On Sustaining Dynamic Adaptation of Context-Aware Services.

Boudjemaa Boudaa, Slimane Hammoudi, Bouguessa Abdelkader, Amel Leila Mebarki, Mohammed Amine Chikh

EAI Endorsed Transactions
on Context-aware Systems and Applications
Research Article ICST.ORG

On Sustaining Dynamic Adaptation of Context-Aware Services

Boudjemaa Boudaa[1,*], Slimane Hammoudi[2], Abdelkader Bouguessa[1], Leila Amel Mebarki[1], and Mohammed Amine Chikh[3]

[1]Département d'Informatique, Université Ibn Khaldoun, Tiaret, Algérie
[2]Equipe MODESTE, Groupe ESEO, Angers, France
[3]Département d'Informatique, Université Abou Bekr Belkaid, Tlemcen, Algérie

Abstract

The modern human is getting more and more mobile having access to online services by using mobile cutting-edge computational devices. 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 a model-driven development life-cycle for context-aware services using the MAPE-K control loop to meet the runtime adaptation. This process is validated by implementing an illustrative application on FraSCAti platform. The main benefit of the proposed process is to sustain the self-reconfiguration of such services at model and code levels by enabling successive dynamic adaptations depending on the changing context.

Keywords: Context-Aware Service, Dynamic Adaptation, Model-Driven Development, MAPE-K.
Received on 15 December 2014, accepted on a6 December 2014, published on 12 March 2015

doi: 10.4108/casa.2.3.e4

1. Introduction

Actually, the tendency is to use more applications and services offered on mobile computing cutting-edge devices for managing people's professional and social lives. Mobile and ubiquitous environments are featured by the richness of context information that increases the need to personalise and adapt services accordingly. This requires considering how to cope the context changes in the mobile services development. These services should be context-aware [1], [2] using information about their execution context for responding and adapting to changes in different computing situations. 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) [3], therefore the consideration of both kinds of context is crucial 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 very essential 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) [4] 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 proposed context-aware aspect models (called ContextAspect) at design and run time into

*Corresponding author. boudjemaa.boudaa@univ-tiaret.dz

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 [5] that supports the adaptation at runtime of context-aware services.

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

2. Related Work

The software adaptation in mobile and ubiquitous computing has been treated by several works in the literature [6]. 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, [7] and [8] 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 [9] and MDD to tackle the complexity of context-aware applications development. In [10], Prezerakos et al. proposed to deal with core service logic and context-aware adaptation as separate concerns using a modified version of ContextUML [11] and aspects to encapsulate context-dependent behaviour in discrete AspectJ code modules. Tanter et al. in [12] 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. [11] proposed the ContextUML metamodel which extends UML syntax to introduce appropriate artefacts enabling the creation of context-aware service models. Ayed and Berbers [13] have presented an UML metamodel that supports context-aware adaptation of service design from structural, architectural, and behavioural perspectives. The Kapitsaki et al. approach [14] 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 [7], [12], [11], [13] 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 [8], [10], [14] have treated this kind of adaptation at code level without a clear and generic dynamic process. However, in [15], the authors propose an approach to develop and evolve context-aware adaptive services at model level, which focuses on models@runtime concept [16]. This approach is built on a specific adaptation process.

In addition, 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.

3. An MDA-Based Approach for CASs

This section introduces in a concise manner our ongoing approach which is based on Model-Driven Architecture (MDA) [17] 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) [9].
- Context elements (yellow coloured): for representing the execution context that surrounds the service with an ontology formalism according to the ODM specification [18].

- Context-Awareness elements (green coloured): are for relating Aspect and Context elements by three context-awareness mechanisms: ContextBinding, BehaviourAd-aptation and ContextTriggering [19], [20].

