depicted in figure 1; the unanticipated approach requires a service-context meta-description and general rules and strategies.
- Existent adaptation control part follows an *anticipated* approach: the adaptation rules are service specific and the adaptation strategies are predetermined; adding new rules and strategies demands user intervention.
- We have found service models and context models [W3C], [RGL04], [Bre03] but not a unified service-context model that shows how the service and the context interact.
- We have found numerous and diverse adaptation techniques making possible to reconfigure dynamically the service architecture but these do not affect the unanticipated character which is determined mainly by the adaptation control part.

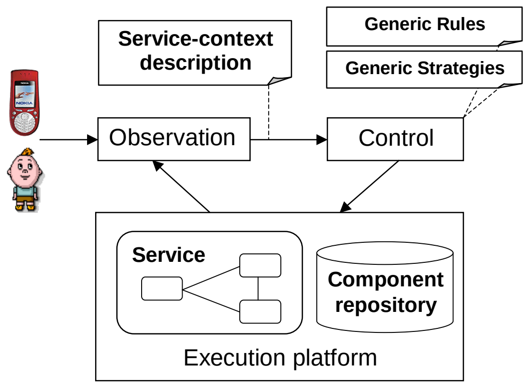


Fig. 1. Proposed architecture for unanticipated adaptation

2 Unanticipated service adaptation.

Our objective is to find a solution for unanticipated and dynamic service adaptation. The solution must enable the service to evolve as the context evolve, without stopping the service and without adding manually adaptation rules and strategies. The figure 2 depicts a comparison between the anticipated and the unanticipated approaches.

Anticipated adaptation (classical approach)	**Unanticipated adaptation (our objective!)**
Adaptation ***strategies*** are predetermined	The best ***strategy*** is searched
Adaptation ***rules*** are service specific	Adaptation ***rules*** are general and service independent
Service ***architecture*** built to deal with any possible and anticipated context	Service ***architecture*** evolve only if the context change
Context evolution requires an architecture redeployment	***Context evolution*** is possible while the service is running
The service ***developer*** role is complex: he needs to anticipate and specify rules and strategies for each adaptive service	The service ***developer*** role simplified: he just assembles the service; any service is adaptive and may evolve

Fig. 2. Comparison between the anticipated and the unanticipated approaches

A closer proposal to the unanticipated approach is, for instance, the semantic component model "CoSMoS" and the service composition platform "SeGSeC" proposed in the paper [FS04a,FS04b] by K. Fujii and T. Suda. The platform takes as input a natural language service request, selects the components based on the words semantic and connect them according to the phrase structure. Unfortunately this proposition does not take into account a dynamic context.

The unanticipated adaptation requires to solve two technical issues:

A. Propose a "semantic" knowledge representation about the service and its context which describes how the service work in a specific context.
B. Propose algorithms for analyzing the previous knowledge representation in order to discover the adaptation problems and find solutions.

3 Service-context knowledge representation

The first requirement for unanticipated adaptation is to have a service-context description that "explains" how the service and the context interact. In order to make this possible we introduce the *profile* concept. The profile most important role is to describe how the different entities, either logical: the components or physical: the context elements such as users, terminals, networks, physical places interact each other. For instance, if a user interacts with a service it would be normal to use a same language and information type (visual, voice).

In the figure 3 we present a three layers model that describes a forum service functioning in a context composed by a several users and a terminal. The model is described from the point of view of a certain user.

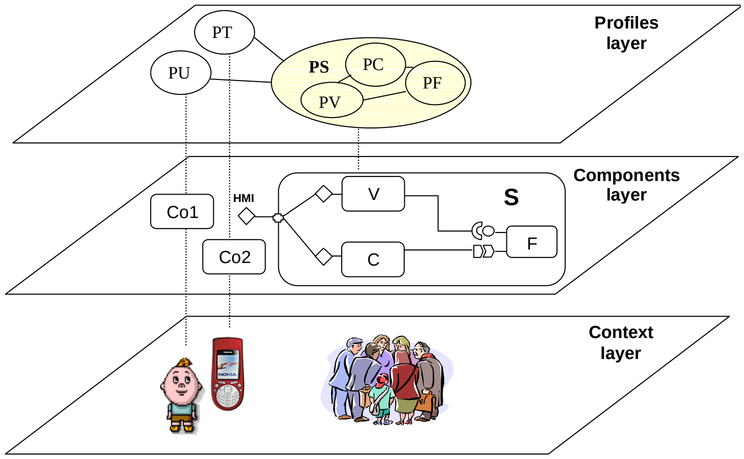


Fig. 3. Service and context description - the three layers model

Context layer. The context layer contains elements such as: users, terminals, networks, environment and others. In order to simplify the service-context unification, we extend the component concept also for the context elements, for instance a user is a component providing HMI.

