Supporting Runtime Adaptation of Context-Aware Services

Boudjemaa Boudaa*
Département d'Informatique
Université Ibn Khaldoun
Tiaret, Algérie
boudjemaa.boudaa
@univ-tiaret.dz

Slimane Hammoudi
Equipe MODESTE
Groupe ESEO
Angers, France
slimane.hammoudi@eseo.fr

Abdelkader Bouguessa
Département d'Informatique
Université Ibn Khaldoun
Tiaret, Algérie
abdelkader.bouguessa
@gmail.com

Mohammed Amine Chikh
Département d'Informatique
Université Abou bekr Belkaid
Tlemcen, Algérie
mea_chikh
@mail.univ-tlemcen.dz

ABSTRACT

The intense use of mobile computing cutting-edge devices that characterizes our professional and social lives raises the need to personalise and adapt services according to the dynamic context frequently changes during the execution of these services. In the last decade, the field of context-aware services had led to emerge several works. However, most of the proposed approaches have not provided clear adaptation strategies in case of unforeseen contexts. Dealing with this last at runtime is also another crucial need that has been ignored in their proposals. This paper aims to propose a generic dynamic adaptation process as a phase in the model-driven development life-cycle for context-aware services using the control loop MAPE-K to meet the runtime adaptation. The proposed process is validated by implementing an illustrative example on FraSCAti platform. The principal advantage of this process is to sustain the self-reconfiguration of such services at model and code levels by enabling successive dynamic adaptations depending on volatile context.

Categories and Subject Descriptors

D.2.9 [**Software**]: SOFTWARE ENGINEERING—*Management*; H.3.5 [**Information Systems**]: INFORMATION STORAGE AND RETRIEVAL—*Online Information Services*

General Terms

Design, Management, Standardization

*Boudjemaa Boudaa is the corresponding author.

Keywords

Context-Aware Service, Runtime Adaptation, MAPE-K, MDD

1. INTRODUCTION

In the passage era from desktop computing paradigm to mobile and ubiquitous computing paradigm, the software applications operate in a variety of new wearable mobile devices accompanying the user in his professional and social life. They should be context-aware [17] using information about their execution context for responding and adapting to changes in the computing environment. In this regard, Context-Aware Services (CAS for short) should be capable to sense, and to collect the context information, so adapting their behaviours to significant context changes in order to provide personalised and relevant information and/or services to end-users.

This context can be static (information which is unlikely to change, e.g. mother tongue) or dynamic (information which changes over time; such as weather conditions) [22], therefore the consideration of both kinds of context is indispensable in every context-aware services development proposal; where the dynamic context requires an adaptation at execution time for quick responses without any human intervention. A CAS being executed should be adapted automatically at runtime if a context change occurs, mainly in real-time systems where there is no time to lose (healthcare, air traffic ...). For example, the cancellation of a flight (as a context-aware adaptation) if there is a strong snowfall (as a context information). In consequence, the adoption of an adaptation strategy at runtime becomes crucial in CAS life-cycle.

To improve development of context-aware services' adaptation, our ongoing work is to propose a Model-Driven Approach taking advantage of combining Model-Driven Development (MDD) [5] and Aspect-Oriented Modelling (AOM, `http://www.aspect-modeling.org/`). MDD simplifies designing and developing of such services piloted by models, which can be converted to other models or codes by transformation techniques, instead of writing them by hand, which is a daunting and error-prone task. Furthermore, AOM allows to achieve the context-awareness logic by weaving context-

ICCASA 2014, October 15-16, Dubai, United Arab Emirates

DOI 10.4108/icst.iccasa.2014.257300

aware aspect models (ContextAspect) at design and run time into the core application model. This approach involves the following phases (modelling, composition, transformation, adaptation) constituting its development process.

In this work, we aim to more highlight the adaptation phase functioning by implementing an illustrative example on FraSCAti platform [18] that supports the adaptation at runtime of context-aware services.