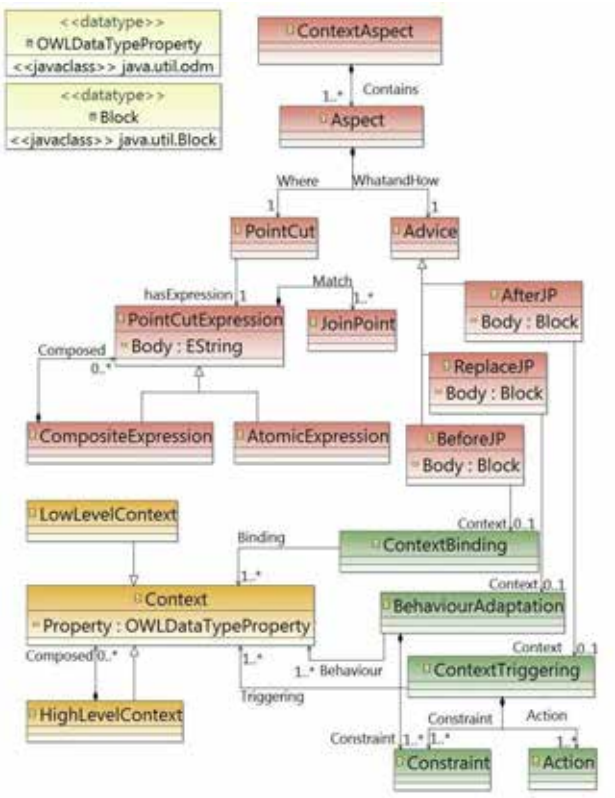

Figure 1. ContextAspect MetaModel

The proposal approach focuses on considering that a context-aware service can be seen as 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:

(i) **Modelling phase**: consists to provide different UML-based models [21] 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).

(ii) **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 (M2M) Transformation.

(iii) **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 (M2T) Transformation for generating FraSCAti platform codes [5].

(iv) **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.

4. Dynamic Adaptation Process

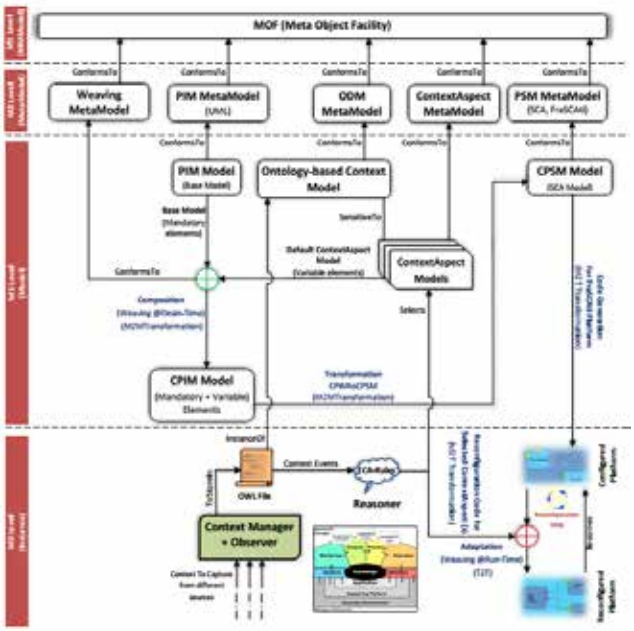

Figure 2. MDA-based Development Approach for CASs

Before giving 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.

