Achieving Flexible Process Interoperability in the Homecare Domain through Aspect-Oriented Service Composition

Duc Viet Bui, Maria Eugenia Iacob, Marten Van Sinderen, Alireza Zarghami

Achieving flexible process interoperability in the homecare domain through aspect-oriented service composition

Duc Viet Bui[1], Maria Eugenia Iacob[2], Marten van Sinderen[2], Alireza Zarghami[2]
[1]Cape Groep
[2]Centre for Telematics and Information Technology,
University of Twente
Enschede, The Netherlands
Emails: d.bui@capegroep.nl; {m.e.iacob; m.j.vansinderen; a.zarghami}@utwente.nl

Abstract. In elderly care the shortage of available financial and human resources for coping with an increasing number of elderly people becomes critical. Current solutions to this problem focus on efficiency gains through the usage of information systems and include homecare services provided by IT systems. However, the current IT systems that integrate homecare services have difficulties in handling the user-context dynamicity and the diversity of needs and preferences of care-receivers. This makes the available homecare services hardly interoperable at the process level, particularly due to the lack of support for process flexibility. In this paper, we present an approach capable of dealing with such interoperability issues based on aspect-oriented service composition. We demonstrate the feasibility of our approach and of the proposed architecture by implementing a prototype for a reminder service scenario.
Keywords: process interoperability, process flexibility; aspect-oriented service composition, homecare services, orchestration

1 Introduction

European countries are experiencing a rapidly growing number of elderly people. According to European Union's Health portal, by 2050, "the number of people aged 65 and above is expected to grow by 70% and the number of people aged over 80 by 170%" [1]. Consequently, healthcare systems are under pressure to address the increasing demands for elderly care. Information systems offering and integrating IT homecare-services for elderly [2] are believed to have the potential of reducing healthcare costs by supporting independent living of elderly in their own home.
There are already several providers of commercial services for remote monitoring services, such as bio-signals monitoring (e.g., blood-pressure, heart-rate and oximetry), and contextual information services (e.g., location and temperature). However, these services fail to deliver the expected benefits since they are to a large extent offered and used as isolated services. In addition, such services cannot cope

with the diversity of needs and preferences of the user (e.g., in the case of a reminder service, one user may prefer a light signal to announce a reminder, while another prefers the vibration from a cell-phone), nor with the dynamicity of the user's context (e.g., change of the user's location or of the activity in which the user is engaged [3]).

Thus, an important challenge is to integrate existing homecare services through a single platform that allows user-driven service composition according to personal needs and preferences and that can automatically adapt the execution of a service composition according to the user-context at hand. We approach this challenge from the enterprise interoperability perspective [21] applied to the homecare domain [4, 24]: homecare services from different organizations should be able to work together or side by side, flexibly coordinated through an independent platform as required by needs, preferences and context. Our focus is not on syntactic or semantic interoperability, but on process interoperability. More precisely, we target *flexible process interoperability*, here defined as *the ability to coordinate different services in order to fulfill a complex user need in a dynamic environment*. User needs are complex since they require multiple services and may be context-dependent. Context-dependency arises from the dynamic nature of the environment in which the user consumes the services.

We propose an aspect-oriented service composition architecture which is based on the principles of service orientation, and is able to deal with context dynamicity and user requirements diversity [4]. We argue that the use of aspect-orientation facilitates the maintenance and management of complex processes and business rules. Also, we adhere to the design science research methodology as proposed in [19, 20].

The remainder of the paper is organized as follows. In Section 2, the reminder service scenario is presented that will serve as illustration, running example and test case for the approach and prototype we further propose in the paper. Then, a review of the existing web-service composition approaches will be presented in Section 3. In Section 4, our aspect-oriented service composition architecture will be presented. The implemented prototype described in Section 5 demonstrates the feasibility of our approach. The proposed architecture has been instantiated in a prototype, which in turn has been tested in two situations arising from the chosen scenario. Finally, in Section 6, we address the advantages and weaknesses of our approach, draw some conclusions and give some pointers to future work.