The remainder of this paper is organized as follows. Section 2 presents briefly our MDA-based approach using ContextAspect models. By a descriptive example, Section 3 details our proposed dynamic adaptation process of CASs at runtime. Section 4 discusses some related works, and Section 5 concludes this paper by indicating the future work.

2. MDA-BASED DEVELOPMENT APPROACH

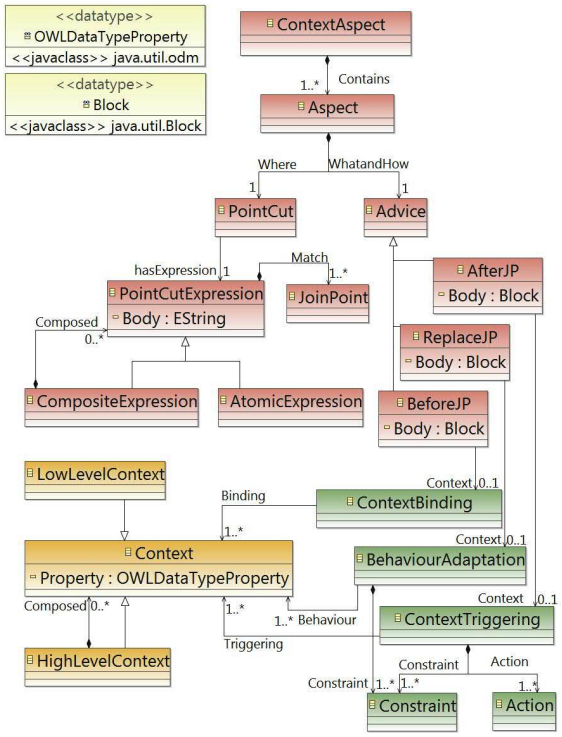

Figure 1: ContextAspect MetaModel

This section introduces with a concise manner our MDA-based approach to develop and adapt CASs. Figure 2 shows its model-driven architecture that is founded on ContextAspect models to deal with context-aware adaptation as a crosscutting concern. The `ContextAspect` metamodel (Figure 1) aims to define and specify where and how the context-aware adaptation logic will take place. A `ContextAspect` model should contain one or more context-aware aspects where each one is composed of :

- Aspect elements (red coloured): to provide where (pointcut), what and how (advice) elements requisite to apply a context-aware adaptation using the concepts of aspect-oriented programming (AOP) [8].
- Context elements (yellow coloured): for representing the execution context that surrounds the service with an ontology formalism according to the ODM specification [4].
- Context-Awareness elements (green coloured): are for relating Aspect and Context elements by three context-awareness mechanisms: ContextBinding, BehaviourAdaptation and ContextTriggering [20].

The proposal approach focuses on considering that a context-aware service can be seen as an application with several sensitive elements to context (as variation points). For each one, a multiple alternative ContextAspect models (as variants) are associated in order to select one of them according to the context and to weave it by AOM techniques.

This approach begins by designing a first context-aware application adaptable to other applications at execution time according to the dynamic context change. It comprises four phases, namely:

1. Modelling phase: consists to provide different UML-based models in accordance with their metamodels which will be included in the CAS development (Ontology-based Context model, Platform Independent Model or PIM, and ContextAspect models).
2. Composition phase: is for composing the PIM with the selected ContextAspect, and it returns a Contextual Platform Independent Model (CPIM) using weaving techniques into Model to Model Transformation.
3. Transformation phase: firstly and by an M2M transformation, the CPIM is converted to a Contextual Platform Specific Model (CPSM) bounded to SCA-based platform. Secondly, the obtained CPSM will be coded by a Model to Text Transformation for generating FraSCAti platform codes [18].
4. Adaptation phase: Once created and deployed on a service-oriented platform, the produced application can then be subject to several dynamic adaptations according to context change. The next section details our contribution in this phase by presenting a dynamic process to accomplish these adaptations.

3. DYNAMIC ADAPTATION PROCESS AT RUNTIME

Before to give the adaptation process to be followed dynamically by context-aware services at execution time, we introduce an illustrative example for unrolling steps of this process.

