Journal of
Information Technology & Software Engineering

Research Article Open Access

An Architectural Approach for the Dynamic Adaptation of Services-based Software

Baroudi Mohammed Yassine*, Benammar Abdelkrim and Bendimerad Fethi Tarik
Department of Electrical Engineering and Electronics, University Aboubakr BELKAID Tlemcen, Algeria

Abstract

Services are very important component of software which will be essential part for future internet applications. Development of several applications considering the need of open environment and large scale usage is the need of the hours. Alternative intelligent presence and absence of services along with maintaining the quality is important. Dynamic adaptations and their optimum efficiency are mandatory to have a better application and solution. Moreover, novel application development requires other factors to be considered, such as, cost effectiveness and reusability of the existing components in a better and effective manner. This article proposes specific software architecture for dynamical service adaptation. The services are constituted by reusable software components. The adaptation's goal is to optimize the service function of their execution context. For a first step, the context will take into account just the user needs but other elements will be added. A particular feature in our proposition is the profiles that are used not only to describe the context's elements but also the components itself. An Adapter analyzes the compatibility between all these profiles and detects the points where the profiles are not compatibles. The same Adapter search and apply the possible adaptation solutions: component customization, insertion, extraction or replacement.

Keywords: Adaptive service; Software component; Service; Dynamic adaptation

Introduction

The operation of a service application must consider different elements that interact with its operation which are provided as following:

A service application is generally constructed by assembling a variety of software components. These pre-existing software components are being reused to build the application. Productions are continued by different manufacturers. Such components are adapted depending on the specific context of their use. In the present days, the Internet has become a free and dynamic market of reusable components.

During the daily usage of these applications, they use different types of terminals. They appear continuously in the market and it is impossible to predict at prior all target platforms, especially, in the field of mobile terminals. We assume that, all such systems have a computing capacity and sufficient RAM, and various means of connection (GSM, GPRS, LAN or Bluetooth).

User needs are diverse and evolving continuously. Some of these depend on the physical context, i.e., location, external noise etc. or the social context, such as, human issues. For example, it may be necessary to complete a service to suit the specific requirements of the user. A visually impaired person in need of a suitable IHM child requires simple vocabulary French speakers quite literate than French.

We grouped all of these items listed in the notion of context of the service. This context is variable and constantly influences the services from different point of views. The context adaptation service is an important problem whose expected solution is variable with time. Considering a service application is built by assembling components, we believe that the adaptation of the latter should be carried out at its architecture level either by adding, removing, or replacement of its components.

Relevant Literature Evidences

It has been observed that several researches have been carried out dealing with dynamic adaptation problems in the past, hence, allowed the emergence of several approaches. In the model driven approach, the dynamic adaptation is based on a component model that is designed to support this kind of adaptation.

Dynamic Component Updating (DCUP) [1] is an example of such approach. In DCUP, the component is divided into two parts: permanent part and replaceable part. Adapting a component refers to the alteration of its replaceable part by a new version at run-time.

In the reflexive approach, an application contains an abstract level (meta-level) that reify the real system. Under such circumstances, the adaptation is made first on the meta-level, after that, the changes are reflected on the executed applications. Thanks to the causal connection between the meta-level and the real system. An example of this system is DYVA [2]: a reflexive framework for dynamic reconfiguration of components-based applications. The framework is decomposed in two main parts: The base-level represents the concrete application that provides the expected functionalities and its execution environment and the reconfiguration of the machine that contains two major characteristics. The different operational modules responsible for achieving the reconfiguration in a major characteristic and the other issue is the meta-level which represents the reification of the concrete application.

Architectures for Adaptation Services

Many systems that support services associated with dynamic adaptation usually consist of three parts:

1. The formed part of the service that can be adapted dynamically according to wide variety of techniques [3].

***Corresponding author:** Baroudi Mohammed Yassine, Department of Electrical Engineering and Electronics, University Aboubakr BELKAID Tlemcen, Algeria, Tel: 213(43)202000; E-mail: yassinebaroudi@hotmail.com

Received July 19, 2015; **Accepted** September 11, 2015; **Published** September 28, 2015

Citation: Yassine BM, Abdelkrim B, Tarik BF (2015) An Architectural Approach for the Dynamic Adaptation of Services-based Software. J Inform Tech Softw Eng 5: 156. doi:10.4172/2165-7866.1000156

2. The party responsible for evaluating continuously the group consisting of the service and its context and performs a monitoring task.

