Dynamic Service Adaptation Architecture

Mohammed Yassine BAROUDI
University of Tlemcen
Tlemcen, Algeria

Abdelkrim BENAMAR
University of Tlemcen
Tlemcen, Algeria

Fethi Tarik BENDIMERAD
University of Tlemcen
Tlemcen, Algeria

***Abstract*—This paper proposes a 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—Adaptative service; software component; service; dynamic adaptation

I. Introduction

Software is executed in complex, heterogeneous, and highly interwoven computer infrastructures, in which a diversity of events may take place. Such events may be, for example, security threats, network problems, reduced performance in one of the servers, etc. In such situations, it is preferable to adapt the software to continue to provide the required functionality. The software adaptation can be considered as the possibility for man to reconfigure the software and then restart it; it may also be seen as the software's ability to reconfigure during execution [1]. The first case of adaptation is considered as static, while the second one is dynamic. It is possible to perform static adaptations or adjustments in cases where the system can be stopped to make the required manual adjustments. However, there are critical systems that cannot be stopped to implement the modifications, e.g. software that run power grids, and software used in the World Bank. In such situations, the software needs to dynamically adapt its behavior during execution in response to the changing conditions within its computer infrastructure support ([2], [3], [4]).

Such software programs are increasingly based on a service-oriented architecture and are implemented in dynamic and distributed large scale environments.

A service-based application should take into account different elements that interact with its functioning. It is well accepted that:

- Service-based applications are characterized by the use of heterogeneous and distributed components provided by third parties, a component is a binary composition unit with specified contractually interfaces and explicit context dependencies; it can be deployed independently and is subject to composition by third parties.
- The miniaturization of the devices and their mobility make them personal objects in their own right. Users then show a particular interest in being able to access a multitude of services and features from anywhere and any device.
- Users are also increasingly sensitive to the customization of the applications they use on their devices, such as adapting the GUI to the capabilities of the device, adapting the application to suit their preferences such as language, or adaptation depending on their movements or physical environment (brightness, temperature, etc.).

All these enumerated items are grouped in the concept of context of service. This context is variable; Dynamic availability of devices, network connectivity, location, and user preferences may change unpredictably during application execution. The adaptation of services to the context is an important problem whose solution varies over time.

Consider that a service application is built by assembling components. It is believed that the adaptation of this application is performed in its architecture by adding / removing / replacing its components.

This article is organized as follows: section 2 consists of a study of the existing domain and presents some models of components and some service architectures. After studying the weaknesses found in previous solutions, an illustrative example is used to present our proposal, in sections 3 and 4. The prototype implemented in this study, in order to validate the architecture, is described in section 5. Finally, the results obtained are exposed in the conclusion.

II. Study of the Existing Situation

Several studies, addressing the dynamic adaptation, are founded on component-based models. Recently, a component model has emerged in the industrial world, namely the Web service.

Basically, a Web service is a special software component that is searched, linked, and executed at runtime. It enables systems to interact through standard Internet protocols [5]. In order to reach their full potential, Web services can be combined to achieve specific functionalities. If the implementation of a Web service for business logic implies the use of other Web services, then this is called a composite service. In the case of web services, the assembly has a temporal dimension and is configured as a workflow assembly.

Research works ([6], [7], [8], [9]) targeted the evaluation of quality attributes of service orchestrations based on

aggregation rules. These works share the same principle. The basic idea is to define the rules of aggregation for each workflow pattern and for each quality attribute. In most cases, the rules are defined for a couple of workflow patterns, often denoted as "composition patterns" ([7], [8]), except for "sequence" and "loop" patterns, which are considered individually.

Related works on the dynamic adaptation of service compositions can be classified into three groups. The first group supports the dynamic adaptation at the language level ([10], [11], [12], [13], [14]). This approach may harm the reasoning of adaptations with complex scripts and may be prone to error ([15]). The second group focuses on the low-level implementation mechanisms for self-adaptation ([15], [16], [17], [18]). This approach is not widely supported for the analysis of the variability inherent in the dynamic adaptation at the time of conception, and can impair the reasoning about adaptations with complex and error-prone scripts.

