

Service Composition in a Context-Aware Setting with Functionally Equivalent Services

Sujata Swain and Rajdeep Niyogi(✉)

Indian Institute of Technology Roorkee, Roorkee 247667, India
sujataswain019@gmail.com, rajdpfec@iitr.ac.in

Abstract. Context-aware systems play a vital role in facilitating our daily life activities. The development of context-aware systems is an inherently complex task. These systems adapt to context change and provide a composed service to satisfy user requests. This paper aims to provide a complete sketch of a context-aware system. To illustrate the working of the proposed system, we consider a context-aware e-learning application that provides different types of study material according to the context. When context changes, a new plan is obtained to satisfy the user request. We implement and evaluate the context-aware e-learning system with the different phases of the proposed system. The experimental results show the efficacy of the approach.

Keywords: Context change · Classical planning · Functionally equivalent · QoS-aware

1 Introduction

Nowadays the pervasive computing is a promising approach for facilitating the daily life activities/tasks. The users may have various smart devices in their vicinity. Therefore, the ubiquity of smart devices helps to satisfy the daily life activities, by modeling the resources as services and by integrating the services in their vicinity. However, the interaction and management of all various devices and services is considered as a challenging task. The context-aware system is an emerging solution to alleviate such challenges. The main goal of context-aware systems is to acquire the context information, to utilize that information and provide the appropriate services to the current context. It will supervise the user's interaction with the ubiquitous environment for automating users' repetitive actions [1]. A context-aware application aims to satisfy the user request/goal G anywhere and anytime, even if a context changes, i.e., for all possible contexts $c = \{c_1, c_2, \ldots, c_k\}$ in a given domain, find a plan π such that plan $\pi \models_c G$. This is shown in Fig. 1. For example, a context-aware system can detect that a user never read a message while driving, and thus a text to voice translation service read the message header through a voice box in the car whenever he/she is driving.

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12253, pp. 106–119, 2020.
https://doi.org/10.1007/978-3-030-58814-4_8

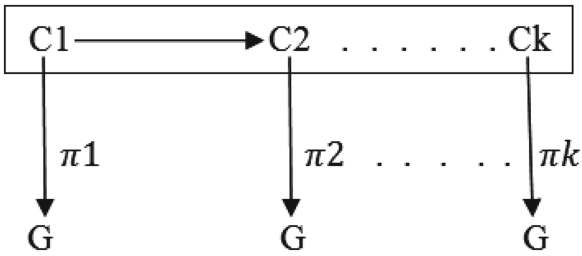

Fig. 1. Goal maintenance with context change

With the proliferation of services suitable automatic service composition algorithms are required to not only synthesize correct plans from thousands of services but also satisfy the quality requirements of the users. The user request can be associated with some end-to-end QoS requirements like maximum price the user can spend, minimum reliability needed for the composition. Therefore, a QoS-aware service composition approach should select services to build an optimal composed service that meets the end-to-end QoS constraints [2]. Metaheuristic approaches [3] have proved to be one of the most promising techniques for QoS-aware service composition [4]. There exists some automated planning based approaches to address the problem of QoS-aware service composition [5]. However these approaches have the following drawbacks: they cannot deal with more than one QoS parameter and they are not designed to handle context changes. In this paper, we propose a complete architecture of the context-aware system.

The main contributions of this work are as follows:

- OWL-S description abstract service is transformed into planning domain. User request, context, and inputs are also transformed into the planning problem. The produced domain and problem are described using well-established standards, such as PDDL.
- A flexible planner can be employed which has the functionality to get a plan, such as Blackbox planner. A plan contains a set of actions which are the abstract services description.
- The user's context may change during execution time. A replanning component is considered to obtain a new plan according to a new context.
- The functionally equivalent services are modeled using a meta-reasoner tree, which may greatly impact on the unavailability of abstract service.
- The above mentioned approaches would be shown with the help of a context-aware e-learning application. In this case study, we also see how a composed service changes when the context changes.

The rest of this paper is organized as follows. Related work is discussed in Sect. 2. The proposed approach is given in Sect. 3. A case study is given in Sect. 4. The implementation and experimental results are given in Sect. 5. We conclude the paper in Sect. 6.