3. The control part generating a logic that is specific to the user's/developer's reconfiguration commands.

These three parts are physically distinct. In some proposals, all the three parts are present in each component. For example, the model proposed by Segara and André [4] allows each component to be adapted, to possess a part related to the observation and possibly generate reconfiguration commands.

Aksit and Choukair [5] proposed an overview of the dynamic reconfiguration and adaptation techniques. Some techniques for dynamic adaptation are as follows: insertable components, alternative algorithms, oriented programming aspects-AOP, composition filters, connectors changeable, interaction patterns as described by Blay-Fornarino et al. [6], reflexive middleware [7], behavior injectors, and adaptive interface.

Objective

In this article, our goal is to propose a software architecture that enables the dynamic adaptation of services built by assembling components depending on a variety of usage contexts. As part of this first experiment, the context relates to the user's needs.

The main feature of our proposal is the behavior of each service in relation to its context of use which is evaluated by the analysis of the behavior of each component constituting the service.

For this, we used profiles that describe not only the elements of context but also each component constituting the service. An adapter analyzes the consistency of different profiles for each component and the contextual elements of the profiles.

The adapter detects the mismatch points through seeking and applying various components of the service changes which are required to restore. Based on such compatibility issues and by changing configuration settings through adding, removing, or replacing components, adapters aid in restoring the service.

Architecture for Dynamic Adaptation of Service

The proposed architecture is consists of three parts which are described in the following section:

- Modifiable part: This part consists of the service containing an assembly of components. The modifiable elements are the components, and the various interconnections between these components.

- Monitoring part: This part is represented by monitors who observe the resources and user profiles. These elements provide the necessary data for a complete description of the service called meta-description of all petrol contexts.

- Control part: This part is represented by the Adapter which from the description of all petrol contexts decides the necessary modification to accommodate the service. In this context, an assembler dictates adapter. The adapter uses existing components found within the basic components, such as, adaptation options (Figure 1).

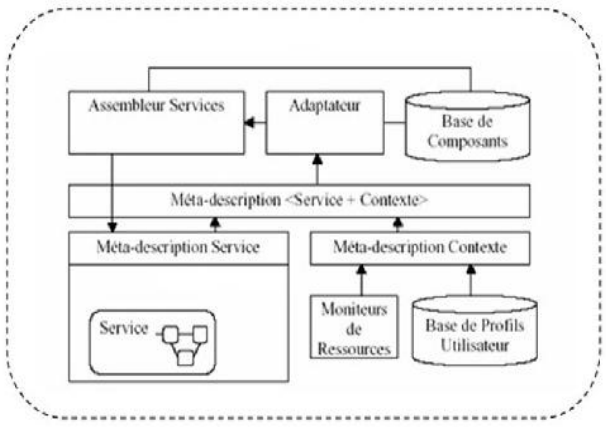

Figure 1: Architecture for dynamic service adaptation.

Illustrative example

A bulletin board service allows a community of students to exchange information on scientific and cultural activities within the university. For customary reasons, the language focus is French. Student members have the opportunity to write and read on the forum.

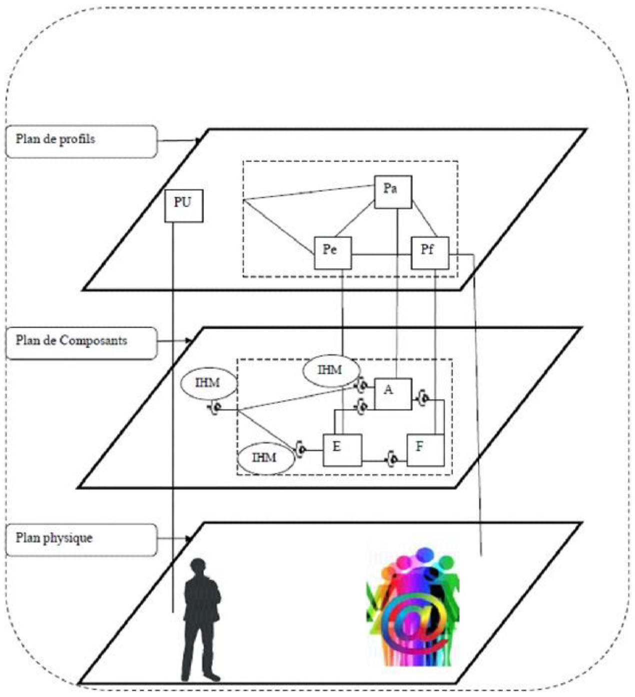

Figure 2: The meta-description of all Service-context "different planes of view.