2 Scenario

To capture the dynamicity and diversity of user-context in the homecare domain, we use the following "reminder service" scenario:

Jan is an elderly person who lives in an apartment which equipped with the necessary infrastructure to support homecare applications. For example in the apartment two medicine dispensers are available, which are connected to the internet and can exchange information with our homecare service platform. Jan has to take some medication at 11:50 PM, on a daily basis. Among other things, the homecare system must remind him to take the medication. A reminder is sent to Jan three times up to 15 minutes later than the scheduled intake time. If the medication is not removed

from any of the available medicine dispensers, an alarm will be sent to the care center. Jan also has a hearing impairment and uses a wheelchair, so the doors inside the apartment open automatically. He prefers to take the medication from the closest medicine dispenser (MD) at night. Two MDs, filled with the required medication, have been installed, one in the kitchen and the other one in the bedroom. The MD inside the kitchen has embedded light. The TV installed in the bedroom, the lights in the apartment and a wristwatch can all be used as reminder devices for taking medication. However, Jan prefers not to be reminded by lights after midnight.

Linda, another care-receiver, prefers her PDA as reminders.

Nancy, as a care-giver, wants to create the desired application for both Jan and Linda. Because she understands better than IT specialists her patients' situation and requirements, she must tailor the application for both of them.

3 Background

In this section, we give first identify the requirements imposed by the homecare application domain to the future service composition architecture, and we give an brief overview of the existing web-service composition approaches.

3.1 Homecare constraints

Besides the dynamicity of user-context and the diversity of users' preferences and needs, there are three major requirements imposed on applications used in the homecare domain, namely, safety, non-intrusiveness for care-receivers and limited technical skills of care-givers and users. Safety constrains require healthcare systems to be error-free [6] due to ethical and legal considerations regarding the impact such systems may have on human lives. Hence, safety means that any reaction/behavior of the system is controllable. In other words, care-givers have to know exactly how the system behaves. The non-intrusiveness requirement refers to the fact that the system should have no impact on the normal life of care-receivers [7,8]. The limited technical skills of care-givers and care-receivers require the system to be designed such that it can be used and tailored by persons with no technical knowledge [6].

3.2 Existing web-service composition approaches

A web-service composition is "an aggregate of services collectively composed to automate a particular task or business process" [9]. According to Rao, et al. [10], there are three approaches to service compositions: static workflow-based compositions, dynamic workflow-based compositions and Artificial Intelligence (AI) planning - based compositions. In the case of the static workflow-based composition approach, a predefined process model has to be specified before the actual composition of web services takes place. Thus, in this type of composition the selection and binding of web services is realized upfront [10]. The dynamic workflow composition approach is based on the generation (at run time) of process models and selection of web services [10]. Based on logical theorem provers or on AI planners,

AI-planning approaches produce service compositions automatically without a predefined workflow [10].

3.3 Why dynamic workflow composition and Aspect-oriented approach

The specific homecare constraints influence the way in which suitable web-service compositions are determined. AI planning approaches allow the creation of the web-service composition in an automatic manner with minimum interactions with care-givers, and, thus, diminishing the impact of limited IT skills of care-givers. However, practically, it is not feasible to generate service compositions automatically in all cases with high accuracy due to the highly complex web service environment and the difficulty in capturing behavior in sufficient detail [10, 11]. Therefore, from the perspective of safety criteria, AI planning approaches have serious disadvantages due to unreliability and lack of control/predictability of the system's behavior.

Static workflow composition approaches serve safety criteria much better because the care-givers know exactly the behavior of system. However, the static approaches seriously constrain the adaptability and flexibility of the service compositions. As the result, the care-givers have to aid the system in dealing with new changes of care-receivers' needs. Furthermore, mostly such compositions will become intrusive, by forcing care-receivers to adapt to the system.

Taking the considerations above into account we have chosen the dynamic workflow composition approach because of the following reasons:

- It is based on a generated workflow. Thus the care-givers can still control the system's main activities and behavior.
- It is capable to capture changes in the user-context or the user needs, and to generate accordingly different composition workflows, by adding extra services into a predefined reference workflow. Which services are inserted into the reference workflow is determined by a set of business rules. In this way, by external changes, the rules may evaluate differently, and consequently the resulting compositions are adjusted accordingly. In this way, diversity and dynamicity are also partly supported.

There are many techniques that focus on the idea of dynamic workflow composition as described above. Inspired by aspect-oriented programming (AOP), Charfi and Menzini [12] propose an approach to externalize business rules from processes by proposing an extension of BPEL (AO4BPEL). This approach requires the modification of process engines to make it possible to handle the so-called pointcuts, advices and aspects [13].