2 Related Work

AI planning [6] is a promising approach for automatic service composition. Services are mapped to actions and a composed service is mapped to a plan. The problem of finding a composed service reduces to a finding a plan. Some of the well-known AI planning approaches are Hierarchical Task Network (HTN) planning, Graphplan, and OWLS-XPlan. Some researchers [7] have adopted service composition using HTN planning. In this method, a plan is obtained by decomposing a complex task into simple sub-tasks. HTN planners (SHOP and SHOP2) are used to obtain a plan. In [7], a context-aware service composition framework based on planning is suggested. The approach uses HTN planning to obtain a plan and the composed service is based on user request, service context, and user context. User context information includes user location. Service context information includes response time, availability, and throughput. Some researchers have adopted service composition using OWLS-Xplan [8]. An OWL-S service composition planner, called OWLS-Xplan is proposed for service composition [8]. Xplan, a hybrid planner, is a combination of Fast Forward (FF) planning and Hierarchical Task Networking (HTN) planning methods.

Different service providers provide services with the same functionality to solve a simple task but offering different QoS parameters, such as price, time, reliability, and availability, etc. QoS-aware service composition approaches provide an optimal composite service to satisfy a user request [9]. These approaches consider a set of predefined simple tasks and a set of candidate services corresponding to a simple task. The QoS-aware service composition approaches select a service from the set of candidate services corresponding to a simple task. Several researchers proposed service composition using genetic algorithm methods [10]. Many researchers adopted particle swarm optimization algorithms for QoS-aware service composition [11]. Some researchers adopted fruitfly optimization algorithm (FOA) for service composition [4]. In [12], a hybrid genetic algorithm (HGA) which combines the genetic algorithm and fruitfly optimization algorithm is proposed and applied to select services. This method has good feasibility, optimality and low execution time.

3 An Adaptive Context-Aware Service Composition Approach

In this section, we present our proposed system that aimed at enabling context-aware service composition application. This is achieved by generating an abstract composed service, selecting an optimal concrete composed service, and adapting to the context changes in the environment. The proposed context-aware service composition system is shown in Fig. 2. The components of the proposed system are discussed below.

Translation Phase: This phase prepares all the required inputs for the upcoming phases. This phase contains two modules: domain translation and problem translation. Input to domain translation module is OWL-S description of