3.1 Illustrative Example

The following example `ConferenceGuide` (`CG`) is intended to facilitate the organization of academic meetings (conference, seminar, workshop, ...), by giving some necessary informations/services to attendees (participants and organizers) about their scientific stays such as: to join the conference, to know the conference program and to participate in social events. `CG application` is a context-aware application that displays a GUI through which participants or organizers may use three services: `JoiningConference, SocialEvents and Program`. This latter gives the whole communications program for an organizer, whereas it gives only the communications related to the interest centre of a participant.

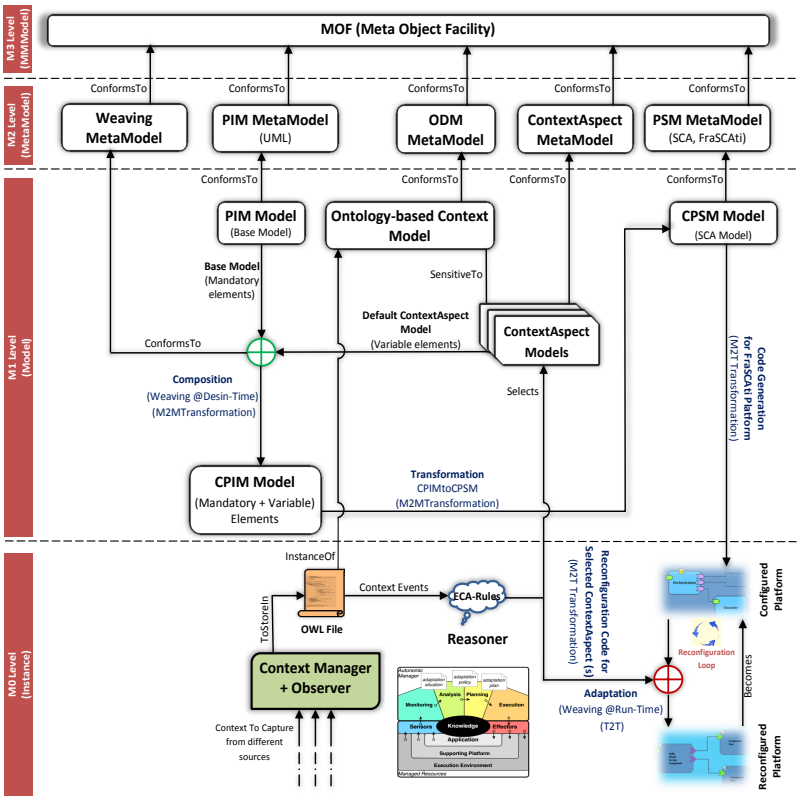

Figure 2: MDA-based Development Approach for CASs

To illustrate the dynamic adaptation process which will be proposed, we will unroll a simple scenario on this `Program` service considering that it is invoked by a conference organizer, which means, the conference program will be displayed. Nonetheless, and at a given moment, a contextual event is observed indicating the absence of a participant (not yet registered) to present his communication, and we will see how the service in question is going to behave.

3.2 Dynamic Adaptation Process

Bobrow et al. [3] have defined the reflection as: "*The ability of a program to manipulate as data something representing the state of the program during its own execution. There are two aspects of such manipulation:* ***introspection*** *and* ***intercession****. Introspection is the ability of a program* ***to observe and therefore reason about its own state****. Intercession is the ability of a program* ***to modify its own execution state*** *or alter its own interpretation or meaning*".

To exploit this reflection principle to realize the dynamic adaptation in our paper's scope, the context-aware service can be considered as an autonomic system with the MAPE-K control loop which consists to monitoring, analysis, planning and execution activities by sharing knowledge between them [13] (Figure 2, M0 level). Currently, the MAPE-K suggested by IBM in 2003 represents the reference model for many self-adaptive software development works like in [23].