Meta-description of all service-context

Figure 2 shows a perspective on the meta-description of all corresponding context-service scenario presented in the illustrative example.

We find three different levels in this:

- The user plane containing the physical elements of context.

- The plan components containing the architecture description language (ADL) service and

- Interface definition language (IDL) components.
- The ADLs describe software architectures. Rewriting the rules capture and consolidate adaptations as sets of rules (patterns) and facilitate their selection.
- The profile plan that meets the service profile and the context profile indicates how the service is.

User plane – physical elements of the context: Physical layer contains the physical elements of the context. In this case, they are the users of the forum service. The items found in this plan have projections in the profile plan. These projections form the meta-description of the context.

The plan components – service components assembly: The plan contains component's "S" service as assembly of software components that are the basis of bundles of OSGI platform. For our example it consists of:

- Component A: Display messages posted on the forum,
- Component E: Writing new message
- Component F: Forum contains all the messages posted on the forum.

The IHM service "S" is constructed from a composition IHM components E and A. This plan components represent the syntactic part of the meta-description of the service. To describe the service in this plan we use ADL type of language that describes the internal architecture of the service, IDL descriptors and interconnections that are MANIFEST. MF files for the components. We have also included the IHM interconnections (or connectors).

The profiles plan – user profile, profile components, composition profiles: The profiles of the plan are essential for adaptation because it represents the semantics of the meta-description of the set service-context. For the example chosen it contains:

The profile of the PS service, the user profile Pu. PS is the result of the composition profiles of the components that make up the service "S"Pe, Pa, Pf. This plan contains the semantics of all service context.

The user profile contains, in our example, a single parameter: language = 'Ar', indicating a user writing in English. The profile of a component indicates how the component works with respect to the parameters of context. The profile of a component translation for Arabic to French:

<profile>

<component>Translation AR_FR</component><point>

<interface>Translation</interface>

<method>translate</method>

<argument>text</argument>

<argtype>String</argtype><precondition>**langue** = 'AR' </precondition>

</point>

<point>

<interface>Translation</interface>

<method>translate</method>

<returntype>String</returntype>

<function>=</function><postcondition>**langue** = 'FR' </postcondition>

</point>

</profile>

This profile shows the behavior of the component with respect to the language, the interface to the "Translation" component "TranslationAR_FR" remains as a precondition to the parameter "language" which calls for the "AR" value.

The return value will be "FR" value. This component, therefore, modifies the language and thanks to the profile Adapter which can discover this fact.

The profile of the service which is a composite component resulting from the composition profiles of components I, Q, F: F requires language = 'FR' by convention."F" is connected with "E", but "E" expresses language neutral point of view via the assigned condition extended to "E". The IHM "E" is connected to the IHM "S" and the condition extended to the IHM "S" (Figure 2).

Adaptation of axioms all service-context: For the semantic part of the set service-context, one that corresponds to the profile level, we need to define the axioms (conditions) which are checked if a service is appropriate for its context. As an example, one axiom we need to have to represent the adaptation of all petrol context if the profiles parameter values are contradictory. In our example, if the user writes in Arabic, there is a contradiction for the parameter "language" (Ar<>Fr) at the IHM Service "S". In this case, the adaptation axiom is not verified.

The adapter-algorithm

The adapter must be able to adapt the service to the condition provided, it must perform the following:

- Verification: For each parameter profile adaptation axioms should get satisfied, especially, when we all followed a suitable service.
- Search an adaptation solution: If at least one parameter is in use, an axiom does not provide proper output, thus it must be adapted to search.
- For each parameter that does not satisfy the axioms, the adapter builds a graph that has nodes of the component interfaces which have a relationship with this parameter and referring the arcs as the interconnections.
- In this graph the adapter associated branches (sequence of nodes) which are of unequal value are represented for each branch adapter looking components, inserting these into the branch, restores equal values.
- Application of the solution: If the adapter able to find the solution, it can apply the solution found.

Prototype

We propose in the prototype (Figure 3), the service consists of the components which contain the following.

A-View forum

E - Write a message

Proxy F - a local connection to the remote component

F - Forum.