Components layer. The component's layer contains the services/components described using existent models: CCM (CORBA Component Model) [Gro99] and FractalADL [BCS02]. The composition is recursive (Fractal model [BCS02]). The forum service S is composed by three components: C is composer or message editor, V is message viewer and F is forum server. The components Co1, Co2 are observer components or context detectors. The service HMI (Humain Machine Interface) is composed by the C HMI and the V HMI.

Profiles layer. The profiles layer unifies service context models. The profile model was determined starting from the interactions that may exist between entities (components and context elements). Two aspects were revealed being the most relevant: the *resources* (memory, screen surface, network connections) and the *information* that is exchanged between entities. The profile model is depicted in the figure 4, it's elements are:

- *Global attributes*, i.e. memory, OS type,
- *Flows* of information between an input port and an output port of the component. A flow example is a message created at the HMI port and as event.
- *Flow attributes* associated with the information, i.e. message language, stream encoding, file type, etc.

Each profile attribute have: a name, a definition domain, a composition operator and a validation operator. For instance, we may have: attribute name "langue", domain {FR, ED, DE, , *, ?}, composition attribute ":=" (because

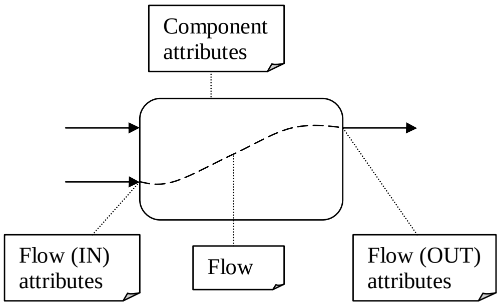


Fig. 4. Profile model

the value is transferred from one component to another), comparison attribute ”=” (because two entities are compatible if they use a same language).

An automatic profile composition is necessary in order to create the service-context description. A component ’C’ composed by a component ’A’ and a component ’B’ will have a profile given by the profiles of A and B. The figure 5 depicts the profile attribute composition that is done attribute by attribute.

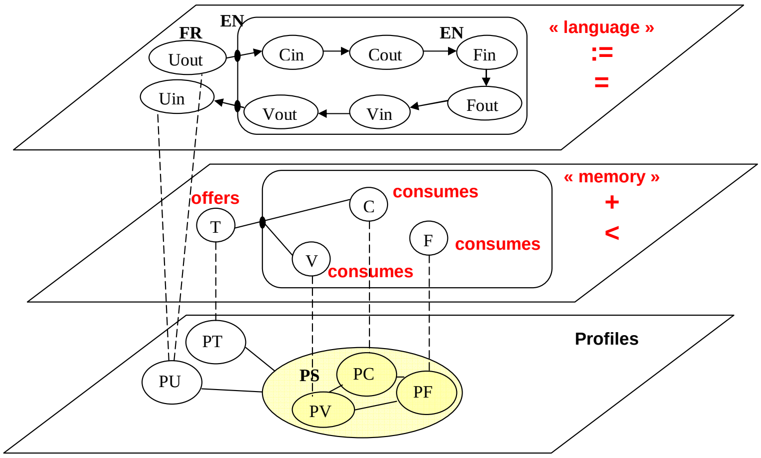


Fig. 5. Profiles composition

The ’memory’ is a global attribute. We use a graph that has as nodes the entities (components, context elements) related to memory. The arcs correspond to the relation existent between the entities from the memory point of view: components placed on a machine will consume machine memory. In our case the components C and V are installed on the client and F is installed on the server machine. The graph has a recursive structure: a node may be expanded because an entity described by a profile may be composed by other entities.

The 'language' is a flow attribute. We use another graph where the nodes are associated with the component's ports and the arcs indicate the information flows. If the profile does not change the attribute value, we keep and propagate the initial value through the graph arcs.

4 Adaptation analyze and solution search algorithms

The second issue of unanticipated adaptation is to use the service-context knowledge representation in order to discover the adaptation problems and find solutions. The adaptation control supposes the following three steps: a) profile validation, b) strategy selection, c) strategy application.

Dysadaptation problems that may exist between the service and the context. They are verified using the validation operators described in figure 5. Our example uses '=' for the attribute 'language' and '<' for the attribute 'memory'. If the service consumes MS bytes and the terminal offers an MT bytes space then the composition is valid only if $MS < MT$. The validation procedure is applied attribute by attribute. We build the attribute graph and, for each arc in the graph that connects a context node with a service node, we apply the validation operators. A problem list is created finally, each problem being specified by: an attribute name, attribute values, validation operator.

The strategy selection takes into account a finite set of service-independent strategies that imply the following operations over the components: parametrization, insertion, replacement, elimination and migration. The automatical selection between strategies is not yet solved in our proposition, a solution may be to check all the possibilities and to select the less expensive (time, resources).