3.2.1 Monitoring

The Context Manager equipped with Observer module can sense any significant change in context acquired from software (web services for example) or hardware (camera, GPS ...) sensors. The Observer can be realized by Complex Event Processing (CEP) system [14]. We consider that the CEP technology could continuously control and process the confluence and fluctuation of the context in order to refine it and to store the proper context in OWL file derived from the proposed ontology-based context model [4]. We believe that the combination WildCat `http://wildcat.ow2.org/`/Esper `http://esper.codehaus.org/` frameworks can be solicited to act as Manager and Observer modules. In this work, we have not more focused on the acquisition of context but about its modelling and how improving the adaptation based on it.

For our example, the OWL file contains `Registered` datatypeproperty which value is `False` less than 30 minutes before beginning the communication of `Author6` participant (TimeToRegister<=30 and Registered=False). This is can lead to cancel his communication as a high-level context information deduced by the following SWRL rule : Participant(?Author6) and IsRegistered(?Author6, ?Registered) and swrlb:Equal(?Registered, False) and HasTime(?Author6, ?TimeToRegister) and swrlb:lessThanOrEqual(?TimeToRegister, 30) implies CommunicationCancelled(?Author6, True). Here, the change in time is considered as a trigger event to initiate the dynamic adaptation process.

3.2.2 Analysis

The new values of context will be analysed by a Reasoner that is an inference engine based on ECA rules. The general syntax of ECA rule is **"on Event if Condition do Ac-**

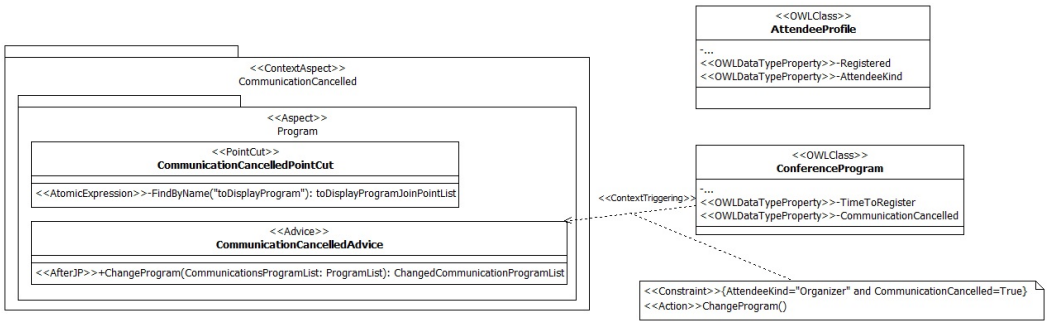

Figure 3: CommunicationCancelled ContextAspect Model

tion". The event part specifies the signal that triggers the rule invocation. In our case, the event is the context change sensed by the Observer. The condition part is a logical test that, if satisfied, causes an action to be carried out. It is one or more constraints on context values. The action part consists of updates or invocations on the local data. Here, it implies the selection of necessary adaptations encapsulated in ContextAspect model. ECA rules are used to select the suitable ContextAspect model to a context situation in order to weave it in application model. Our ECA rules can be established from constraints ("`Constraint`" class) designed in ContextAspects models (see Figure 1). These ECA rules can be implemented and checked by a Drools Engine `http://www.jboss.org/drools` by transforming the event part to "When" clause and both parts of condition and action to "Then" clause. In our case, we have a Drool rule that decides to weave `CommunicationCancelled` ContextAspect model (Figure 3) for the communication cancellation contextual case to different reasons (no registration, for example). This ContextAspect proposes to the organizer to modify the conference program accordingly.

3.2.3 Planning

The selected ContextAspect model will be converted by an M2T transformation to generate FPath and FScript codes [7], requisite to reconfigure the running application on FraSCAti platform. The `pointcut` expressions of this model are transformed into FPath code allowing the navigation in the architecture of the running FraSCAti-based applications to search the impacted weaving elements. Furthermore, the `advice` is transformed into FScript code which is a scripting language dedicated to architectural reconfiguration of such applications [15].

