A UML based deployment and management modeling for cooperative and distributed applications

Mohamed Nadhmi Miladi, Fatma Krichen, Mohamed Jmaiel, Khalil Drira

A UML based deployment and management modeling for cooperative and distributed applications

Mohamed Nadhmi MILADI[1], Fatma KRICHEN[1], Mohamed JMAIEL[1], and Khalil DRIRA[2,3]

[1] University of Sfax, ReDCAD laboratory, ENIS, Box.W 1173, 3038, Sfax, Tunisia, MohamedNadhmi.Miladi@isimsf.rnu.tn,
[2] CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
[3] Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France

Abstract. Thanks to the major evolutions in the communication technologies and in order to deal with a continuous increase in systems complexity, current applications have to cooperate to achieve a common goal. Modeling such cooperatives applications should stress regular context evolutions and increasingly users requirements. Therefore, we look for a model based solution suitable to cooperative application that can react in response to several unpredictable changes. Driven by the cooperative application structure, we propose, in this paper, an UML extension named "DM profile" ensuring a high-level description for modeling the deployment and its management in distributed application. The proposed contribution is validated through a "Follow Me" case study and implemented through an Eclipse plug-in.

1 Introduction

Current distributed systems are continuously increasing in size and especially in complexity. Cooperating several software entities, to achieve a common goal, is a key to cope with such complexity. These cooperative applications have to adapt their deployed architectures due to several purposes: improving performance, evolutionary user requirements, context changes, etc.

A successful adaptation is based on providing architecture deployment models that can be dynamically managed to meet such required purposes. This deployment and management modeling should be, in addition, suitable to the distributed and cooperative features of the application. However, modeling the architecture deployment and its management is often closely coupled to the underlying supported platform. Such a description especially targets the modeling of real deployment units such as artifacts, communication links, and computing units. This requires knowing the context and the underlying deployment platforms before starting the deployment process. For the actual systems where adaptively properties are usually unpredictable, such a deployment and its management modeling remains inappropriate. User requests and the application context are continuously evolving. In addition, the availability of deployment structures such as deployment platform and communication flows are not always guaranteed.

The challenge is to design high-level deployment and management solution that can easily handles diverse deployment infrastructures. This modeling should provide not only platform-independent models, but also a modeling abstraction that handles various architecture deployment approaches applicable for the service-oriented and the component-based architectures. This deployment and management modeling should ensure a best effort adaptation while taking advantage of all available resources whatever their architecture development approach or their underling platform are. Moreover, such modeling should take advantage of the structured organization of the cooperative application.

Basing on the UML standard language, several works propose extensions to handle software architecture deployment such as [9], [12], [3], and [22]. These works follows a structural management reconfiguration on behalf of a deployment management modeling. Other works focus on a high-level modeling that describes the deployment management such as [10], [13], and [17].

The contribution made by this paper merges both: the modeling power of the UML language specification and a high-level modeling. It proposes an UML profile extension providing an abstract model that describes the deployment and its management of distributed software architectures while taking into consideration cooperative architectures specificities. This modeling ensures a best effort solution for the management of an architecture deployment to meet their adaptiveness requirements.

The rest of this paper is organized as follows: Section 2 discuses the related work. In section 3, we present our UML extension profile. Then, in Section 4, we illustrate our extension by a case study Follow Me. Section 5 presents the realization of our profile as a plug-in for eclipse. Finally, section 6 concludes this paper and presents future work directions.

2 Related work

Many works dynamically manage their software architecture in response to the context evolution requirement. Various management techniques are used. We distinguish the interface management, implementation management, structure management [11], [15]. Other works including [23] merge some of these techniques in order to ensure dynamic management. Despite these research efforts, they remains not well appropriate with the distributed architecture specificities. Works including [13] rely on a deployment management for the context adaptivity requirement of distributed architectures. Modeling dynamic architecture management carries several techniques including ADLs (Architecture Description Languages), formal techniques, graphical techniques.... Several research works focus on standard modeling techniques which are based especially on the UML language and its standard mechanisms. Since our proposed contribution follows a standard modeling techniques and it grants a dis-

tributed architecture managing, we focus in this section on researches that handle both: the deployment management and UML extension modeling.

Modeling deployment management through UML extensions can be subdivided into two main issues. The "heavyweight" [21], [22] which set new meta-classes extending UML meta-classes. The other category defines UML profiles while maintaining the UML meta-model. Among these works some researches focus on a context modeling using an UML profile in order to meet the adaptiveness requirements. Some other research efforts including [8], [12], [18] are based on the UML component diagram to achieve a dynamics structural management of architecture applications. However, such works describe a structural management reconfiguration instead of a deployment management modeling.

In order to manage the deployed software architecture, works including [9] are based on a UML profile extending the modeling power of the deployment diagram. Other research efforts addressing the deployment management modeling driven by the architecture type. Works including [20] focus on a service-oriented approach while others including [5] opted for a component-based approach. A third class including [10], [13] and [17] provides a high level description for modeling the deployment and its management.