Rosenberg and Dustdar [14] propose a Rule Interceptor Service which intercepts all incoming and outgoing Web service calls, maps them to business rules, and then applies associated business rules. A mapping document is used to map a call to business rules. Eijndhoven et al. [15] exploit the power of a business process engine (Aqualogic BPM Studio) and ILOG business rule engine. At variability points in the process, the process engine sends a request to the rule engine. Based on the input data from the request and the current context, the rule engine evaluates its business rules and the returns the result to the process engine. In [23], a tuple space has been proposed to provide more flexibility with respect to data flows. In this approach, the data can be added and shared by a process or rule engine on the fly. This approach, is somewhat similar to [15], in the sense that it also assumes that the process engine

only needs to call the rule engine at some specific variability points. In [24], the decision-making rules have been wrapped and provided as a so-called decision service which can be called by the process engine. Moreover, the decision service can notify the processes asynchronously to update their behavior.

In the homecare domain, the process specifying the behavior of the system in case of a user-context change can be very complex. For example, the process to aid care-receivers with Alzheimer disease to go out for physical exercise can be very complex and may involve many rules and activities. In addition, in order to be able to manage service compositions for care-receivers with different diseases, we need a method that can diminish the maintenance tasks. Taking this into account and the specific requirements imposed by the homecare domain, we argue that a service composition approach in should be capable 1) to execute the rules anywhere and anytime during the process instead of limiting that to some variability points and 2), to insert new services anywhere in the process (if necessary) as a reaction of system to a context change. Both requirements cannot be satisfied by approaches such as [15, 23], since both the invocation of rules and the adaptation of processes is limited to variability points. Using the Aspect-Oriented approach (AOA) by Charfi and Menzini [12] would enable us to support these capabilities. However, AOA should be applied in such a way it can be used in combination with existing implementation platforms.

4 An aspect-oriented service composition architecture

4.1 Aspect-Oriented Approach

Before presenting our architecture, we start with description of the AOA [12] and the basic constructs used in aspect orientation.

As mentioned above, in all dynamic workflow composition approaches, there are two basic elements: general workflows and business rules. More exactly, the general workflow captures the non-variable part of the service composition, and models the basic control flows, services and the dataflow. Business rules are used to represent policy-sensitive aspects of the composition, which are likely to change over time. Charfi and Menzini [12] introduce the AOA in order to separate business rules from business processes by using three elements: aspects, joint points and advices. Aspect information encoded in XML files (so-called aspect files) includes a set of joint points. A joint point is a specific point, after or before one activity in the workflow. A joint point also links to advices, which are external services or processes that have to be executed when the process execution reaches that joint point. Conditional statements can be embedded in the joint point to check the inputs taken from the general workflow and decide which advices are deployed. In a similar way, aspect information may also describe the computation rules which are applied to calculate new values of certain business process variables according to user-context changes [13]. Figure 1 shows the relation between the elements mentioned above.

It is worth noticing that an aspect is different from a business rule because besides containing a business rule, an aspect also specifies where the business rule is applied by its joint points list. Thus, one can separate business rules from specific processes and can increase the reusability of business rules.

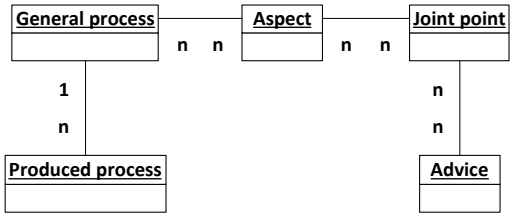

Figure 1: relationship between AOA elements

For example, the general process shown in Figure 8 involves some mandatory services, such as a service to activate the reminder and a service to check whether the medicine is taken. One possible business rule (in the form of an aspect) may concern saving user-context after each activity of the general process. When reaching the joint points (following all activities of the general process), the process executor will invoke the respective advice and save the user-context.

4.2 The proposed architecture

In this section, we propose the aspect-oriented service composition architecture that enables the interoperability and integration of homecare services. These third-party services providing information on bio-signals (heartbeat rate and blood pressure) and location are assumed to be available and can be exposed through their interfaces. It should be noted that we do not intend to design such services.

To support aspects, joint points and advices, we introduce the following three components. First, an *advice repository* is introduced to store advices. An advice, as mentioned earlier, is an external service or process which is written in BPEL and can be executed by a process engine. Second, an *aspect repository* is used to store aspect files. Third, we develop an *aspect manager* with two main functions – calculating new values for variables in general processes and determining advices. When determined, the advices are then handled by the process executor.

To support dynamicity and diversity, besides the components above, the system also needs the following infrastructure components.