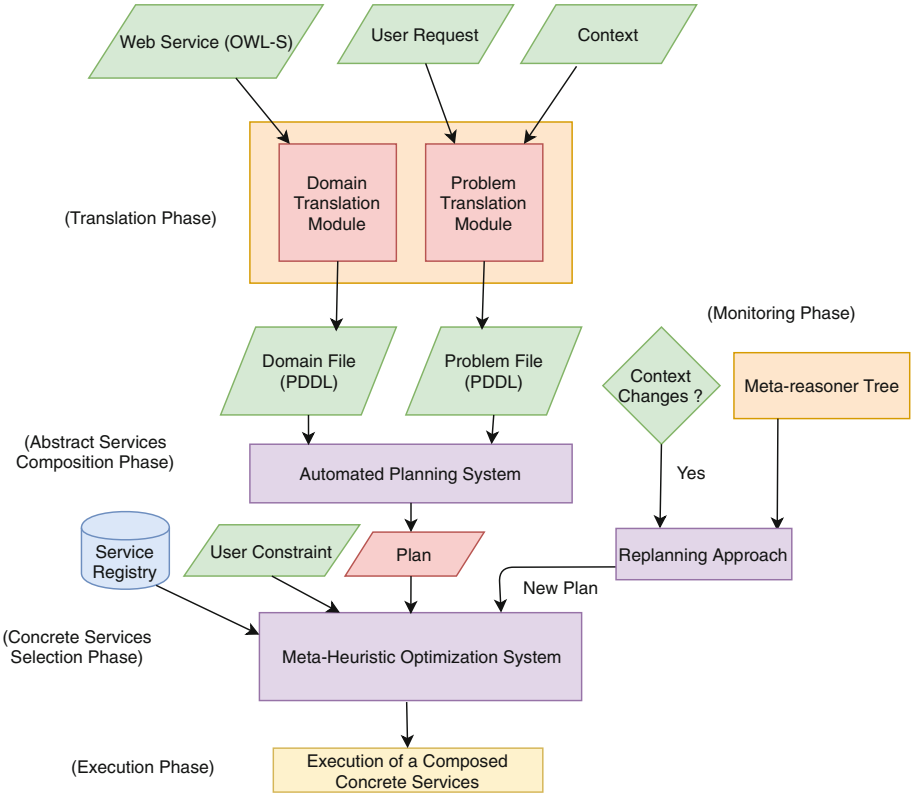

Fig. 2. Architecture of the proposed QoS-based context-aware system

abstract services that are located in the service repository. It translates the OWL-S description of an abstract service into an action of Domain PDDL. The translation phase collects user's request, user's inputs and also collects context of the environment through sensors. The problem translation module constructs an initial state from the user's inputs and context, whereas, from the user's request, it constructs a goal state. The combination of initial state and goal state is Problem PDDL.

Abstract Services Composition Phase: This phase aims to get an abstract composed service (plan) from Domain PDDL and Problem PDDL. Blackbox planner [18] is used to obtain a plan. A plan is a sequence of actions (services).

Concrete Services Selection Phase: This phase aims to get an optimal concrete composed service corresponding to an abstract composed service (plan). It retrieves the sets of concrete services (CS) corresponding to the abstract services from the service repository. It gets the global constraints ($GCst$) and user's preference (w) on QoS attribute which are provided by a user. It uses a meta-heuristic approach to obtain an optimal concrete composed service.

Execution Phase: This phase aims to execute a concrete composed service to satisfy a user request. If any one of the concrete service becomes unavailable then it selects another concrete composed service to satisfy a user request.

Monitoring Phase: This phase aims to monitor the environment of the system. It continuously monitors the changes in the environment, due to dynamic nature of the environment. If there is any context change in the environment, then some new services are available in the new context and also some services (which are available in previous context and being executed to satisfy the request) may be unavailable in the new context, due to which the composed service fails to satisfy the request. We use a replanning approach that provides a new plan in a new context. This approach replaces an unavailable service by a functionally equivalent service [14–16]. A functional equivalent service corresponding to an unavailable service is obtained and replace that unavailable service in the plan.

Functionally Equivalent Service: Two services m, m' are functionally equivalent if any one of the following conditions hold:

1. m, m' are equivalent w.r.t structure
2. m, m' are equivalent w.r.t function

Example-1: Functionally equivalent service w.r.t structure.
In a payment service application there is a service '*payment through credit card*' which takes as input a card number and an one time password (OTP) and provides payment confirmation as output. There is another service '*payment through debit card*' which takes as input a card number and an one time password (OTP) and provides payment confirmation as output. These two services are functionally equivalent w.r.t structure as it takes same inputs and provides same outputs. In Fig. 3, Service A and Service B are functionally equivalent w.r.t structure, if $I_1 = I_3$ and $I_2 = I_4$ and $O_1 = O_2$.

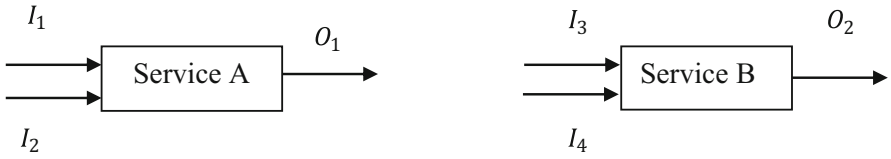

Fig. 3. Representation of services based on input and output

Example-2: Functionally equivalent service w.r.t function.
According to [13], a service can be anything e.g., hardware devices, network resources, a piece of computation, and even a human being. A medicine domain is presented in [14], where medicine is considered as a service. 'Lupiclor' is a

medicine used for hypertension. There is another medicine 'Lorvas' which is also used for hypertension disease. Doctor is an expert in the medicine domain and he/she can decide whether two medicines are equivalent or not. Doctor may recommend the use of 'Lorvas' medicine in place of 'Lupiclor' medicine. If Lorvas can be used in place of Lupiclor, then these two medicines are functionally equivalent w.r.t function as they are used for hypertension disease.