A final group of research efforts, including [2] merges both: a deployment management modeling based on the UML language and a high-level of abstraction modeling. Our contribution match these efforts while providing a more explicit models for the deployment and its management description especially for cooperative applications. It provides a description that takes advantages of the modeling power of a model based description, a high-level of abstraction modeling and a deployment and management modeling. It is also based on standard direction as well for adopted approach: the MDA approach, as for the used modeling language: the UML language.

3 The deployment and management (DM) profile

This work addresses the deployment and its management modeling for software architectures in general and more specifically for collaborative applications. It proposes models with a high-level of abstraction ensuring a platform independent deployment description as well as a best effort management solution for adaptive requirement evolving.

This work is the high-level of a multi-level based approach 1. Based on the MDA approach, the proposed models are transformed towards more specific models through a model transformation process. This process tends to reduce the gap between our abstract models and a more refined model supporting service-oriented [19] or component-based Architectures. These models are mapped towards specific platforms such as CCM [6], OSGi [1], and other platforms as depicted in Figure 1.

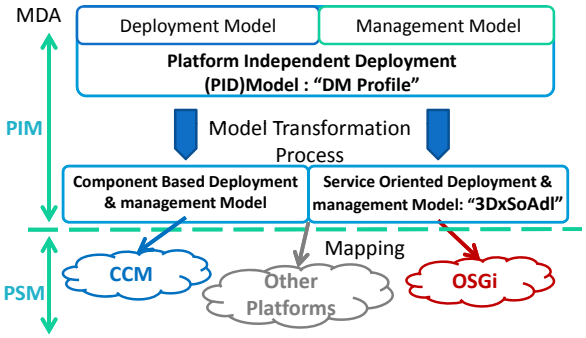

Fig. 1. The multi-level based approach for deploying and managing software architectures

3.1 The deployment model

The architecture deployment modeling is based on two major ideas. The first idea highlights a deployment model without any prior description of the real deployment architecture entities. This modeling is based on the Unified Modeling Language. The architecture deployment modeling in UML2 is ensured mainly through the deployment diagram. Meta-models of this diagram, which are related to the L2 level of the MOF approach, provide the basic concept for modeling deployment architectures.

In UML2, the deployment modeling process is closely coupled with the real physical architectural entities. Most provided meta-models and versioning update efforts on this diagram follow a modeling vision that focus on the description of physical architecture entities such as "device", "execution environment", "artifact".... Thus, the deployment diagram is associated with a PSM level. In the proposed deployment model, we extend the modeling power of UML deployment diagram to enable a PIM level deployment modeling. This extension ensures a high-level description to model an architectural deployment process. Two new stereotypes are achieved to overtake such deployment modeling:

≪Deployed Entity≫ stereotype: it ensures the modeling of the functional aspect of the deployment architecture. "Deployed Entity" models each software entity that establishes a functional contract through its provided and required interfaces and can be deployed in a container. Such modeling enhances the description power of UML component model in order to cover several software entities including components, services, service-components [4]... Therefore, it describes the basic software unit in a high deployment level. The defined stereotype extends the component meta-model, as depicted in Figure 2, establishing a more abstract semantic in a deployment context.

≪Logical Connection≫ stereotype: it ensures the connections modeling between deployed entities of the architecture. The described connections span several connection types ensuring the communication between two "deployed entities". The

"Logical Connection" stereotype enhances the connector semantics enabling to span physical connections such wired, wireless or satellites connections, software connections, or connections that express functional dependencies between two "Deployed Entities". This stereotype extends the connector meta-model of UML as depicted in Figure 2.

Modeling software architecture deployment requires the description of, first, the suitable/available deployment containers, second, the deployable software entities such as components and services in their related nodes and, third, the connection links within these entities. This modeling, although it's higher-level description, remains especially focused on concepts reduced to their location typically modeled through the node concept.