The third group carries on by using the reconfiguration transition system (RTS). Authors in [19] defined a transition graph model at the time of conception. The nodes of the graph represent the configurations of the system, and the arcs (edges) represent the reconfiguration transitions, which are the required operations to adapt a system from one configuration to another. However, the number of valid configurations may grow exponentially for a large number of variability points of the system due to the combination of characteristics. Therefore, the construction of such a diagram model may be impossible in a large number of areas.

III. Objective

The present study aims to provide a software architecture that enables the dynamic adaptation of services which are built by assembling components, depending on different use contexts.

In more detail:

The Evaluation of the behavior of each service in relation to its context of use, is done by analyzing the behavior of each component that constitutes the service.

Use a profile for each constituent of the service, which describes the component itself and the set of context elements.

An adapter analyzes the conformance between the different profiles of each component and the profiles of the context elements.

The adapter must then be able to identify the different incompatibilities, apply necessary modifications, with add, remove or replace one or more service components to remedy the problem of maladaptation.

As part of this first experiment, the context concerns the needs of users. The main characteristic of our proposal is that the behavior of each service with respect to its context of use is evaluated from the analysis of the behavior of each component making up the service. For this reason, profiles that describe not only the elements of context but also each component constituting the service are used. An Adapter analyzes the conformity between the different profiles of each component and the profiles of context elements. The adapter detects the mismatch points, then seeks the required changes and applies them to the different components of the service to restore this compatibility, by changing the configuration parameters, by adding, removing, or replacing components.

IV. Architecture for Dynamic Adaptation of Service

The proposed architecture consists of three parts, as illustrated in Figure 1:

- The modifiable part. This portion is composed mainly by the service, which consists of an assembly of components. The modifiable items include the components and the various interconnections between them.
- The monitoring part. This portion is represented by monitors which observe the resources and user profiles. These elements provide the necessary data for a complete description of the service, called meta-description of the service-context.
- The control part. This section is represented by an adapter which, from the description of the service-context, decides which modification is necessary in order to accommodate the service, and an assembler which executes what the adaptor decides. The adapter uses the existing components as adaptive solutions (by adding, modifying or removing components). These components are predefined by the application designer and grouped into a component-based set.