4 Case Study

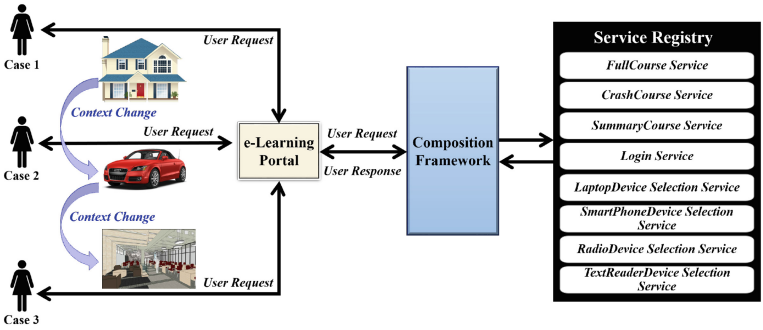

Fig. 4. Context-aware e-Learning application (CEA)

An e-learning domain as a context-aware application, named as Context-aware e-Learning Application (CEA) is considered. This application helps in the learning process of a student and saves time and effort for preparing notes. When a user makes a request regarding a subject she wants to study, the portal uses a composition framework to combine different atomic services. There are different atomic services in this system: login service, subject material service, and device selection service. Each of them accepts certain inputs and produces certain outputs.

User requests are enriched with context information. For example, the portal takes into account the following context types: student's location, number of days remaining for the examination and the computing device in use. These contexts can be sensed by a module in the application. The CEA delivers lectures based on the number of days remaining for the examination and the best device available at the student's location. Let X be the number of days remaining before the first examination and course duration be 5 months (150 days approximately). Let the CEA provide a course material α where

$$\alpha = \begin{cases} Fullcoursematerial & if\ 30 \leq X < 150 \\ Crashcoursematerial & if\ \ 2 \leq X < 30 \\ Summarycoursematerial & if\ \ 0 \leq X < 2 \end{cases}$$

Figure 4 describes the following three cases.

Case 1. In the first case, student is at Home. Student has the subscription to the e-Learning portal. The portal provides the login service which allows it acquires the information about some information the subscribed user. The portal provides student's requested subject material through the best device available at Home. The portal receives the context information like, student's location, number of days remaining before the examination, and devices available at that location. Let the number of days remaining before the examination be 45; the full course subject material service will be selected by the portal. Let the devices that are available at home be the Laptop and Smartphone. The devices are ranked according to screen size, memory, and network bandwidth. Assuming that network bandwidth is high at Home, the portal will select Laptop as the device to which communication will be made. In order to satisfy a user request, the resulting service is composed from atomic services, such as Login Service, Full course subject material service, and Laptop device selection service.

Case 2. In the second case, student has left Home and enters a Car to go to College. Now the student's location has changed from Home to Car. The portal receives the current context information, like, the devices that are available in the Car. Let the devices that are available in the Car be Radio and Smart phone. Assuming that network bandwidth is medium at Car, the portal will select Radio as the device to which communication will be made. In order to satisfy a user request, the resulting service is composed from atomic services, such as Login Service, Full course subject material service, and Radio device selection service.

Case 3. In the third case, student has left Car and enters into the Classroom. Now student's location has changed from Car to Classroom. The portal receives the current context information, like, the devices that are available in the Classroom. Let the devices that are available with the student be Text reader and Smart phone. Assuming that network bandwidth is low at Classroom, the portal will select Text reader as the device to which communication will be made. In order to satisfy a user request, the resulting service is composed from atomic services, such as Login Service, Full course subject material service, Text reader device selection service.

5 Experimental Results

We implement the proposed system in a context-aware e-learning application (CEA). The system is developed with the help of different tools. These are protege, automated planner, MATLAB, and Eclipse IDE (a Java-based platform). Protege is an ontology editor, which contains OWLS-editor, SPARQL query language, semantic web rule language (SWRL) and pellet reasoner as inference engine. Experiments are run on the Intel Core i5 2.53 GHz machine with 4 GB of RAM.

Table 1 shows the various devices that are registered to be used at specific location. For example, devices 'SonyLaptop' and 'SamsungSmartPhone' are registered at location 'Home', devices 'StereoRadio' and 'SamsungSmartPhone' are

registered at location 'Car', devices 'Kindle' and 'SamsungSmartPhone' are registered at location 'Classroom'. When the location does not match any one of Home/Car/Classroom, then 'SamsungSmartPhone' is selected.