We have tested the insertion and the replacement strategy. The insertion requires to solve two problems: a)decide what component needs to be inserted and b)determine the insertion point. In order to solve the problem a) we use the problem list resulted from the validation procedure and, for each problem we search all the components that may solve that problem. If we have a problem defined by: attribute 'language', context value 'FR', service value 'EN', operator '=' we need to add an information treatment that transforms an 'FR' input to an 'EN' output. The insertion point is determined using a search algorithm that analyzes the attribute graph, figure 6.

The algorithm tries to insert the solution component by checking the component interfaces compatibility. In this case, if it is no possible to insert the component between the first two components (the user HMI and the service HMI) the algorithm goes deeply into the service structure and tries to insert the solution component between other existent components, following the graph branches.

The control part, figure 7, uses a service-independent table containing the attributes definitions and operators for composition and validation.

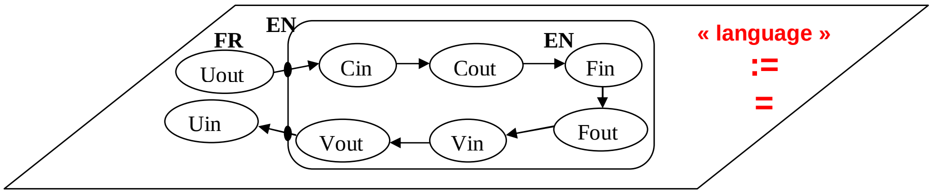


Fig. 6. Adaptation validation and resolution for a "language" related problem

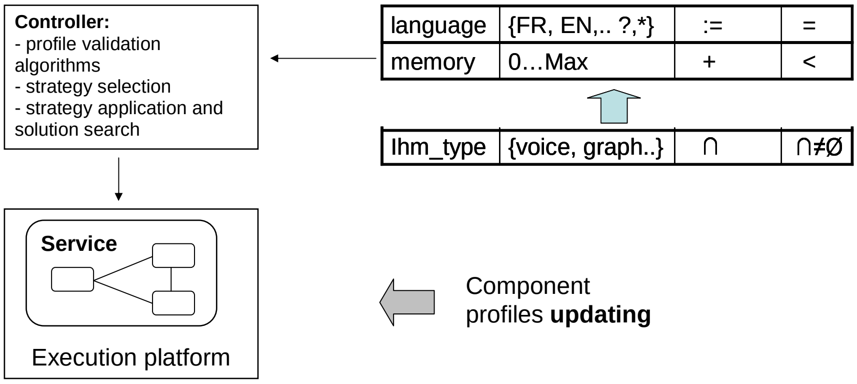


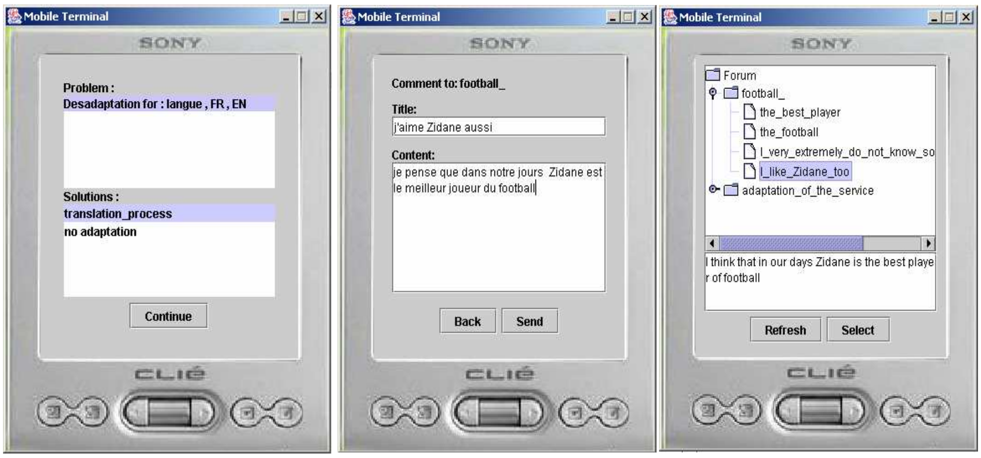
Fig. 7. Functioning and extensibility

Each component must have a profile. The component profiles must be specified by the component developers and the adaptation depends on the profiles content: if an attribute is not present in the profile we cannot take this attribute in account for the adaptation process. The extension requires to add new lines in the attribute table and to update the existent component profiles that is still a drawback of our approach because both operations requires intervention of a human operator.

5 Prototype

The figure 8 depicts the forum GUI. The user log in, the platform detects a conflict between the user profile (previously stored) an the service profile because the user language and the service language are different.