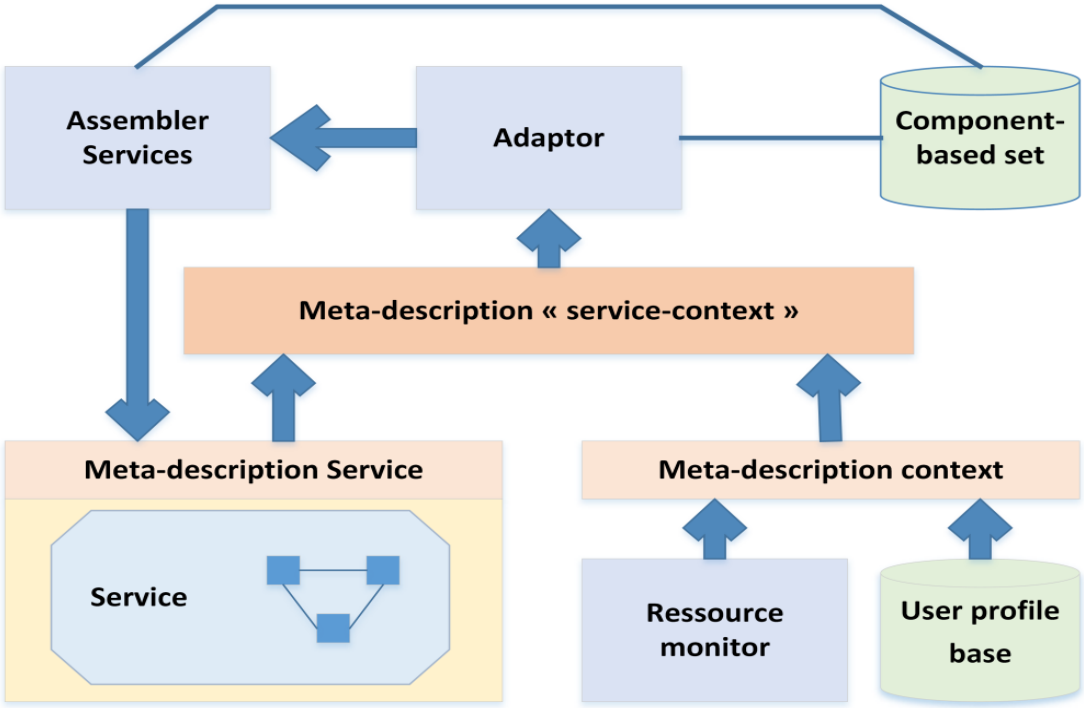

Fig. 1. Architecture for dynamic adaptation of service

A. Illustrative example

An electronic forum service or bulletin board system (BBS) allows a community of students to exchange information on scientific and cultural activities taking place within the university.

For practical reasons, the language of communication is the French language, and student members have the opportunity to write and read on the forum.

B. Identify the Headings

Figure 2 illustrates a perspective on the meta-description of the service-context that corresponds to the scenario presented in the illustrative example. Three different levels are found:

- The user plane (UP) which includes the physical elements of the context
- The arrangement scheme of components containing the architecture description language (ADL) of the service as well as the interface definition language (IDL) of the components.
- The ADLs describe the software architectures. The rewriting rules capture and combine the adaptations in the form of sets of rules (patterns) and facilitate their selection.
- The profiles plane combines the service profile and the context profile; it indicates the operation of the service.

a) The user plane – physical elements of context: The physical plane comprises the physical elements of the context. In this case, these elements are the users of the forum service. The elements found in this plane have projections in the profile plane. These projections constitute the meta-description of the context.

b) The compnent plane – Service, components, and assembly: The component plane includes service S in the form of an assembly of software components, which form the basis of the Bundles of the OSGI platform. In our example, this plane consists of:

- Component A to display messages posted to the forum,
- Component E to write a new message,
- Component F to represent the forum containing all the messages posted to the forum.

The HMI of service S is developed from a composition of the HMIs of components E and A. The component plane represents the syntactic part of the meta-description of the service. To describe the service in this plane, an architecture description language is used to determine the internal architecture of the service, the IDL descriptors and the interconnections, which are MANIFEST.MF files for the components. The HMI is also included in the interconnections (or connectors).

c) The profile plane - User profile, component profile, composition of profiles: The profile plane is essential for adaptation, because it represents the semantic part of the meta-description of the service-context. For the chosen example, this plane contains:

The profile service (PS) and the user profile (UP). The profile service (PS) results from the combination of profiles of components that make up the S Service, i.e. profile for writing a message (WP), profile for displaying a message (DP), and

forum profile (FP). This plane contains the semantics of the service-context.