Table 1. Device registry corresponding to the location.

Location	Device
Home	Laptop, SmartPhone
Car	SmartPhone, Radio
Classroom	Textreader, SmartPhone
Any other location	SmartPhone

The various services are available in registry, as shown in Fig. 5.

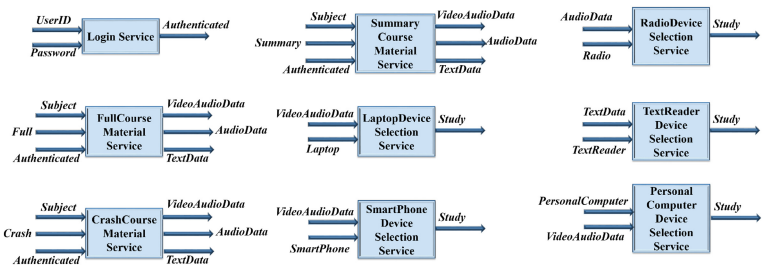

Fig. 5. Services in registry

The user request is *to study Computer Network*.

5.1 Translation Phase

Service descriptions are translated to Domain PDDL. A service is translated into an action in Domain PDDL. Inputs and outputs of a service are translated into the preconditions and effects of an action respectively. The Domain PDDL is shown below.

```
( define (domain EducationSystem_Domain)
(:types location device service subject - object user - agent)
(predicates: (UserId ?u)(Password ?p)
(Authenticated ?u)(Subject ?sub)
(Full ?f) (Crash ?c) (Summary ?s)
(VideoAudiodata ?sub) (Audiodata ?sub) (Textdata ?sub)
(Laptop ?d)(SmartPhone ?d) (Radio ?d))
(Study ?sub ?d))
```

```
   (:action LoginService
   :parameters (?u - UserId ?p - Password )
   :precondition (and (UserId ?u)(Password ?p))
   :effect (Authenticated ?u))
   (:action FullCourseMaterialService
   :parameters (?u - Authenticated ?sub - Subject ?f - Full )
   :precondition (and (Authenticated ?u)(Subject ?sub)(Full ?f))
   :effect (and (VideoAudiodata ?sub) (Audiodata ?sub) (Textdata
?sub) ))
   (:action CrashCourseMaterialService
   :parameters (?u - Authenticated ?sub - Subject ?c - Crash )
   :precondition (and (Authenticated ?u)(Subject ?sub)(Crash ?c))
   :effect (and (VideoAudiodata ?sub) (Audiodata ?sub) (Textdata
?sub) ) )
   (:action LaptopDeviceSelectionService
   :parameters (?sub ?d )
   :precondition (and (VideoAudiodata ?sub) (Laptop ?d))
   :effect (Study ?sub ?d))
   (:action SmartPhoneDeviceSelectionService
   :parameters (?sub ?d )
   :precondition (and (VideoAudiodata ?sub) (SmartPhone ?d))
   :effect (Study ?sub ?d) )
   (:action RadioDeviceSelectionService
   :parameters (?sub ?d )
   :precondition (and (Audiodata ?sub) (Radio ?d))
   :effect (Study ?sub ?d) )
    ... )
```

A user request, input, and context are translated into Problem PDDL. User provides his user id ('sujata'), password ('swain') and submits his request ('to study CN'). The application obtains the location information ('Home') and number of remaining days to the examination ('Full'). From the device registry, the application selects a device available at that location ('SonyLaptop'). The Problem PDDL is shown below.

```
   (:init
   (UserId sujata)
   (Password swain)
   (Subject CN)
   (Full CN-full)
   (Laptop SonyLaptop) )
   (:goal (Study CN SonyLaptop) )
```

5.2 Planning Phase

In this phase, a Blackbox planner is used to obtain a plan. Blackbox planner takes Domain and Problem PDDL as input. Blackbox planner obtains a plan, which

contains a sequence of actions. Actions are same as services. The Blackbox plan is shown below. It contains Login service, Full course material service, Laptop device selection service. These services are composed to satisfy a user request.

```
Begin plan
1 (loginservice sujata swain)
2 (fullcoursematerialservice sujata cn cn-full)
3 (laptopdeviceselectionservice cn sonylaptop)
End plan
```

The composed service structure of the plan is shown in Fig. 6.