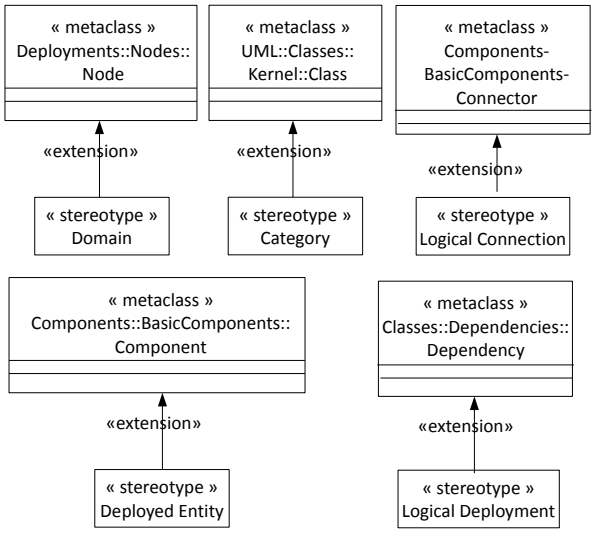

Fig. 2. "DM profile" stereotypes for the deployment modeling

Our second basic idea is to extend the container deployment scope using some virtual structures which have a more enhanced semantics than the traditional node concept. These structures ensure a more flexibility and better organization in the deployment process. Each entity will be deployed in virtual structures meeting the context requirements and accustoming to the available resources. In addition, structuring software entities under some virtual structures emphasize the cooperative aspect of an application and match its cooperative guard. Typically, in these applications, deployed entities are virtually underlying other entities to cooperate in the achievement of the same goal. In a cooperative application, several actors should be established. Each one plays a specific role in the cooperation process. For instance,

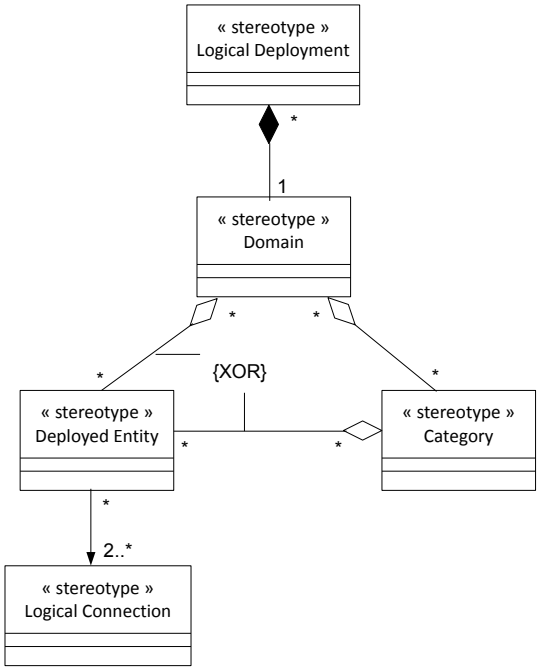

Fig. 3. 'DM profile" structure for the deployment modeling

in a cooperative document edition, various roles are identified such as writer, reader, manager... This carries a vertical vision, as depicted in Figure 4, of the cooperative deployment structure. On the other hand, actors cooperate in the establishment of complex activities to meet a common goal through several activity categories. For example, in a cooperative document edition, we can identify various categories such writing, document correction, review.... This carries a horizontal vision, as depicted in Figure 4, of the cooperative deployment structure. Moreover, it achieves a better management of the deployed architecture. This will be more detailed in the next section.

This structured modeling is achieved through the definition of three new stereotypes on our profile:

≪Domain≫ stereotype: it models a virtual structure owning "Deployed Entities" that ensure the same role. This stereotype extends the node meta-model, as depicted in Figure 2, establishing a more specific semantics in the deployment context of cooperative application.

≪Category≫ stereotype: it models a virtual structure owning "Deployed Entities" that cooperate in the same activity category. This stereotype extends the class meta-model, as depicted in Figure 2, establishing a more specific classification of the "Deployed Entities" upon cooperative activities process.

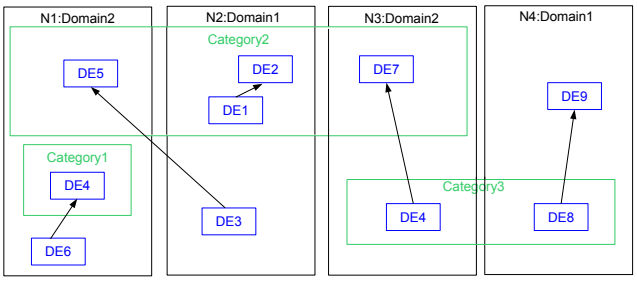

Fig. 4. A description of the cooperative deployment structure

≪Logical deployment≫ stereotype: it expresses all available dependencies within the profile stereotypes. This stereotype extends the dependency meta-model, as depicted in Figure 2, establishing deployment dependency in cooperative application. Restrictions bound to the definition of this profile depicted in Figure 3 presented in the following:

- A relation of composition is established between the stereotype "Logical Deployment" and the stereotype "Domain", while a deployed architecture includes several "Domain";
- A relation of aggregation is established between the stereotype "Domain" and the stereotype "Category" (respectively "Deployed Entity"), while the same "Category" (respectively "Deployed Entity") can belong to several "Domains";
- A relation of aggregation is established between the stereotype "Category" and the stereotype "Deployed Entity", while the same "deployed entity" can to be deployed in several categories;
- The relation "XOR" between the two previous aggregation relations ensuring that a "Deployed entity" belongs to a "Domain" as well as to a "Category" but not at the same time;
- An association between the stereotype "Deployed Entity" and the stereotype "Logical Connection", while two or more "Deployed entities" can be connected through several "logical connections".