Generally, a reconfiguration (or dynamic adaptation) consists of two main steps: (1) to find the context-aware places matched by pointcut through FPath code, and (2) to weave the modifications expressed in advice and coded by FScript. The modifications' weaving will be planned in actions conforming to a Weaving metamodel and will be directed by an algorithm and rules. For each place impacted by FPath, this weaving should go through the following actions: (i) stopping the running FraSCAti components (or composite), (ii) removing all components and wires relevant to the advice script of the ContexAspect model already woven (if any), (iii) adding components and wires for weaving the script of the new advice element, and finaly (iv) restarting the stopped components.

In other hand and at the same time, an adaptation at model level should be made. Again, the composition phase (Section 2) is called considering that the running CPIM model becomes PIM model for accepted to be composed with the ContextAspect model selected in this adaptation phase. The dynamic adaptation at model level can be enabled using models@runtime [2] in order to establish a causal connection between the model and the running application. Also, this adaptation can serve to validate the reconfigured context-aware service by automatic CPIM to CPSM to code transformations if they will be available.

Applying on the `CG` example and by an M2T transformation, the selected ContextAspect model (`CommunicationCancelled`) will be converted to FPath and FScript codes for generating the needed reconfiguration scripts to be applied on the `Program` service. The pointcut expression is transformed into FPath code (toDisplayJP=$root/descendant-or-self::*[name(.)=='toDisplayProgram']) which selects `toDisplayProgram` Service Component to be adapted without affecting `JoiningConference` and `SocialEvents` services which continue their execution. As well and using weaving actions, the advice is transformed into FScript code as a reconfiguration script for weaving `CommunicationCancelled` ContextAspect. In Figure 4, the script listing implements a weaving operation. Firstly, it brings the application in a quiescent state (line 2), and then creates an instance of the `ChangeProgramServiceComponent` (line 4, which behaviour is described in advice). This new component described by `<<AfterJP>>` advice is included as supercomponent by the `addFcSuperComponent` primitive (respectively, `addFcSubComponent` for `<<BeforeJP>>` advice) in the application architecture (line 5) and wired to the `toChangeProgramServiceComponent` (lines 6-7). Finally, the service `ChangedCommunicationProgramList` is promoted to the enclosing composite (line 8) and exposed as a SOAP binding in order to be discovered and connected as web service (line 9). When these actions are completed, the execution of the application can be resumed (line 10). For unweaving action, can use `RemoveFcSuperComponent`, `RemoveFcSubComponent` and `UnbindFC` primitives [18, 7].

3.2.4 Execution

Thanks to FraSCAti platform (plugged in Eclipse `https://www.eclipse.org/`) which supports the weaving at runtime and enables the dynamic adaptation of applications being executed [18]. Firstly, and before adaptation (at 10:00 am), `Program` service displays the whole list of communica-

```
1 action WeaveAfterJP(ProgramApplication, CommunicationCanceledAdvice){
2  stop($ProgramApplication);
3  // supercomponent of totoDisplayProgramServiceComponent
4  ChangeProgramServiceComponent = adl-new($CommunicationCanceledAdvice);
5  addFcSuperComponent($ProgramApplication, $ChangeProgramServiceComponent);
6  bindFc($ChangeProgramServiceComponent/interface:: CommunicationsProgramList,
7  $toDisplayProgramServiceComponent/interface:: CommunicationsProgramList);
8  promote($ChangeProgramServiceComponent/interface:: ChangedCommunicationProgramList, $ProgramApplication);
9  bind($ChangeProgramServiceComponent/interface:: ChangedCommunicationProgramList, "soap");
10  start($ProgramApplication);
11 return $ProgramApplication;}
```