In our example, the user profile contains one single parameter, namely language = 'Ar' indicating a user who writes Arabic. The profile of a component indicates how this component operates with respect to the context parameters. The profile for a translation component Arabic – French is given next:

```
<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 indicates the behavior of the component relative to the language; at the interface "Translation" of the component "Translation AR_FR", there is a pre-condition on the parameter "language" which requests the value "AR".

The returned value gives the value "FR". Therefore, this component changes the language. Thanks to the profile, the adapter can discover this fact.

The service profile, which is a composite component, results from the composition of the profiles of components W (write), D (display), F (forum); conventionally, F requires language = 'FR'.

d) Adaptation axioms of the service-context: Regarding the semantic part of the service-context, i.e. the one corresponding to the profile plane, there is a need to define the axioms (evidence or condition) that are verified to see if a service is adapted to its context. For the example under consideration, there is one axiom only. There is a need to adapt the service-context if the profile parameter values are contradictory. In our example, if the user writes in Arabic, there is a contradiction for the "language" parameter (Ar <> Fr) at the HMI of service S. In this case, the adaptation axiom is not verified.

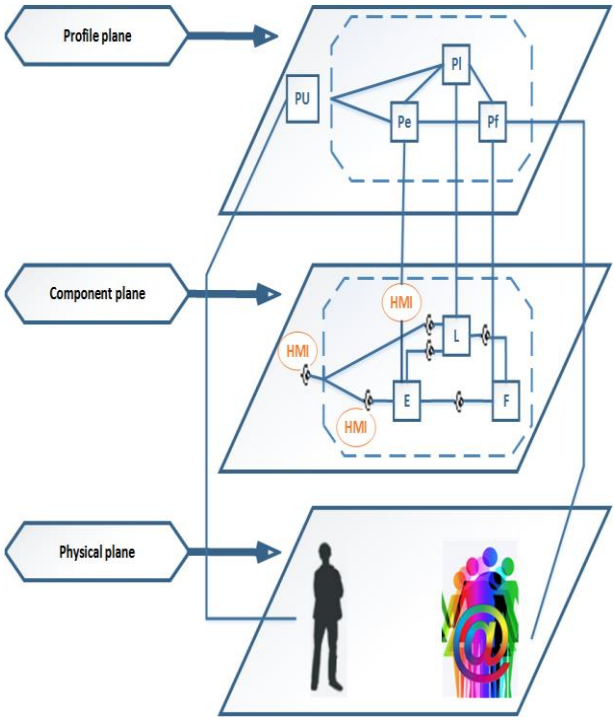

Fig. 2. Meta-description of the service-context - Various view planes

C. The adapter - Algorithm

The adapter should be able to adapt the service; its automation requires the use of dynamic formalisms and proceeds in three stages [20]:

- A technique is developed to detect in compatibilities,
- An abstract description of the properties of the adapted system, the mediator between components (adapter), or a simple correspondence between interfaces is provided; this is called the mapping of adaptation,
- The adaptation process is then generated automatically, using that Mapping with the interfaces of the components to be adapted. When the adapter is placed within the execution context, it allows correcting the system.

The Mapping of adaptation can be obtained automatically by using techniques from the field of semantic Web services [21]; It may also be provided by the software architect, and this is the case treated here.

An adapter is proposed here to perform the following operations:

a) Checking: The adaptation axioms are verified, for each parameter of the profiles; if they are all satisfied, it is an adapted service.

b) Searching for an adaptation solution: . If for at least one parameter, an axiom is not satisfied, an adaptation should be found:

- For each parameter that does not satisfy the axioms, the adapter builds a graph with nodes representing the component interfaces which have a relationship with that parameter and whose arcs are the interconnections,
- In this graph, the adapter locates the branches (chain of nodes) containing unequal values. For each branch, the adapter searches for components whose insertion in this branch helps restore the equal values.

c) applying the solution: if the adapter reaches that point, it can apply the solution obtained.

V. PROTOTYPE

The proposed architecture consists of three parts, as illustrated in Figure 1:

A – Display the forum

E - Write a message,

Proxy F - a local connector towards the remote component F - Forum.

Component T - Translation is dynamically added to the request of the adapter, using:

- The Wcomp platform [22] which offers a development environment based on software components; it allows creating dynamically the interconnections with ISL4Wcomp. The dynamic adaptation to Wcomp is done by the addition, removal, connection and disconnection of the software components during the execution of the application.
- The interaction language ISL4Wcomp [23] (Interaction Specification Language For Wcomp) is based on the interaction specification language (ISL) which enables to describe the interaction patterns between objects [24]. However, ISL4Wcomp tends to adapt these specifications in order to take into account the interactions that are based on messages or events in the components encountered in the assembly aspects.

The adapter finds the translation component T in the component directory. The search is carried out from the component profile while respecting the type of connections.

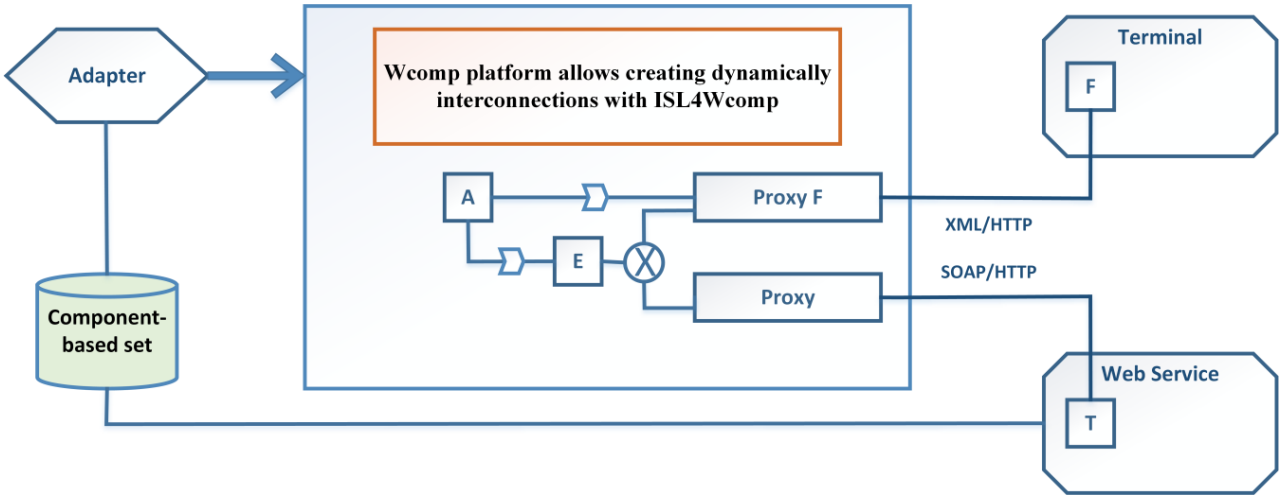

Fig. 3. Schematic of the prototype

VI. CONCLUSION

An architecture that allows the dynamic adaptation of services is developed in this article. Our proposal is based on the meta-description of the service-context in which we have a limited set of adaptation axioms that are based on the semantics of the service. It is not necessary to describe the different evolution rules; they can be found through the analysis of inadequacy cases, under the condition that each component is described with its profile.

To show that this architecture works, a prototype of forum service was proposed. This service was initially created for French-speaking users, but the proposed architecture enables adapting this service by dynamically adding a translation component if the user's language is not French.

The main weakness of the proposed architecture is its complexity. The suggested model still needs to be generalized and simplified so that the various profile parameters may combine in a simpler manner, while retaining the richness which is required to express the axioms of adaptation.

Despite these limitations and difficulties, the authors believe that the future of adaptation belongs to the semantic composition of services. To achieve this, the functioning of the service-context must be understandable both for the machine and the human services developer.

REFERENCES

[1] Akkawi, F., Akkawi, K., Bader, A., Ayyash, M., Fletcher, D., Alzoubi, K., 2007,March. Software adaptation: a conscious design for oblivious programmers. In:Proceedings of the IEEE Aerospace Conference, pp. 1–12.

[2] McKinley, P.K., Sadjadi, S.M., Kasten, E.P., Cheng, B.H.C., 2004, July. Composing adap-tive software. Computer 37, 56–64.

[3] Cetina, C., Giner, P., Fons, J., Pelechano, V., 2009, October. Autonomic computingthrough reuse of variability models at runtime: the case of smart homes. Com-puter 42, 37–43.

[4] Alférez, G.H., Pelechano, V.,2011a. Context-aware autonomous web services in soft-ware product lines. In: Proceedings of the 2011 15th International SoftwareProduct Line Conference. SPLC'11. IEEE Computer Society, Washington, DC, USA,pp. 100–109.

[5] Koning, M., Sun, C.-a., Sinnema, M., Avgeriou, P., 2009, February. VxBPEL: supportingvariability for web services in BPEL. Information and Software Technology 51,258–269.

[6] Cardoso, J., Miller, J., Sheth, A., and Arnold, J. (2002). Modeling quality of service for workflows and web service processes. Journal of Web Semantics, 1 :281–308.

[7] C. Jaeger, M. (2007). Optimising Quality-of-Service for the Composition of Electronic Services. PhD thesis, Berlin University, Germany.

[8] Rosenberg, F. (2009). QoS-Aware Composition of Adaptive Service-Oriented Systems. PhD thesis, Technical University Vienna, Austria.

[9] Coppolino, L., Romano, L., Mazzocca, N., and Salvi, S. (2007). Web