3.2 The management model

In order to take over an evolving context and unexpected events or requirements, a dynamic architecture deployment should be modeled. The challenge is to provide notations and mechanisms to cope with the current architecture properties such as large scale deployment while being able to target the appropriate entities to manage. The idea is to describe specific redeployment rules. A redeployment rule describes

what's transformation should be achieved on the fly upon the actual deployment architecture to meet the requirement adaptations.

In a large-scale deployment, in order to meet unexpected events or requirements, several redeployment rules should be achieved. In fact, an occurrence of a single event may lead to achieve a set of redeployment rules. In a cooperative context, such redeployment rules may affect all "deployed entities" without expecting their task in a cooperative process. These rules should target some specific "Deployed Entities" establishing a cooperative task. Describing all these redeployment rules requires both a meticulous and an excessive modeling time and efforts. This modeling solution became especially limited in applications that require a quick and specific management description. The idea is to propose a more expanded redeployment rules. The deployment scopes of such rules are no longer limited to a single "deployed entity" but they handle a set of "deployment entities". The execution of a single expanded deployment rule induces the execution of several underlying deployment rules. Establishing these expanded rules is based on the deployment structures proposed in the previous section including "Domain" and "Category". This enables a more target deployment management driven by the cooperative aspects of the application. Describing these expanded redeployment rules is based on a multi-formalism management approach.