Figure 4: Reconfiguration FScript code for Adapted Program Service

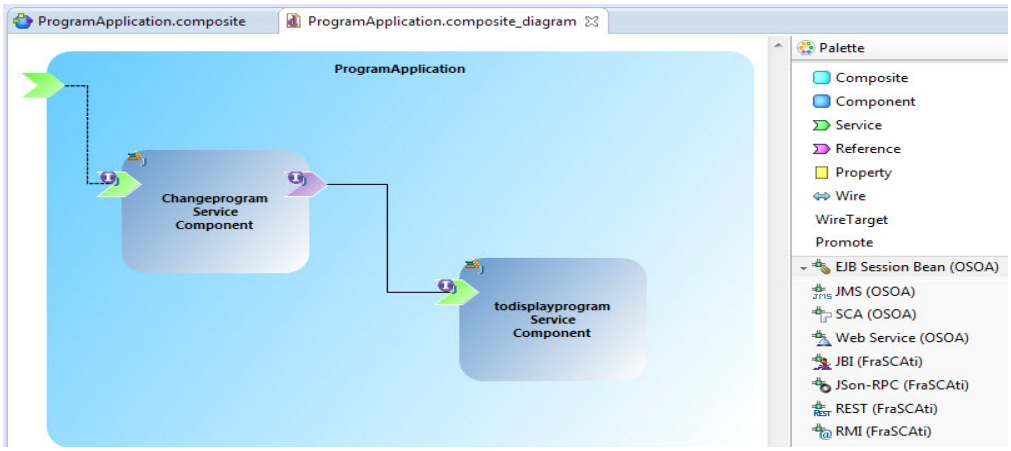

Figure 5: SCA Architecture of Adapted Program Service

tion program either `Presented` or `Programmed` (status column) as shown in Figure 6 (Sreen 1).

Then, and after weaving `CommunicationCancelled` ContextAspect, the new SCA architecture of `Program` service is as depicted in Figure 5.

And, after restarting the adapted `Program` service (at 10:02 am), the communication program list will be displayed with `Cancelled` status for `Author6` communication and the below communications in list are advanced in time (atTime column) by informing their authors (Figure 6, Screen 2).

4. RELATED WORK

The software adaptation in mobile and ubiquitous computing has been treated by several works in the literature [11]. This work is focused on model-driven approaches proposed for developing and adapting context-aware services. The viewpoint of context-aware adaptation as a crosscutting concern in core service-based application is not new. It has been used by some works at model and/or code levels. To the best of our knowledge, [6] and [9] are only two works that combine MDD with AOM to develop context-aware applications at model level. However, a few works are proposed at code level. They have combined AOP and MDD to tackle the complexity of context-aware applications development. In [16], Prezerakos et al. proposed to deal core service logic and context-aware adaptation as separate concerns using a modified version of ContextUML [19] and aspects to encapsulate context-dependent behaviour in discrete AspectJ code modules. Tanter et al. in [21] presented the contextual adaptation in what is called context-aware aspects. Otherwise, there are other works which have based on MDD to handle context-aware services development. Sheng et al. [19] proposed ContextUML metamodel which extends UML syntax to introduce appropriate artefacts enabling the creation of context-aware service models. Ayed and Berbers [1] have presented an UML metamodel that supports context-aware adaptation of service design from structural, architectural, and behavioural perspectives. The Kapitsaki et al. approach [12] has proposed a context adaptation architecture of web services composition and a model-driven methodology for the development of such context-aware composite applications.

The approaches of [6, 21, 19, 1] are generally limited to the static aspect of context at design time and ignore completely its dynamic aspect. They do not take into account the dynamic change of context during the application execution and consequently no adaptation plan at runtime is expected in their proposals. However, the works [9, 16, 12] have treated this kind of adaptation at code level without a clear and generic dynamic process. However, In [10], the authors propose an approach to develop and to evolve context-aware adaptive services at model level which focuses on models@runtime concept. This approach is built on a specific adaptation process.

Also, all cited approaches are not based on a standard feedback loop such as MAPE-K used in ours. The main advantage of the present adaptation process is to allow the self-reconfiguration of context-aware service-based applications at code level in accordance with model level by weaving successive adaptations to context-aware services being executed depending on context change over time.