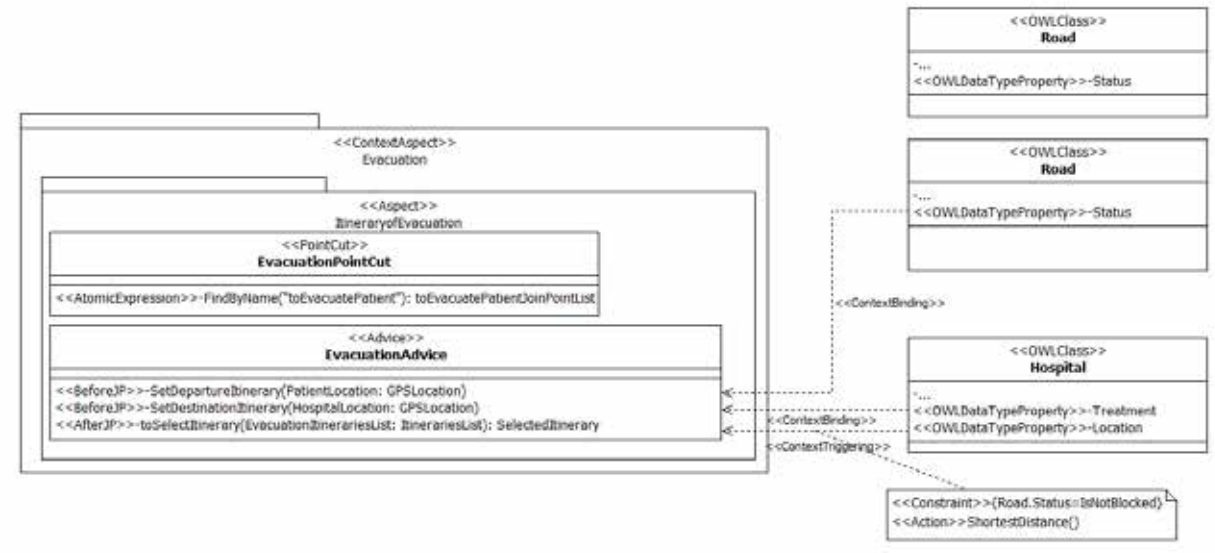

Figure 3. Evacuation ContextAspect Model

4.1. Descriptive Application

The following example illustrates the way we provide urgent medical services to patients in need, as the SAMU service in France*. It consists to receive patient's call and to evacuate him/her to the nearest hospital according to his/her chronic disease. The application of this example can be achieved by standing on the SCA architecture, as the described below (Figure 4). The application contains the needed components in order to accomplish the SAMU services.

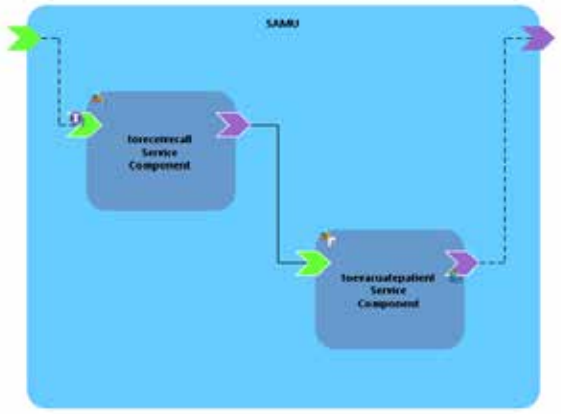

Figure 4. SAMU Application

Using our MDA-based approach, the SAMU application can be seen as a context-aware application. Figure 5 shows the context-aware service of Evacuation component built by weaving the context-awareness logic included in `Evacuation ContextAspect` (Figure 3). The building of context-aware application by the present approach is out of scope for this paper and will be detailed in other work. Here we are interested only by exhibiting the adaptation phase of this approach.

Now, the context-aware Evacuation service can acts without human intervention by deducing the itinerary source from the patient call's location using the GPS technology, and the hospital destination from disease information found in the patient profile. Those two context parameters would help the service to select the appropriate itinerary with shortest distance from several proposed ways.

To illustrate the dynamic adaptation process, which will be proposed, we will unroll a simple scenario on this Evacuation service considering that it is invoked, and the itinerary is selected and displayed to evacuate a patient. Nonetheless, and after traffic on this itinerary, a contextual event is observed indicating that the selected way is crossed out because of car accident or unforeseen work. In what follows we will see how this service in question is going to behave.

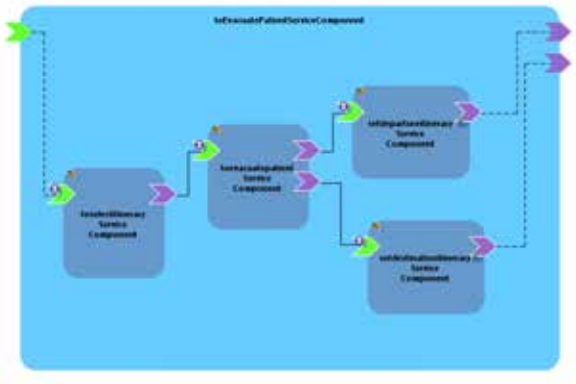

Figure 5. Context-aware Evacuation Service

* http://www.aphp.fr/urgence/le-centre-15/

4.2. Dynamic Adaptation Process

Bobrow et al. [22] 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 for realizing 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 [23] (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 [24].