Adaptor: this component has the ability to "provide connectivity, semantic disambiguation and transition services" between our application and 3-rd party services [16]. Therefore, not only it can enable communications in two directions between third party services and the system but it can also convert the different interfaces, protocols, data formats of different parties into the standardized ones for the system and vice versa.

Context server: the context server has four functions: listening to the user-context changes from adaptors, storing context information of users and devices in a database, allowing querying of context information by the aspect manager and informing the aspect manager about user-context changes.

Process executor: the process executor takes care of the execution of general processes and of the external services/processes, as shown in Figure 2.

Service discovery manager: in case there are many third-party services offering the same functions, the service discovery manager aims to assist care-receivers by searching services, prioritizing them and select the most suitable ones.

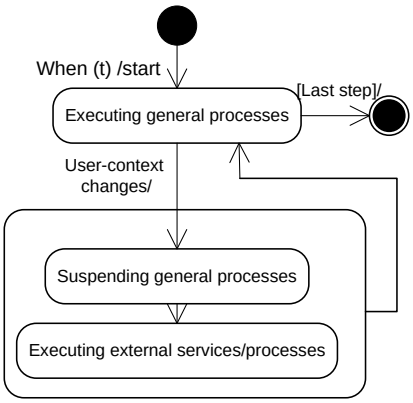

Figure 2: The behavior of the process executor

In the scope of this paper, we assume that the service discovery manager, adaptors and the context server are available and ready to use. We focus solely on the process executor, the aspect repository, the advice repository and the aspect manager. The proposed architecture is shown in Figure 3.

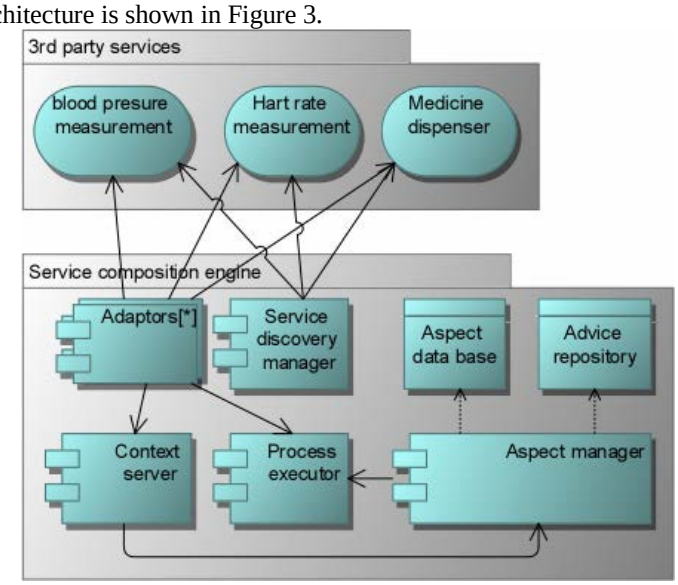

Figure 3: The proposed SOA-based architecture

5 Implementation

To show the feasibility of our approach, we have implemented a prototype that follows the proposed architecture. In this section we discuss the development platforms we used and we explain the implementation of each element, with an emphasis on the aspect manager. Finally, we use in two scenarios to demonstrate how the prototype can handle user-context dynamicity and diversity of needs/preferences.

5.1 Development platform

For building the prototype we have used the Lombardi process engine [17]: a business process manager that allows creating process models, implementing process steps, running and inspecting processes, optimizing and installing process applications [17]. Another feature of this engine we have used is its JavaScript API, which allowed us to invoke one process/service programmatically from another process.

5.2 Implementing the architecture's components

Below we discuss the implementation of the three elements supporting AOA.

Aspect files' structure: With respect to aspects, we follow the specification of AO4BPEL proposed by Charfi and Menzini [18] describing the structure of aspect files. In short, this structure starts with the name of the aspect, followed by pointcut element containing a set of joint points. An advice is a BPEL activity. Other elements like variables, partner links, fault handlers, structured and basic activities in AO4BPEL are inherited from BPEL.