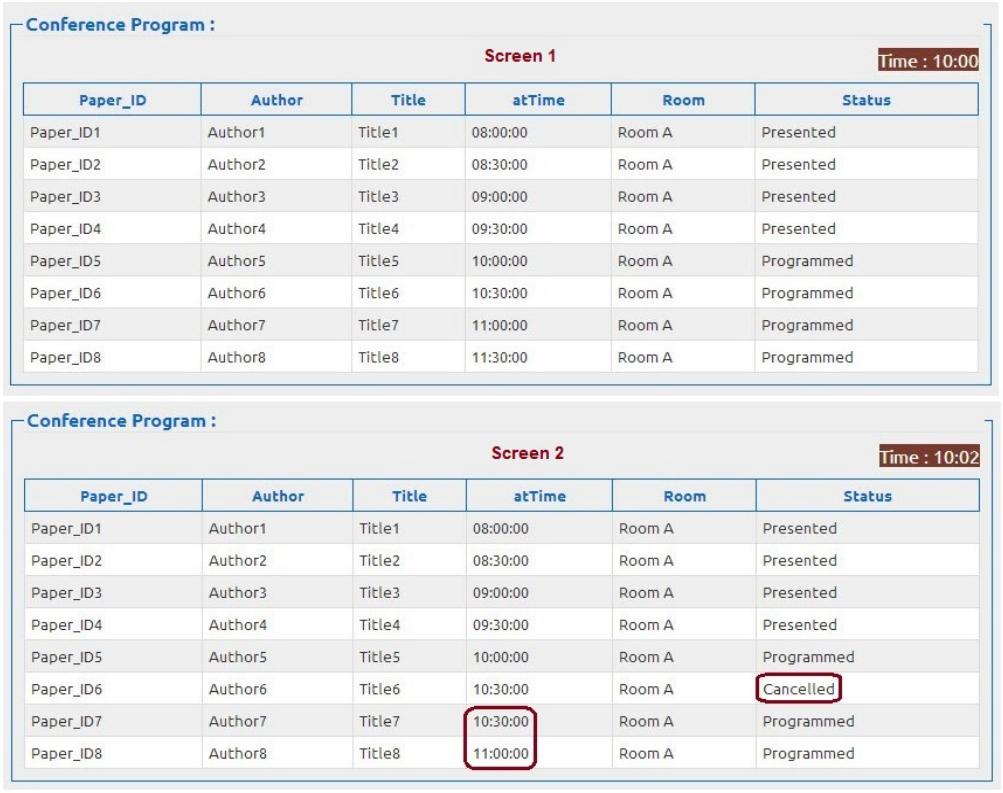

Figure 6: Communications Program List Before and After Adaptation

5. CONCLUSION

Context-aware services are a promising technology to construct customised ubiquitous applications. Besides the importance of how to build such services, their context-aware adaptation during the execution time represents a challenge to overcome. The present article proposed a model-driven dynamic adaptation process at runtime respecting the MAPE-K control loop. This process consists to weave at model and code levels necessary adaptations specified in ContextAspect models into the running application according to dynamic context change. The aspect weaving enables to produce a wide range of context-aware services models without designing them from the beginning. An illustrative example is implemented using the FraSCAti platform so smoothly achieving this runtime adaptation. Our future work will be on evaluating the execution time taken by this process. We will also focus on reducing the adaptation's response time so improving the context-awareness performed for such kind of services.

6. REFERENCES

[1] Ayed, D., and Berbers, Y. 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 2006* (Melbourne, Australia, December 2006), I. Gorton, L. Zhu, Y. Liu, and S. Chen, Eds., vol. 183 of *ACM International Conference Proceeding Series*, ACM, pp. 1–5.

[2] Blair, G., Bencomo, N., and France, R. B. Models@ run.time. *Computer 42* (2009), 22–27.

[3] Bobrow, D. G., Gabriel, R. P., and White, J. L. Clos in context: the shape of the design space. 29–61.

[4] Boudaa, B., Hammoudi, S., and Chikh, M. A. ODM-Based modeling for User-Centered Context-Aware mobile applications. In *The 3rd International Conference on Information Technology and e-Services ICITeS'2013(ICITeS'2013)* (Sousse, Tunisia, Mar. 2013).

[5] Brambilla, M., Cabot, J., and Wimmer, M. *Model-Driven Software Engineering in Practice.* Morgan & Claypool, Sept. 2012.