Fig. 6. A composed service for a user request in CEA when user's location is Home



Fig. 7. A composed service for a user request in CEA when user's location is car and SmartPhone is selected

5.3 Selection Phase

In this phase, the concrete services are selected corresponding to the abstract services. We use a meta-heuristic selection algorithm called hybrid genetic algorithm (HGA) [12], to obtain a list of composed concrete service ($CCSs$). We have used QWS dataset [17] of real web services. The dataset contains 2507 rows and 9 columns. A row represents a concrete web service, and a column represents a QoS attribute. The QoS attributes are response time, availability, and throughput, the likelihood of success, reliability, compliance, best practices, latency, and documentation. In our experiments, we consider 2500 rows (2500 concrete web services) and 4 columns (4 attributes). We consider four QoS attributes: response time, latency, availability and reliability. We consider that each QoS attribute is given equal preference. The stopping criterion for HGA algorithm is maxItr = 300.

In CEA application, a plan contains three number of abstract services. These are login service, full course material service, and laptop device selection service. Therefore, the chromosome size is 3. The population size is the number of concrete services corresponding to an abstract service. The population size should

be less than or equal to 2500/chromosome size. We have considered population size is 350 ($\leq$2500/3). These services contain 350 concrete services in each. After applying HGA algorithm, the five best composed concrete services are given in Table 2. The first row can be interpreted as: 151 denotes 151st concrete service of the login service, 182 denotes 182nd concrete service of the full course material service, 311 denotes 311st concrete service of the laptop device selection service, and the value of objective function is 2.4224.

Table 2. Concrete services selection in CEA application.

Login service (AS1)	Full course material service(AS2)	Laptop device selection service (AS3)	Value of objective function
151	182	311	2.4224
338	182	311	2.4235
106	182	311	2.4339
79	182	311	2.4461
79	149	311	2.4472

5.4 Monitoring Phase

When a context changes, some concrete services may become unavailable. Due to which, some selected abstract services may not have any concrete services. Let AS_i be an abstract service that is unavailable due to context change. To handle such situation, a replanning strategy is suggested. We describe the replanning strategy through two cases.

Case-1: User's context (location) changes from Home to Car and available network bandwidth is high.

When the user's context changes, there may be a change in the set of available services and resources. In CEA application, we take user's location as context. When the location changes from Home (c) to a Car (c'), a set of available devices also changes, due to which the request cannot be satisfied. In CEA application, the user submits her request for the full course material of a subject. At that time user's location is at 'Home'. The proposed system provides the full course material through 'Laptop'. While providing the material, user's location changes from 'Home' to 'Car'. Let 'Laptop' be not available at the 'Car'. Due to which, Laptop device selection service becomes unavailable. Available devices at Car: SamsungMobile (SmartPhone), StereoRadio (Radio). Due to high network bandwidth available, SamsungMobile is selected to display the output. The application obtains a new plan for the unavailable service. The application generates a new problem PDDL for an unavailable service at the new context. The new Problem PDDL is generated as shown below.

```
(:init
(UserId sujata)
(Password swain)
(Subject CN)
(Full CN-full)
(SmartPhone SamsungMobile) )
(:goal (Study CN SamsungMobile) )
```

A new plan is obtained for the unavailable service using Blackbox planner. The obtained new plan is shown below.

```
Begin plan
1 (loginservice sujata swain)
2 (fullcoursematerialservice sujata cn cn-full)
3 (smartphonedeviceselectionservice cn samsungmobile)
End plan
```

The composed service structure of the plan is shown in Fig. 7.

Case 2: Full course service becomes unavailable.
Sometime only context information cannot help to obtain a new plan for the unavailable service. Therefore, a meta-reasoner is required to obtain the functionally equivalent services to an unavailable service. In CEA application, a meta-reasoner tree is implemented by ontology in protege tool. The functionally equivalent services are becomes siblings in the meta-reasoner tree.