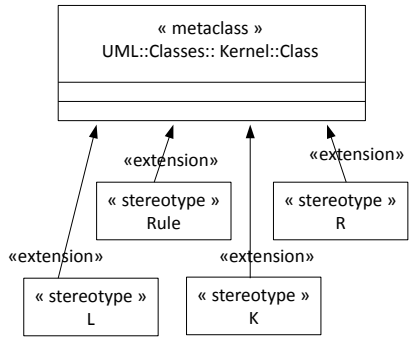

Fig. 5. "DM profile" stereotypes for the management modeling

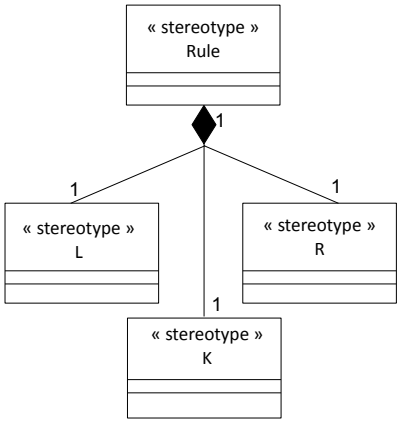

Fig. 6. "DM profile" structure for the management modeling

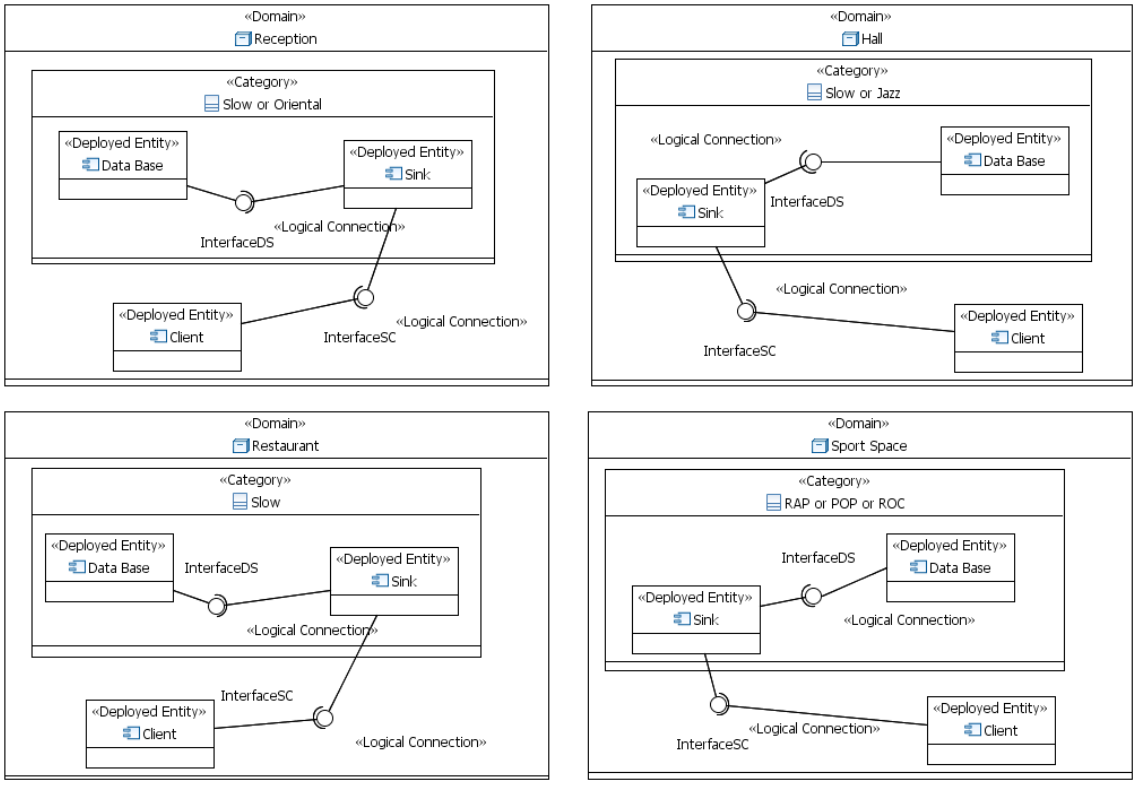

Fig. 7. "Follow Me" high-level deployed structured modeling

This management approach combines the power of two formalisms. First, it is based on the theoretical efforts achieved on grammar productions techniques such as graph DPO, Δ, Y [16], [12]. We are based especially on the DPO technique. DPO is a richer structure for grammar productions. These productions are specified with a triplet <L;K;R>. The application of this production is achieved through the removal of the graph corresponding to the occurrence of Del=(L\K) and the insertion of a copy of the graph Add=(R\K). Indeed, DPO technique, describes each uplet graph as an autonomic entity. There is no exclusion as the Y and/or Δ techniques. This is very useful when, for instance, we should express the relationship of two elements sharing the same container; one should be preserved and the other one should be added.

Second, the proposed management approach is based on the high expressive power of UML language and its standard notations. Based on a graphical notation, our profile provides a clear solution modeling for dynamic deployment management. This solution is achieved through four new stereotypes as depicted in Figure 5:

≪Rule≫ stereotype: It models an expanded redeployment rule that its description is ensured through the three following stereotypes: **≪L≫**, **≪K≫**, and **≪R≫** stereotypes. **≪L≫** stereotype presents the initial sub-architecture from the system where the redeployment rule can be applied. **≪K≫** stereotype presents the sub-architecture to preserve from the sub-architecture stereotyped by **≪L≫** stereotype. **≪R≫** stereotype present the sub-architecture after the execution of the rule. Otherwise, **≪L≫****≪K≫** (respectively **≪K≫**, **≪R≫****≪K≫**) models the deployment structures ("Domain", "Category") and their owning "Deployed Entities" to be deleted (respectively preserved, added) in the rule execution. In other words, after the execution of a redeployment rule the current deployed architecture shifts by adding **≪R≫****≪K≫** sub-architecture, keeping **≪K≫** sub-architecture and deleting **≪L≫****≪K≫** sub-architecture.

A restriction, depicted in Figure 6, bound to the definition of this profile. In fact, a relation of composition is established between the stereotype "Rule" and the stereotype "L" ("K" and "R"), because the same sub-architecture "L" ("K" or "R") can belong to only one reconfiguration rule.

4 Case study: Follow Me

In this section, we present a case study called "Follow Me" for illustrating our profiles. The "Follow Me" case study, which is similar to the one presented in [14], is an adaptive application reacting on the context change. It is an audio application whose audio flow follows the listener movement among many rooms. Each room has some predefined sinks (a player and its speakers). The "Follow Me" architecture ensures the context adaptation thanks to its deployment management ability. In fact, "Follow Me" architecture depends on the available system listeners. If there is

no person present in a given room, sinks stop the audio flow. In addition, if a listener moves from a room to another, audio flow follows him/her in a continuous manner.