T component - Translation is dynamically added to the request of

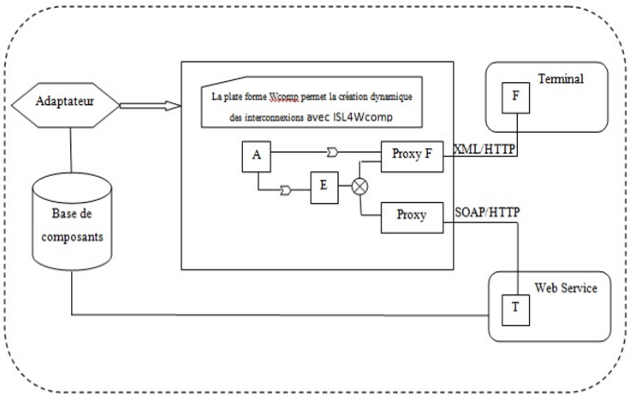
Adaptateur
La plate forme Wcomp permet la création dynamique des interconnexions avec ISL4Wcomp
Terminal
F
A
Proxy F
XML/HTTP
E
Proxy
SOAP/HTTP
Base de composants
Web Service
T

Figure 3: Diagram of the prototype.

the adapter which uses the following:

- The platform Wcomp [8]: This platform provides a development environment based on software components, it allows the dynamic creation of interconnections with ISL4Wcomp. The dynamic adaptation of Wcomp is done by adding, removing, the connection and disconnection of software components during the execution of the application.

- The language of ISL4Wcomp (Interaction Specification Language for Wcomp) interactions [9]: It is based on the interaction specification language called ISL (Interaction Specification Language), which we used to describe patterns of interactions objects [10]. ISL4Wcomp meanwhile fits these specifications for the consideration of interactions based on messages or events in the components in the assembly.

The adapter is the component in the components directory that is searched from the profile of the component by typing connections.

Conclusion

In this paper we presented an architecture that allows dynamic service adaptation. Our proposal is based on a meta-description of the overall context where in the service we have a limited set of axioms adaptation based on the semantics of the service. As it is not mandatory to describe the different rules of evolution which will be discovered by analyzing case of inadequacy subject only considering each component described with its profile.

To demonstrate the architecture, we implemented a prototype board service. This service is initially created for French-speaking users, but the proposed architecture can adapt other service by adding a dynamic translation component if the user language is not French. The main disadvantage of this architecture is its complexity. Generalization and further simplification of the model is required so that different profile settings could be assessed for the expression of adaptation axioms.

Despite these limitations and difficulties, we believe that the future belongs to adapting semantic service composition. To achieve this, the functioning of the whole context Service must be understandable for the machine, not just the human being who is building the services.

References

1. Plasil F, Balek D, Janecek R (1997) DCUP: Dynamic Component Updating in Java/CORBA Environment.
2. Ketfi A (2004) UneApprocheGénérique Pour La Reconfiguration Dynamique Des Applications A Base De ComposantsLogiciels. Thèse de doctorat de l'Université. Joseph Fourier de Grenoble.
3. Ledoux T (2001) Projet RNTL ARCAD D.1.1 Etat de l'artsurl'adaptabilité. Ecole de Mines de Nantes, 4, rue Alfred Kastler, 44307 Nantes Cedex.
4. Segara MT, André F (2000) A Framework for Dynamic Adaptation in Wireless Environments. IRISA Research Insitute, Technology of Object Oriented Languages and systems (TOOLS 33), St. Malo, France.
5. Aksit M, Choukair Z (2003) Dynamic, Adaptive and Reconfigurable Systems Overview and Prospective Vision. ICDCSW'03, Providence, Rhode Island, USA.
6. Blay-Fornarino M, Ensellem D, Occello A, Pinna-Dery A-M, Riveill M, et al. (2002) Un service d'interactions : principes et implémentation . Journées composants 1: 16.
7. Kon F, Costa F, Blair GS, Campbell R (2002) The Case for Reflective Middleware: Building middleware that is flexible, reconfigurable, and yet simple to use. ACM Comminications 45.
8. Cheung-Foo-Wo D, Blay-Fornarino M, Tigli J-Y, Lavirotte S, Riveill M (2006) Adaptation dynamiqued'assemblage de dispositifs par des modèles. 2ème journéesurl'ingénieriedirigée par les modèles (IDM).
9. Blay-Fornarino M, Charfi A, Emsellem D, Pinna-Dery A-M, Riveill M (2004) Software interactions. Journal of Object Technology 3 : 161-180.
10. Berger L (2001) Mise en Oeuvre des Interactions en EnvironnementsDistribués, Compilés et FortementTypés: le Modèle MICADO. Thèse de doctorat, Université de Nice-SophiaAntipolis - Faculté des sciences et techniques, Ecoledoctorale.