[6] Carton, A., Clarke, S., Senart, A., and Cahill, V. Aspect-oriented model-driven development for mobile context-aware computing. In *SEPCASE '07: Proceedings of the 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments* (Washington, DC, USA, 2007), IEEE Computer Society, p. 5.

[7] David, P.-C., Ledoux, T., Coupaye, T., and Léger, M. Fpath and fscript: Language support for navigation and reliable reconfiguration of fractal architectures. *Annales des Télécommunications 64*, 1-2 (Février 2009), 45–63.

[8] Elrad, T., Filman, R. E., and Bader, A. Aspect-oriented programming: Introduction. *Commun. ACM 44*, 10 (Oct. 2001), 29–32.

[9] Grassi, V., and Sindico, A. Towards model driven design of service-based context-aware applications. In *Proceedings of the 2007 International Workshop on Engineering of Software Services for Pervasive Environments, ESSPE 2007* (Dubrovnik, Croatia, September 2007), A. L. Wolf, Ed., ACM, pp. 69–74.

[10] Hussein, M., Han, J., Yu, J., and Colman, A. Enabling runtime evolution of context-aware adaptive services. In *IEEE SCC* (2013), IEEE, pp. 248–255.

[11] Kakousis, K., Paspallis, N., and Papadopoulos, G. A. A survey of software adaptation in mobile and ubiquitous computing. *Enterp. Inf. Syst. 4*, 4 (Nov. 2010).

[12] Kapitsaki, G. M., Prezerakos, G. N., Tselikas, N. D., and Venieris, I. S. Model-driven development of composite context-aware web applications. *Information & Software Technology 51*, 8 (2009), 1244–1260.

[13] Kephart, J. O., and Chess, D. M. The vision of autonomic computing. *Computer 36*, 1 (Jan. 2003), 41–50.

[14] Luckham, D. C. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[15] Parra, C., Blanc, X., Cleve, A., and Duchien, L. Unifying design and runtime software adaptation using aspect models. *Science of Computer Programming 76*, 12 (Jan. 2011), 1247–1260.

[16] Prezerakos, G. N., Tselikas, N. D., and Cortese, G. Model-driven composition of context-aware web services using contextuml and aspects. In *Proceedings of 2007 IEEE International Conference on Web Services (ICWS 2007)* (Salt Lake City, Utah, USA, July 9-13 2007), IEEE Computer Society, pp. 320–329.

[17] Schilit, B., and Theimer, M. Disseminating active map information to mobile hosts. *IEEE Network 8* (1994), 22–32.

[18] Seinturier, L., Merle, P., Rouvoy, R., Romero, D., Schiavoni, V., and Stefani, J.-B. A component-based middleware platform for reconfigurable service-oriented architectures. *Software: Practice and Experience 42*, 5 (May 2012), 559–583.

[19] Sheng, Q. Z., and Benatallah, B. Contextuml: A uml-based modeling language for model-driven development of context-aware web services. In *Proceedings of 2005 International Conference on Mobile Business (ICMB 2005)* (Sydney, Australia, 11-13 July 2005), pp. 206–212.

[20] Sheng, Q. Z., Yu, J., Segev, A., and Liao, K. Techniques on developing context-aware web services. *IJWIS 6*, 3 (2010), 185–202.

[21] Tanter, É., Gybels, K., Denker, M., and Bergel, A. Context-aware aspects. In *5th International Symposium on Software Composition (SC 2006)* (Vienna, Autriche, 2006), W. Löwe and M. Südholt, Eds., vol. 4089 of *LNCS*, Springer.

[22] Turner, R. M. Determining the Context-Dependent Meaning of Fuzzy Subsets. In *Proceedings of the 1997 International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-97)* (Rio de Janeiro, 1997).

[23] Vogel, T., and Giese, H. Model-driven engineering of self-adaptive software with eurema. *ACM Trans. Auton. Adapt. Syst. 8*, 4 (Jan. 2014), 18:1–18:33.