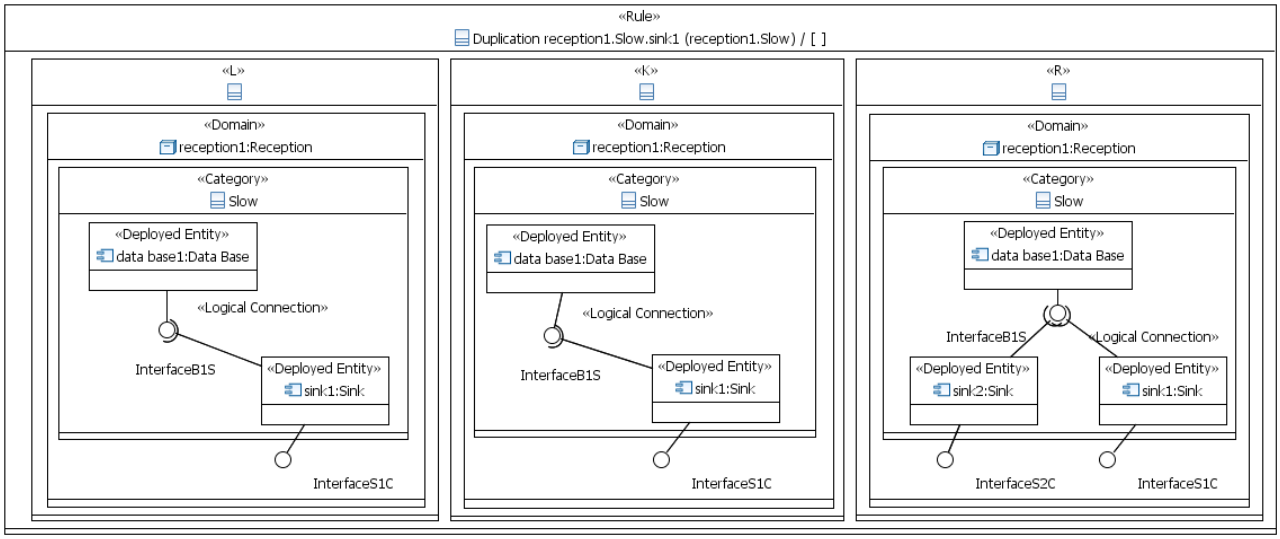

Fig. 8. High-level Rule for the duplication of "Sink" instance

As an example, we present the case of hotel that has a set of rooms with different roles: reception, hall, restaurant.... Each room has a data base which contains songs and players which play music. Rooms sharing the same role play the same music category. Besides, each room can provides to the maximum three players, and each player can serve to the maximum fifteen clients in order to offer better audio flow quality.

In order to describe our case, we need to have some virtual structures which have a more enhanced semantics than the traditional node concept. Thus, we defined the two semantics domains and categories. Additionally, this system must react with the change of context. For that and in order to handle the cooperative aspect, we describe this architecture with a high-level of description: deployed entities interconnect with logical connectors. The following section describe with details our case study "Follow Me" of a hotel.

4.1 Structural architecture description

In this section, we propose a modeling process that guide the architecture designer in order to describe a high-level modeling of the deployed structured architecture. First, we begin by identifying the various architecture types (domains, categories and deployed entities), then we associate categories to domains and deployed entities to domains and categories.

Figure 7 describes deployed structured architecture of the "Follow Me" case study , and in particular the case of hotel, through our realized plug-in and profile.

Identify architecture types:

- Domains: Reception, Hall, Restaurant, Sport Space
- Categories: Slow, Oriental, Jazz, RAP, POP, ROC
- Deployed entities: Sink, Client, Data Base
  - Sink: It provide audio flow
  - Client: He use audio service
  - Data Base: this is an audio data base
- Logical Connections
  - Connection between Data Base and Sink
  - Connection between Sink and Client

Associate categories to domains:

- Reception = Slow, Oriental
- Hall = Slow, Jazz
- Restaurant = Slow
- Sport Space = RAP, POP, ROC

Associate deploymed entities to domains and categories:

- Deployed entities Sink and Data Base belong to the defined categories
- Deployed entities Client belong to defined domains

From this high-level description of the "Follow Me" architecture, we can define several architectural instances of hotels.

Besides, this high level description enables us to specify the different reconfiguration rules of our case study. In the following, we present an example of reconfiguration rules. Such rules model some of elementary redeployment actions that can be applied on deployed structured architecture instance.

4.2 Reconfiguration architecture description

In order to highlight the dynamic management aspect, we consider the case of the arrival of client number sixteen in the reception, a duplication of the deployed entity "Sink" instance should be achieved to serve the new client. Figure 8 models the duplication of "sink1" instance (regarding the deployed entity "Sink") in the category "Slow" in the instance "reception1" regarding the "Reception" domain :

- Add new instance sink2 of deployed entity Sink identical to instance sink1 (same state)
- Add connection between the new instance of deployed entity Sink and the instance data base1 of deployed entity Data Base