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 [25]. 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 [18]. We believe that the combination WildCat http://wildcat.ow2.org//Esper and 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 `Status` datatypeproperty which value is switched from "`IsNotBlocked`" to "`IsBlocked`" less for the selected itinerary. This is considered as a trigger event to initiate the dynamic adaptation process in order to change the itinerary of evacuation.

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 Action".** 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 ContextAspect model (see Figure 1). These ECA rules can be implemented and checked by a Drools Engine `http://` `http://www.drools.org/` by transforming the event part to "When" clause and both parts of condition and action to "Then" clause. In our case, we have a Drools rule that decides to weave `Evacuation` ContextAspect model once again for application adaptation (Figure 3) after unweaving the former in order to change the previous itinerary of evacuation that is blocked for different reasons (for car accident, unforeseen work or congestion road, for example). This ContextAspect proposes for the ambulance a new GPS itinerary accordingly to this situation of context. Other ContextAspect models can be woven for other contextual situations for other application examples [26].

Planning

The selected ContextAspect model will be converted by an M2T transformation to generate FPath and FScript codes [27], 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 [28].

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 that will not be presented here. For each place impacted by

FPath, this weaving should go through the following actions:

1. stopping the running FraSCAti components (or composite);
2. removing all components and wires relevant to the advice script of the ContexAspect model already woven (if any);
3. adding components and wires for weaving the script of the new advice element;
4. and finally, restarting the stopped components.

On the other hand and at the same time, an adaptation at model level should be made. Again, the composition

```
1 action WeaveEvacuation(EvacuationApplication, EvacuationAdvice){
2  stop($EvacuationApplication);
3  // subcomponent of SetDepartureItineraryServiceComponent
4  SetDepartureItineraryServiceComponent = adl-new($SetDepartureItineraryAdvice);
5  addFcSubComponent($EvacuationApplication, $SetDepartureItineraryServiceComponent);
6  bindFc($toEvacuatePatientServiceComponent/interface:: DepartureItinerary,
7  $SetDepartureItineraryServiceComponent/interface:: DepartureItinerary);
8  // subcomponent of SetDestinationItineraryServiceComponent
9  SetDestinationItineraryServiceComponent = adl-new($SetDestinationItineraryAdvice);
10  addFcSubComponent($EvacuationApplication, $SetDestinationItineraryServiceComponent);
11  bindFc($toEvacuatePatientServiceComponent/interface:: DestinationItinerary,
12  $SetDestinationItineraryServiceComponent/interface:: DestinationItinerary);
13  // supercomponent of toSelectItineraryServiceComponent
14  toSelectItineraryServiceComponent = adl-new($toSelectItineraryAdvice);
15  addFcSuperComponent($EvacuationApplication, $toSelectItineraryServiceComponent);
16  bindFc($toSelectItineraryServiceComponent/interface:: EvacuationItinerariesList,
17  $toEvacuatePatientServiceComponent/interface:: EvacuationItinerariesList);
18  promote($toSelectItineraryServiceComponent/interface:: ItinerarySelected, $EvacuationApplication);
19  bind($toSelectItineraryServiceComponent/interface:: ItinerarySelected, "soap");
20  start($EvacuationApplication);
21 return $EvacuationApplication;}
```

Figure 6. Reconfiguration FScript code for Adapted Evacuation Service

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 [16] 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 `SAMU` example and by an M2T transformation, the selected `Evacuation` ContextAspect model will be converted to FPath and FScript codes for generating the needed reconfiguration scripts to be applied on the `toEvacuatePatient` service. The pointcut expression is transformed into FPath code (toDisplayJP=$root/descendant-or-self::*[name(.)=='toEvacuatePatient']) which selects `toEvacuatePatient` Service Component to be adapted without affecting `toReceiveCall` one, which continues its execution. Using weaving actions, the advice is as well transformed into FScript code as a reconfiguration script for weaving `Evacuation` ContextAspect. In Figure 6, the script listing implements a weaving operation:

Firstly, it brings the application in a quiescent state (line 2), and then creates an instance of three service components, namely: `SetDepartureItinerary` (line 4), `SetDestinationItinerary` (line 9) and `toSelectItinerary` (line14). The two first components are described by `<<BeforeJP>>` advice (see Figure 3), where they are included in the application architecture as subcomponent by `addFcSubComponent` primitive (lines 5 and 10) and wired to `toEvacuatePatientServiceComponent` (lines 6-7 and 11-12). However, the last new component is included as subcomponent by `addFcSuperComponent` primitive (line 15) for `<<AfterJP>>` advice (Figure 3) and wired to `toEvacuatePatientServiceComponent` (lines 16-17). Finally, the service `ItinerarySelected` is promoted to the enclosing composite (line 18) and exposed as a SOAP binding in order to be discovered and connected as web service (line 19). When these actions are completed, the execution of the application can be resumed (line 20). For unweaving action, can use `RemoveFcSuperComponent`, `RemoveFcSub-Component` and `UnbindFC` primitives [5], [27].

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 [5]. the adaptation process can be done so efficiently. Firstly, and before adaptation (at 15:30 pm, for example), `toEvacuatePatient` service displays the whole list of itineraries either `Blocked` or `NotBlocked` (status column) with a map of the shortest and selected itinerary to be displayed on Ambulance's GPS as shown in Figure 7 (Screen 1).

Then, and after a change in status (as context information) of the selected itinerary (from `NotBlocked` to `Blocked`, at 15:45 pm), an adaptation of `toEvacuatePatient service will` occur taking account the new departure of the itinerary (i.e. where it received the new context information about the blocking of the previous itinerary) and then, it selects other new itineraries list with new distances to the same destination. Finally, the system will show the new map for the new selected itinerary to be followed by the Ambulance driver (Figure 7, Screen 2) for a rapid evacuation of the patient in order to save his/her life.