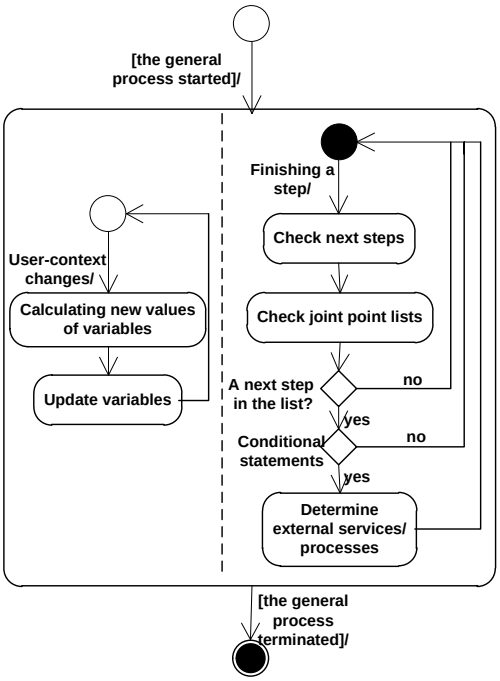

Figure 4: The behavior of the aspect manager

Advices: as mentioned above, advices can be defined in aspect files as BPEL activities. However, in our approach, we decided to define advices in separate files. In this way, it is possible to reuse advices in different aspects.

Aspect manager: to avoid a modification of the process engine, we propose an independent aspect manager. This component is a Java script embedded before or

after one step of a general process. Because of the predefined structure and the content of aspect files, the script can parse aspect files to archive information about joint points, conditional statements and advices. Then, it evaluates condition statements to determine suitable advices or calculates new values for variables in the general process according to user-context changes. The behavior of the aspect manager is depicted in the diagram shown in Figure 4.

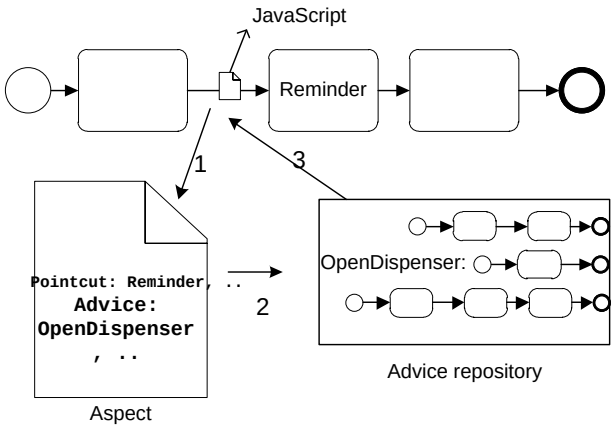

Figure 5: Inserting a process in a general process

Figure 5 shows the location of an aspect in a general process and explains how external processes/services (in the form of advices) can be inserted, via aspects, between the steps of this process. The arrows in Figure 5 have the following significance:

1: The aspect manager (JavaScript code) parses aspect files

2: The advice is found in advice repository

3: The process engine executes the advice

After the addition of the OpenDispenser external (advice) process to the general process, the actual generated and executed process is as depicted in Figure 6.

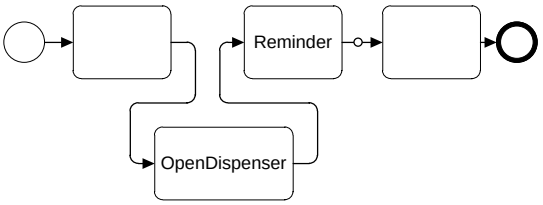

Figure 6: The produced process

To update the variables of the general process, a java script is also placed in a parallel process such that the update task is performed independently without influencing or being influenced by the general process.

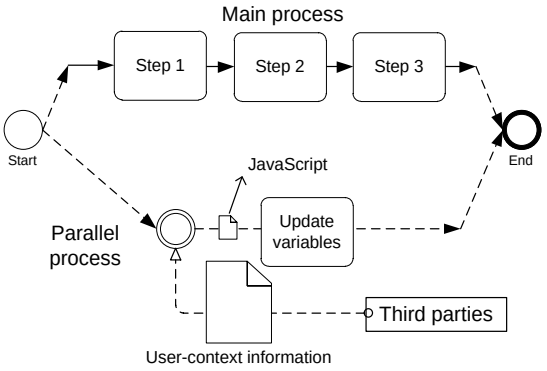

Figure 7: Update variables

5.3 Test cases

In this section, two situations related to the chosen scenario will be addressed. Each of them is associated with different types of requirements. The first one focuses on dynamicity aspects and refers to the case when the care-receiver (Jan) moves from the kitchen to the bedroom where the dispenser needs to be opened automatically (a service to open this dispenser should be available). The second case captures the diversity of preferences/needs of different care-receivers by introducing a second care-receiver. However, before discussing these situations in further detail, the general process for the reminder service scenario is presented.