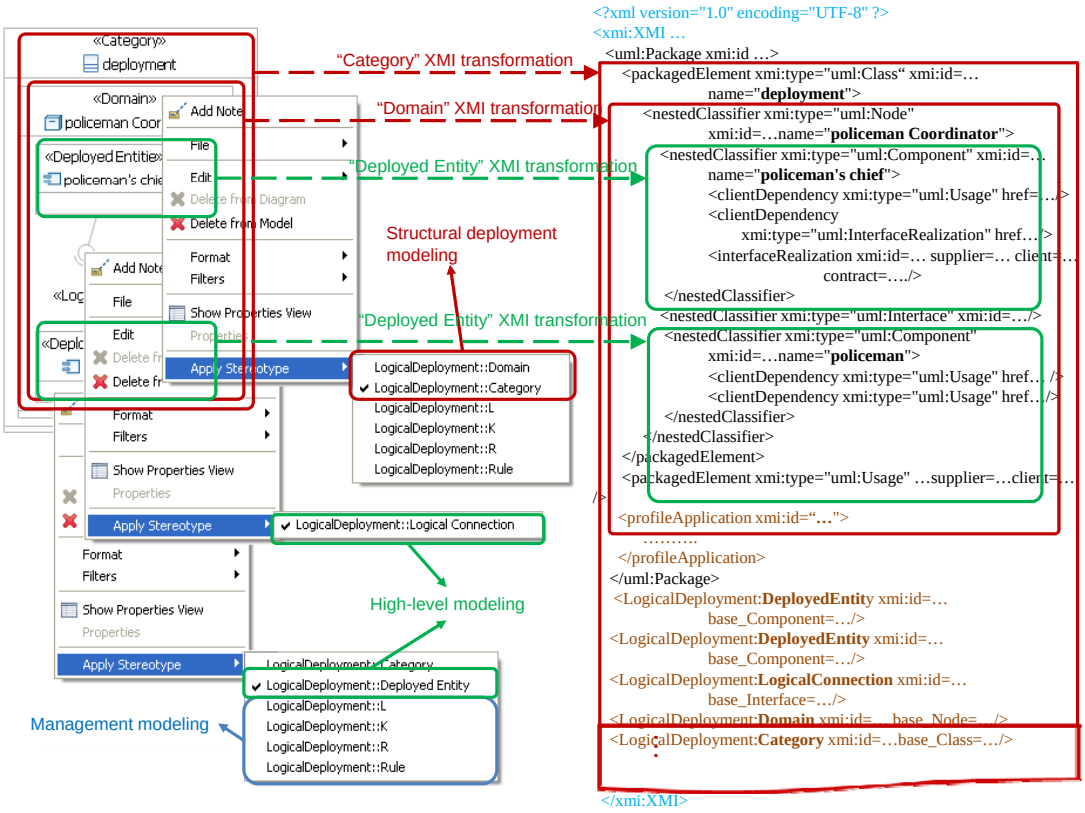

Fig. 9. Mapping of the "DM profile" models towards XML language

5 Eclipse plug-in extension

In this section, we provide an UML graphical editor as a plug-in in Eclipse that implements the proposed "DM profile". It ensures a technical issue to model a structural deployment and its management in cooperative distributed architectures. Implementing the "DM Plug-in" is directed by the "UML2Tools" [7] project. This project aims at providing a graphical solution for modeling UML diagrams with respect to their latest version. More specifically, the "DM plug-in" is implemented using the "UML2Tools" deployment diagram. In addition, developing the proposed plug-in is based on several frameworks: the GMF framework (Graphical Modeling Framework) for graphical editor generation, The EMF Framework (Eclipse Modeling Framework) for meta-model construction, and the GEF Framework (Graphical Editing Framework) for graphical drawings.

The Developed "DM Plug-in" did not guarantee only a graphical modeling but also it ensures the mapping of the achieved models towards the XML language as depicted in figure 9. The resulted XML files are generated with respect to the XMI (XML Metadata Interchange) standard recommended by the OMG group. Baring in mind, all XMI files are automatically validated through XML schemas (integrated in the Plug-in). Generating XML files is a fundamental step in a refinement process which starts with already designed models towards a more platform specific description.

Figure 9 shows the different stereotypes defined in our profile and applied in a simple example. For each graphical modelling, an XMI file is generated which contains with details all used UML models and stereotypes.

6 Conclusion

In this paper, we propose an UML profile, named "DM profile" for the deployment and its management in cooperative and distributed architectures. Driven by the cooperative system structure, the proposed profile ensures: first, a high level description for a deployment modeling decoupled with related platforms and architecture style specificities. Second, an explicit model for managing the deployed architecture based on graph transformation theories. Third, a suitable solution for a large scale deployment of cooperative architectures. The proposed solution is illustrated through a follow me example and implemented through an Eclipse plug-in.

The modeling solution depicted, here, ensures a generic and a platform independent modeling according to the MDA approach. In our future works, we will focus on the model transformation process that fit our multi-level based approach introduced in the top of "The deployment and management (DM) profile" section. This approach seeks to refine the "DM profile" models towards a deployment and management description [19] which can easily mapped to a specific platform such

as OSGi. Such model transformation process, is driven by a set of implemented algorithms that translates our high-level models to a set of elementary redeployment actions more suitable with specific platform management.