[10] services workflow reliability estimation through reliability patterns. Security and Privacy in Communications Networks and the Workshops,.

[11] Colombo, M., Di Nitto, E., Mauri, M., 2006. SCENE: a service composition exe-cution environment supporting dynamic changes disciplined through rules.In: Dan, A., Lamersdorf, W. (Eds.), Service-Oriented Computing – ICSOC 2006.Vol. 4294 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg,pp. 191–202.

[12] Baresi, L., Guinea, S., 2011, March. Self-supervising BPEL processes. IEEE Transactions on Software Engineering 37, 247–263.

[13] Narendra, N.C., Ponnalagu, K., Krishnamurthy, J., Ramkumar, R.,2007. Run-timeadaptation of non-functional properties of composite web services using aspect-oriented programming. In: Proceedings of the 5th International Conference on Service-Oriented Computing. ICSOC'07. Springer-Verlag, Berlin, Heidelberg, pp.546–557.

[14] Sonntag, M., Karastoyanova, D.,2011, August. Compensation of adapted service orchestration logic in BPEL'n'aspects. In: Proceedings of the 9th InternationalConference on Business Process Management (BPM 2011). Springer-Verlag,Clermont-Ferrand, France, pp. 1–16.

[15] Moser, O., Rosenberg, F., Dustdar, S.,2008. Non-intrusive monitoring and serviceadaptation for WS-BPEL. In: Proceedings of the 17th International Con-ference on World Wide Web. WWW'08. ACM, New York, NY, USA,pp. 815–824.

[16] Fleurey, F., Solberg, A.,2009. A domain specific modeling language supportingspecification, simulation and execution of dynamic adaptive systems. In:Proceedings of the 12th International Conference on Model Driven Engineer-ing Languages and Systems. MODELS'09. Springer-Verlag, Berlin, Heidelberg,pp. 606–621.

[17] Erradi, A., Maheshwari, P.,2005. wsBus: QoS-aware middleware for reliable webservices interactions. In: Proceedings of the 2005 IEEE International Confer-ence on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology,e-Commerce and e-Service. EEE'05. IEEE Computer Society, Washington, DC,USA, pp. 634–639.

[18] Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F., 2010. Adaptive management ofcomposite services under percentile-based service level agreements. In: Maglio,P., Weske, M., Yang, J., Fantinato, M. (Eds.), Service-Oriented Computing. Vol.6470 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp.381–395.

[19] Mosincat, A., Binder, W.,2008. Transparent runtime adaptability for BPEL processes.In: Proceedings of the 6th International Conference on Service-Oriented Com-puting. ICSOC'08. Springer-Verlag, Berlin, Heidelberg, pp. 241–255.

[20] Oliveira, N, Barbosa LS (2014) A Self-adaptation Strategy for Service-based Architectures In: VIII Brazilian Symposium on Software Components, Architectures and Reuse. SBCARS'2014, 44–53.. IEEE, Maceió, Alagoas.Google Scholar

[21] CANAL C., MURILLO J. M., POIZAT P., « Software Adaptation », L'Objet. Special Issue on Coordination and Adaptation Techniques, vol. 12, n o 1, 2006, p. 9-31.

[22] BEN MOKHTAR S., GEORGANTAS N., ISSARNY V., « Ad Hoc Composition of User Tasks in Pervasive Computing Environments », Software Composition, Springer Verlag, LNCS 3628, 2005, p. 31-46.

[23] Cheung-Foo-Wo D., Blay-Fornarino M ., Tigli J-Y., Lavirotte S., Riveill M., « Adaptation dynamique d'assemblage de dispositifs par des modèles », 2ème journée sur l'ingénierie dirigée par les modèles (IDM), 2006a.

[24] Blay-Fornarino M., Charfi A., Emsellem D., Pinna-Dery A.-M., Riveill M., « Software interactions », Journal Of Object Technology, vol. 3, n° 10, p. 161-180, 2004.

[25] Berger L., Mise en Oeuvre des Interactions en Environnements Distribués, Compilés et Fortement Typés : le Modèle MICADO, Thèse de doctorat, Université de Nice-Sophia Antipolis - Faculté des sciences et techniques, Ecole doctorale STIC - Informatique, Octobre, 2001.