The general process for reminder service scenario. In the reminder service scenario, the general process that is stable for all user-context changes and user preferences is depicted in Figure 8. The process is triggered by a care giver. The first activity is an inquiry of the user-context information to initialize the variables of the process. For example, based on the care-receiver's location information, the system can calculate t1, t2 and the endpoints of the web-service to invoke (i.e., the suitable reminder device). t1 is the waiting time from the moment the process is triggered until the first reminder is sent. After that, the system waits in t2 before checking whether the medicine is taken or not. If not, the process goes back to the reminder task. This loop is executed until the number of sent reminders is equal to a predefined number, resulting in one alarm sent to care-givers. The logic for the calculation is specified in an aspect file. The reason is that this variable calculation is a crosscutting concern occurring in many places: after the first inquiry and after any information update sent by the context server.

Situation 1: user-context dynamicity. In this paragraph, we describe the expected behavior of the system's in case of a context change and the results of the generated process execution.

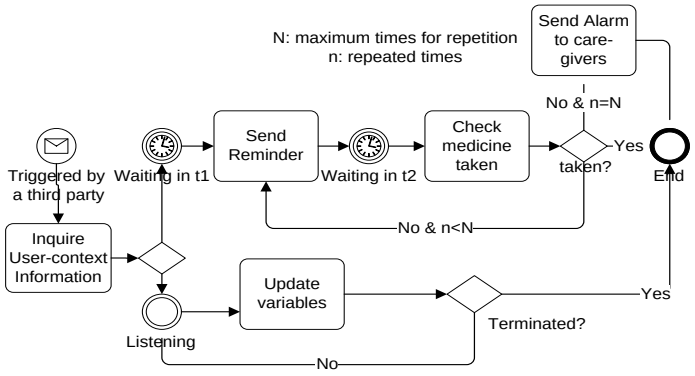

Figure 8: The general process for the reminder service scenario

a) User-context change: Jan moves from the kitchen to the bedroom. In the kitchen, built-in lights are used as reminders while in the sleeping room the TV is used. The medicine dispenser in the bedroom needs to be opened automatically.

b) The process' variables are:t1, t2 as described above; an endpoint address pointing to a specific device (as reminder); user-context information including user' ID number, location and time.

c) Expected system behavior: in dealing with the change, the system needs to update the endpoint address to point to the TV in the bedroom. The external service to open the dispenser is required to be invoked before Jan can remove his medicine.

d) Aspect configurations: there are two aspect configurations: first, for calculating new values of variables; second, for inserting a service to open dispenser. For the sake of simplicity and to avoid the confusion that may be caused by XML tags, we simplify the two aspects' specification by using natural language as shown in Figure 9 and in Figure 10.

Aspect name: calculating variables		
Pointcut: inquire user-context information; listening		
Conditions: user's ID is "p104jan" (Jan's ID)		
Type: after		
Variable: location		
Location	**Endpoint**	**Device**
Kitchen →	http://**130.89.227.130**:9090/ws/Ucare_WS_notifyReminder/	Lights
Bedroom →	http://**130.89.227.132**:9090/ws/Ucare_WS_notifyReminder/	TV

Figure 9: Calculate variables for Jan

Aspect name: open dispenser
Pointcut: send reminder
Conditions: user's ID is "p104jan"
Type: before
Variable: location
Conditions of advice: current location is "bedroom"
Advice: open dispenser

Figure 10: Open dispenser

The first aspect, called *calculating variables*, states that the following action will be performed if the care-receiver is Jan and the current step is after *inquire user-context information* or *listening* in the general process. This aspect simply matches the user's location with the endpoint of services, which invokes the corresponding device at the

user's location. Hence, when Jan is in the bedroom, the television becomes the reminder. This can be considered a *computation rule* without calling advice.

However, as the care-receiver changes his location to the bedroom, one advice (open dispenser) needs to be injected in the general process. This aspect (rule) is shown in Figure 10. This aspect, called *open dispenser*, means that for the user with ID "p104jan", if the current step is before *send reminder* step, an advice to invoke *open dispenser* will be performed.

e) Result: after applying these two aspects, the general process will change into the generated process shown in Figure 9.