References

1. O. Alliance, "sgi service platform core specifcation the osgi alliance," http://www.osgi.org/Download/Release4V41, April 2007, release 4, version 4.1.
2. J. P. A. Almeida, M. van Sinderen, L. F. Pires, and M. Wegdam, "Platform-independent dynamic reconfiguration of distributed applications," in *Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS 2004)*. IEEE Computer Society, May 2004, pp. 286–291.
3. D. Ayed and Y. Berbers, "UML profile for the design of a platform-independent context-aware applications," in *Proceedings of the 1st workshop on MOdel Driven Development for Middleware (MODDM 06)*. New York, NY, USA: ACM, 2006, pp. 1–5.
4. F. Curbera, "Component contracts in service-oriented architectures," *Computer*, vol. 40, no. 11, pp. 74–80, 2007.
5. A. Dearle, G. N. C. Kirby, and A. J. McCarthy, "A Framework for Constraint-Based Deployment and Autonomic Management of Distributed Applications," in *Proceedings of the 1st International Conference on Autonomic Computing (ICAC 2004)*. IEEE Computer Society, May 2004, pp. 300–301.
6. G. Deng, J. Balasubramanian, W. Otte, D. Schmidt, and A. Gokhale, "Dance: A qos-enabled component deployment and configuration engine," in *in Proceedings of the 3rd Working Conference on Component Deployment*, 2005, pp. 67–82.
7. T. E. Foundation, "UML2 Tools," http://www.eclipse.org/modeling/mdt/ downloads/project=uml2tools.
8. S. Göbel, "An MDA Approach for Adaptable Components," in *Proceedings of the first European Conference of Model Driven Architecture - Foundations and Applications(ECMDA-FA 05)*. Nuremberg, Germany: Springer, November 2005, pp. 74–87.
9. V. Grassi, R. Mirandola, and A. Sabetta, "A UML Profile to Model Mobile Systems," in *Proceedings of the 7th International Conference on The Unified Modelling Language: Modelling Languages and Applications*. Springer-Verlag, October 2004, pp. 128–142.
10. ——, "A model-driven approach to performability analysis of dynamically reconfigurable component-based systems," in *Proceedings of the 6th international workshop on Software and performance*. New York, NY, USA: ACM, 2007, pp. 103–114.
11. T. Han, T. Chen, and J. Lu, "Structure Analysis for Dynamic Software Architecture," in *Proceedings of the 6th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2005)*. Towson, Maryland, USA: IEEE Computer Society, May 2005, p. 338.
12. M. H. Kacem, M. N. Miladi, M. Jmaiel, A. H. Kacem, and K. Drira, "Towards a UML profile for the description of dynamic software architectures," in *Component-Oriented Enterprise Applications, Proceedings of the Conference on Component-Oriented Enterprise Applications (COEA 2005)*, ser. LNI. GI, September 2005, pp. 25–39.
13. A. Ketfi and N. Belkhatir, "Model-driven framework for dynamic deployment and reconfiguration of component-based software systems," in *Proceedings of the 2005 symposia on Metainformatics (MIS 05)*. New York, NY, USA: ACM, 2005, p. 8.
14. R. Kirk and J. Newmarch, "A Location-aware, Service-based Audio System," in *Proceedings of the Second IEEE Consumer Communication and Networking Conference (CCNC 05)*. IEEE Computer Society, January 2005.
15. A. B. Letaifa, Z. Choukair, and S. Tabbane, "Dynamic Reconfiguration of Telecom Services Architectures According to Mobility and Traffic Models," in *Proceedings of the 18th International Conference on Advanced Information Networking and Applications (AINA 04)*. Fukuoka, Japan: IEEE Computer Society, March 2004, pp. 447–450.

16. I. Loulou, A. H. Kacem, M. Jmaiel, and K. Drira, "Towards a Unified Graph-Based Framework for Dynamic Component-Based Architectures Description in Z," in *Proceedings of the The IEEE/ACS International Conference on Pervasive Services (ICPS 04)*. IEEE, 2004, pp. 227–234.
17. M. Mikic-Rakic, S. Malek, N. Beckman, and N. Medvidovic, "A Tailorable Environment for Assessing the Quality of Deployment Architectures in Highly Distributed Settings," in *Proceedings of the Second International Working Conference on Component Deployment*. Springer, May 2004, pp. 1–17.
18. M. N. Miladi, M. H. Kacem, and M. Jmaiel, "A UML profile and a FUJABA plugin for modelling dynamic software architectures," in *MoDSE'07: Workshop on Model-Driven Software Evolution, March 20-23*. Amsterdam, Netherlands: IEEE - CSMR, March 2007.
19. M. N. Miladi, I. Krichen, M. Jmaiel, and K. Drira, "An xADL extension for managing dynamic deployment in distributed service oriented architectures," in *Proceedings of the 3rd International Conference on Fundamentals of Software Engineering (FSEN)*. Kish Island, Persian Gulf, Iran: Springer, April 2009, pp. 439–446.
20. F. Moo-Mena and K. Drira, "Reconfiguration of Web Services Architectures: A model-based approach," in *Proceedings of the 12th IEEE Symposium on Computers and Communications (ISCC 2007)*. IEEE Computer Society, July 2007, pp. 357–362.
21. J. E. Pérez-Martínez, "Heavyweight extensions to the UML v metamodel to describe the C3 architectural style," *SIGSOFT Softw. Eng. Notes*, vol. 28, no. 3, pp. 5–5, 2003.
22. A. Poggi, G. Rimassa, P. Turci, J. Odell, H. Mouratidis, and G. A. Manson, "Modeling Deployment and Mobility Issues in Multiagent Systems Using AUML," in *Proc. of the 4th International Workshop on Agent-Oriented Software Engineering IV (AOSE)*, ser. Lecture Notes in Computer Science, vol. 2935. Springer, 2003, pp. 69–84.
23. D. Walsh, F. Bordeleau, and B. Selic, "A Domain Model for Dynamic System Reconfiguration," in *Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2005)*. Springer-Verlag, October 2005, pp. 553–567.