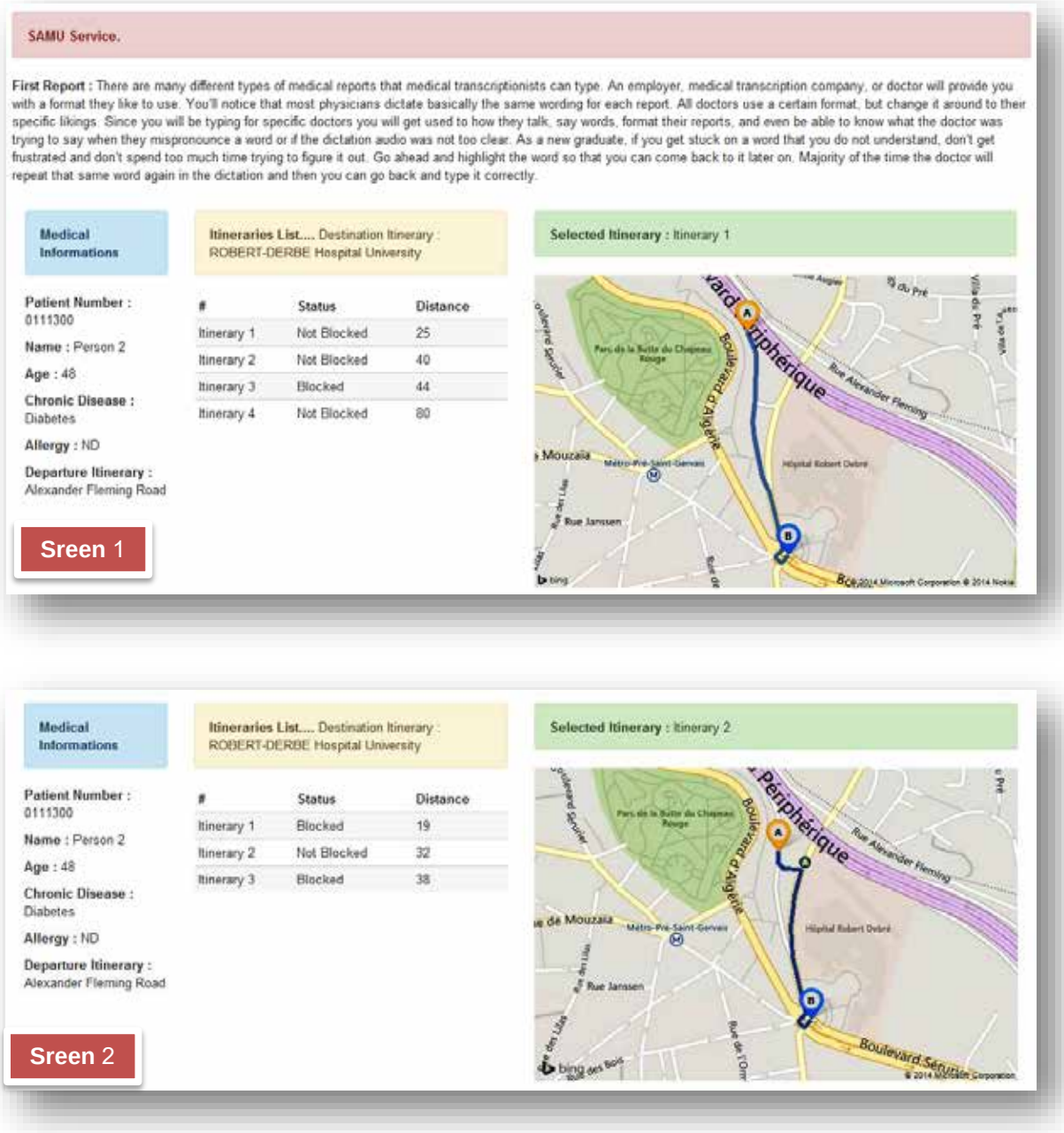

Figure 7. Evacuation Itineraries Before and After Adaptation

Note that the maps in this example are obtained by using the Bing Maps REST Services http://msdn.microsoft.com/en-us/library/ff701713.aspx.

5. Conclusion

Context-aware services are a promising technology to construct personalised pervasive 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 case study 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.

References

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

[2] Kapitsaki, G. M., Prezerakos, G. N., Tselikas, N. D., 2010. In book "Enabling Context-Aware Web Services: Methods, Architectures, and Technologies", 1st Edition. Chapman & Hall/CRC Press, Ch. Context-aware Web Service Development: Methodologies and Approaches, pp. 3–29.

[3] 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).

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

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

[6] 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).

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] Tanter, E., Gybels, K., Denker, M., and Bergel, A. Context-aware aspects. In 5th International Symposium on Software Composition (SC 2006) (Vienna, Autriche, 2006), W. L• owe and M. S• udholt, Eds., vol. 4089 of LNCS, Springer.

[13] Ayed, D., and Berbers, Y. Uml pro le 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.

[14] 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.

[15] 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.

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

[17] OMG, 2003. Model driven architecture (mda), mda guide version 1.0.1.URL http://www.omg.org/cgi-bin/doc?omg/03-06-01.

[18] 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).

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

[20] Boudaa, B. Towards a Model-Driven Requirements Specification of Context-Aware Services. The 10th International Conference on Signal Image Technology & Internet - Based Systems (SITIS'14) - Track On Web

Computing And Applications (WeCA), November 23 - 27, 2014, Marrakech, Morocco.
[21] OMG, August 2011. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1.URL http://www.omg.org/spec/UML/2.4.1
[22] Bobrow, D. G., Gabriel, R. P., and White, J. L. Clos in context: the shape of the design space. 29-61.
[23] Kephart, J. O., and Chess, D. M. The vision of autonomic computing. Computer 36, 1 (Jan. 2003), 41-50.
[24] 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.
[25] 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.
[26] Boudaa, B., Hammoudi, S., Bouguessa, A., Chikh, M. A., 2014. Supporting runtime adaptation of context-aware services. In: Proceedings of 3rd International Conference on Context-Aware Systems and Applications, ICCASA 2014.
[27] David, P.-C., Ledoux, T., Coupaye, T., and L eger, M. Fpath and fscript: Language support for navigation and reliable recon guration of fractal architectures. Annales des Telecommunications 64, 1-2 (Fevrier 2009), 45-63.
[28] 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.