The platform proposes to user two possibilities: use a translator or leave the service unchanged. Supposing the user chose to use the translator from French to English, his messages are translated. The prototype has two versions: in the first one the user language is supposed to be stored in a database, in the second the language is detected at each message.



a) b) c)

Fig. 8. Forum UI

6 Conclusions and perspectives

In this paper we have proposed a solution for *unanticipated* and dynamic service adaptation. The majority of the existent adaptation platforms use an anticipated approach: the adaptation rules and strategies are service specific and specified a priori by a human expert. This fact is a service autonomy limitation.

In our proposal the service and the context are described using a unified model that allows the machine to reason about their interactions and possible dysadaptation. We propose: a profile model describing components and context elements behavior, composition operators that make possible to compute automatically the profiles of a composite component and validation operators that allow us to detect the dysadaptation problems. Based on detected problems description we search for external components as possible solution. Until now we have tested only the component insertion. A drawback is that the profiles updating still need human operator intervention.

The proposed solution is an alternative to the service specific adaptation control used in the majority of the existent proposals and use service independent rules and strategies.

In perspective we intend to focus on the following problems: strategy selection, algorithms improvement, test more complex examples, develop the profiles algebra, introduce AI tools such as logical systems, inference engines, feedback based learning and others.

References

[BCS02] E. Bruneton, T. Coupaye, and J. B Stefani. Recursive and dynamic software composition with sharing. In *ECOOP Workshop on Component-Oriented*

Programming, pages ??–??, Malaga, Spain, 2002.

[Bre03] P. Brezillon. Context-based modeling of operators practices by contextual graphs. In *14th Mini Euro Conference: Human Centered Processes*, pages 129–137, May 2003.

[CC03] Alvin T. S. Chan and Siu Nam Chuang. Mobipads: A reflective middleware for context-aware mobile computing. *IEEE Trans. Software Eng.*, 29(12):1072–1085, 2003.

[CEM01] Licia Capra, Wolfgang Emmerich, and Cecilia Mascolo. Reflective middleware solutions for context-aware applications. In *Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns*, pages 126–133, Springer-Verlag London, UK, 2001.

[DC01] J. Dowling and V. Cahill. The k-component architecture meta-model for self-adaptive software. Technical report, Trinity College Dublin, Computer Science Department, Dublin, Ireland, 2001.

[FS04a] Keita Fujii and Tatsuya Suda. Component service model with semantics (cosmos): A new component model for dynamic service composition. In *International Symposium on Applications and the Internet Workshops (SAINTW'04)*, pages 348–355, Tokyo, Japan, 2004.

[FS04b] Keita Fujii and Tatsuya Suda. Dynamic service composition using semantic information. In *2nd International Conference on Service Oriented Computing (ICSOC 04)*, pages ??–??, New York City, NY, USA, 2004.

[GCH+04] David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley R. Schmerl, and Peter Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *IEEE Computer*, 37(10):46–54, 2004.

[Gro99] Object Management Group. Corba components : Joint revised submission. Technical report, Sun Microsystems Inc. 2550 Garcia Avenue, Mountain View, CA 94043, http://java.sun.com/beans, August 1999.

[JV03] Keeney John and Cahill Vinny. Chisel: A policy-driven, context-aware, dynamic adaptation framework. In *Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 3–13, Lake Como, Italy, June 2003.

[Lem04] Tayeb Lemlouma. *Architecture de Ngociation et d'Adaptation de Services Multimedia dans des Environnements Heterogenes*. PhD thesis, L'Institut National Polytechnique, 2004.

[LP00] Radu Litiu and Atul Prakash. Challenges in using a mobile component framework to develop adaptive groupware applications. In *Proceedings of CBG2000, the CSCW2000 workshop on Component-Based Groupware*, Philadelphia, Pennsylvania, USA, December 2000.

[MR00] Philippe Merle Michel Riveill. La programmation par composants. *Techniques de l'Ingnieur - Informatique, H2759*, December 2000.

[NSN+97] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile application-aware adaptation for mobility. In *Sixteen ACM Symposium on Operating Systems Principles*, pages 276–287, Saint Malo, France, 1997.

[RGL04] Gustavo Rossi, Silvia Gordillo, and Robert Laurini. Gnration de services dpendant du contexte pour des applications mobiles. In *Proceedings of Mobilit et Ubiquit 2004, Premires Journes Francophones*, pages 3–13, Nice, Sophia-Antipolis, Essi (Ecole Suprieure en Sciences Informatiques), June 2004.

[Rom03] Manuel Roman. *An Application Framework for Active Space Applications*. PhD thesis, University of Illinois at Urbana-Champaign, 2003.

[SA00] M.T. Segara and F. Andr. A framework for dynamic adaptation in wireless environments. In *Proceeding of TOOLS Europe 2000*, Mont St. Michel, St. Malo, France, June 2000.

[W3C] W3C. Recommendation 15 january 2004, composite capability/preference profiles (cc/pp): Structure and vocabularies 1.0. Technical report, WWW Consortium.