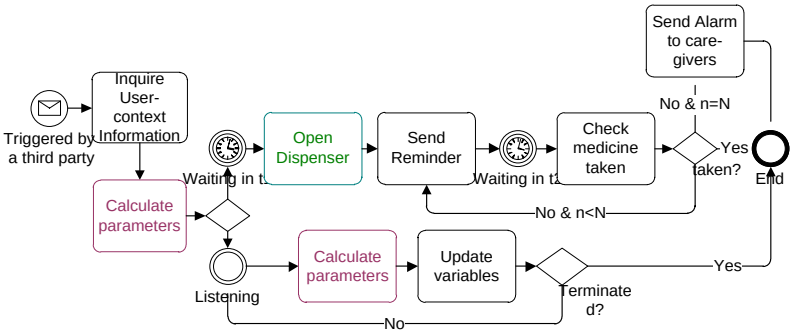

Figure 9: Produced composition

Situation 2: diversity of preferences/needs. To illustrate the behavior of the system in case of diversity of user preferences/needs, we use another example, in which a new care-receiver, Linda, is introduced, adding new references.

a) Different references: Linda, another care-receiver, prefers her PDA over lights or any other devices as reminders.

b) Aspect configuration: similar to the previous situation, by changing the conditional statements, a different preference is formed (see Figure 12).

Aspect name: calculating variables		
Pointcut: inquire user-context information; listening		
Conditions: user's ID is "p104Linda" (Linda's ID)		
Type: after		
Variable: location		
Location	**Endpoint**	**Device**
Kitchen →	http://**130.89.227.131**:9090/ws/Ucare_WS_notifyReminder/	PDA
Bedroom →	http://**130.89.227.131**:9090/ws/Ucare_WS_notifyReminder/	PDA
Corridor →	http://**130.89.227.131**:9090/ws/Ucare_WS_notifyReminder/	PDA

Figure 10: calculate variables for Linda

c) Result: as the ID is a part of user-context information, composing aspects with different IDs reflect the diversity of users' preferences/needs.

6 Conclusions

In this paper, motivated by two current problems of the homecare domain, namely, dynamicity and diversity, we have applied an aspect-oriented approach to the design and implementation of an architecture for the dynamic workflow composition of services. By this approach, not only we have externalized business rules from business processes, but we have also ensured the flexible process interoperability of homecare services in complex processes. Moreover this approach facilitates enterprise interoperability through the composition of services and processes resulting in complex integrated applications. As demonstrated in the previous section, the combination of aspect orientation and SOA has many advantages. The diversity of user preferences is easily controlled by changing the aspect configuration as discussed in the second situation described in the previous section. With regard to the dynamicity of user-context, due to the separation of business rules (in aspect files) from processes, the care-givers can easily update or add new business rules to adapt the system's behavior according to different care-receiver needs or contexts. Furthermore, the idea of externalizing advices and storing them into an advice repository can increase the reusability of advices. This is particularly useful in complex processes where finding places to insert/remove services is time-consuming. Another advantage worth being noted is the fact that aspect manger provides a light-weight AOA implementation solution in the sense that it does not require a modification of the process engine to support aspects (as suggested by [12]), as long as a JavaScript API is supported. Finally, as a part of general processes, the aspect manager can access directly the its variables, minimizing the effort to pass data to external processes and services.

Our approach also has some limitations. Some are inherited from the aspect – oriented approach of Charfi and Menzini and concern the lack of support for complex and multiple business rules [12]. The usage Java script also raises concerns about its flexibility and about its ability to handle different types of business rules. We are currently improving our approach to also support inference rules. Regarding the non-intrusiveness criterion, it should be noted that only pre-defined changes can be handled by the system. In the case of unforeseen events, the care-givers have to assist system developers in defining new business rules and incorporate them in general processes.

References

1. Health-EU. Elderly. 2011 http://ec.europa.eu/health-eu/my_health/elderly/index_en.htm.
2. Gaßner, K. and M. Conrad, ICT enabled independent living for elderly, A status-quo analysis on products and the research landscape in the field of Ambient Assisted Living (AAL) in EU-27. October 2010.
3. Dey, A.K., G.D. Abowd, and D. Salber, A conceptual framework and a toolkit for supporting the rapid protx`otyping of context-aware applications. Hum.-Comput. Interact., 2001. 16(2): p. 97-166.
4. Zarghami, A., Zarifi Eslami, M., Sapkota, B., van Sinderen, M. Service Realization and Compositions Issues in the Homecare Domain. 6th International Conference on Software and Data Technologies (ICSOFT 2011), Seville, Spain, July 2011.
5. O'Brien, L., P. Merson, and L. Bass, Quality Attributes for Service-Oriented Architectures, in Proceedings of the International Workshop on Systems Development in SOA Environments. 2007, IEEE Computer Society. p. 3.

6. Garde S,Knaup P., Requirements engineering in health care: the example of chemotherapy planning in paediatric oncology. Requirements Engineering, 2006, 11(4), 265-278.
7. Shin, J.H., et al. Ubiquitous House and Unconstrained Monitoring Devices for Home Healthcare System. in Information Technology Applications in Biomedicine, 2007. ITAB 2007. 6th International Special Topic Conference on. 2007.
8. Eslami, M.Z. and M.V. Sinderen. Flexible Home Care Automation. in Proceedings of PervasiveHealth 2009 conference. 2009.
9. Erl, T., SOA Design Patterns. 2009: Prentice Hall.
10. Rao, J. and X. Su, A Survey of Automated Web Service Composition Methods, in Semantic Web Services and Web Process Composition, J. Cardoso and A. Sheth, Editors. 2005, Springer Berlin / Heidelberg. p. 43-54.
11. Hull, R., et al., E-services: a look behind the curtain, in Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. 2003, ACM: San Diego, California. p. 1-14.
12. Charfi, A. and M. Mezini, AO4PBEL: an aspect-oriented extention to BPEL. World Wide Web, 2007. 10: p. 309-344.
13. Charfi, A. and M. Mezini, Hybrid web service composition: business processes meet business rules, in Proceedings of the 2nd international conference on Service oriented computing. 2004, ACM: New York, NY, USA. p. 30-38.
14. Rosenberg, F. and S. Dustdar, Business Rules Integration in BPEL " A Service-Oriented Approach, in Proceedings of the Seventh IEEE International Conference on E-Commerce Technology. 2005, IEEE Computer Society. p. 476-479.
15. van Eijndhoven, T., M.E. Iacob, and M.L. Ponisio. Achieving Business Process Flexibility with Business Rules. in Enterprise Distributed Object Computing Conference, 2008. EDOC '08. 12th International IEEE. 2008.
16. Papazoglou, M. and W.-J. van den Heuvel, Service oriented architectures: approaches, technologies and research issues. The VLDB Journal, 2007. 16(3): p. 389-415.
17. IBM-InfoCentre. Lombardi tasks. 2011 [cited 2011 June 01]; Available from: http://publib.boulder.ibm.com/infocenter/wle/v7r2/index.jsp.
18. Charfi, A., Aspect-Oriented Workflow Languages: AO4BPEL and Applications, in Fachbereich Informatik. 2007, TU Darmstadt: Darmstadt.Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195--197 (1981).
19. Hevner, A.R., March, S.T., and Park, J. (2004), Design research in information systems research, MIS Quarterly, 28, 1, p. 75–105.
20. Peffers, K., Tuunanen, T., Rothenberger, M., and Chatterjee, S. (2008), A Design Science Research Methodology for Information Systems Research, Journal of Management Information Systems, Vol. 24 Issue 3, no 3, p. 45-77.
21. Chen, D., Doumeingts, G., and Vernadat, F. (2008), Architectures for Enterprise Integration and Interoperability:Past,Present and Future, Computer in Industry59,647-659.
22. Zarifi Eslami, M. and van Sinderen, M.J. (2009) Flexible home care automation adapting to the personal and evolving needs and situations of the patient. In: 3rd Intl. Conf. on Pervasive Computing Technologies for Healthcare (PervasiveHealth 2009), pp. 1-2. IEEE.
23. B. Sapkota, C. Asuncion, M. Iacob, and M. van Sinderen, “A Simple Solution for Information Sharing in Hybrid Web Service Composition,” in 15th IEEE Int. Conf. on Enterprise Distributed Object Computing Conference (EDOC 2011), 2011, pp. 235 –244.
24. Zarghami, A., Sapkota, B., Zarifi Eslami, M., van Sinderen, M. (2012) Decision as a service: Separating decision-making from application process logic. In: 16th IEEE Intl. Conf. on Enterprise Distributed Object Computing (EDOC 2012), 103–112.
25. Asuncion, C.H. and Iacob, M.-E. and van Sinderen, M.J. (2010) Towards a flexible service integration through separation of business rules. In: 14th IEEE International EDOC Enterprise Computing Conference (EDOC 2010), pp. 184-193. IEEE Comp. Soc.