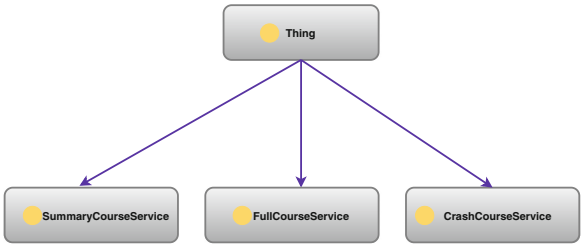

Fig. 8. Meta-reasoner tree of CEA application

In the CEA application, FullCourseService, CrashCourseService, and SummaryCourseService are providing lecture materials of a subject. As, we know these are providing a full description, a brief description, and a summary description respectively. Therefore these services are becoming the siblings in the meta-reasoner tree, as shown in Fig. 8.

A user has requested for FullCourseService of a subject CN, let's say S. However, there is no concrete service available to satisfy S. The existing system fails under this circumstance. The application gets the siblings of S, i.e., CrashCourseService (let's say, S_1) and SummaryCourseService (let's say, S_2). The service S_1 or S_2 are FE services of a service S. Therefore, CrashCourseService (S_1) or SummaryCourseService(S_2) can be substituted in the place of FullCourseService

(S) in the plan. The meta-reasoner provides the type 'Crash' which is functionally equivalent with the type 'Full'. The application generates a new Problem PDDL as shown below.

```
(:init
(UserId sujata)
(Password swain)
(Subject CN)
(Crash CN-Crash)
(Laptop SonyLaptop) )
(:goal (Study CN SonyLaptop) )
```

A new plan is obtained for the unavailable service using Blackbox planner. The obtained new plan is shown below.

```
Begin plan
1 (loginservice sujata swain)
2(crashcoursematerialservice sujata cn cn-crash)
3 (laptopdeviceselectionservice cn sonylaptop)
End plan
```

The composed service structure of the plan is shown in Fig. 9.

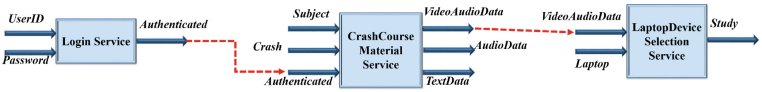

Fig. 9. A composed service for a user request in CEA when user's location is home and full course service is not available

6 Conclusion

We developed an architecture of a QoS based context-aware system. As a case study we considered a context-aware e-learning application that provides different types of study material according to the context. When context changes, a functionally equivalent service is replaced in place of an unavailable service using automated planning. The experimental results show the efficacy of the approach.

Acknowledgements. The authors thank the anonymous reviewers of ICCSA 2020 for their valuable suggestions. The second author was in part supported by a research grant from Google.

References

1. Chihani, B., Bertin, E., Jeanne, F., Crespi, N.: Context-aware systems: a case study. In: Cherifi, H., Zain, J.M., El-Qawasmeh, E. (eds.) DICTAP 2011. CCIS, vol. 167, pp. 718–732. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22027-2_60

2. Alrifai, M., Risse, T., Nejdl, W.: A hybrid approach for efficient Web service composition with end-to-end QoS constraints. ACM Trans. Web (TWEB) **6**(2), 1–31 (2012)
3. Osman, I.H., Kelly, J.P.: Meta-heuristics theory and applications. J. Oper. Res. Soc. **48**(6), 657–657 (1997)
4. Zhang, Y., Cui, G., Wang, Y., Guo, X., Zhao, S.: An optimization algorithm for service composition based on an improved FOA. Tsinghua Sci. Technol. **20**(1), 90–99 (2015)
5. Peer, J.: A PDDL based tool for automatic web service composition. In: Ohlbach, H.J., Schaffert, S. (eds.) PPSWR 2004. LNCS, vol. 3208, pp. 149–163. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30122-6_11
6. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Elsevier, Amsterdam (2004)
7. Omid, M.: Context-aware web service composition based on AI planning. Appl. Artif. Intell. **31**(1), 23–43 (2017)
8. Klusch, M., Gerber, A., Schmidt, M.: Semantic web service composition planning with OWLS-Xplan. In: Proceedings of the 1st International AAAI Fall Symposium on Agents and the Semantic Web, pp. 55–62 (2005)
9. Singh, R.P., Pattanaik, K.: An approach to composite QoS parameter based web service selection. Procedia Comput. Sci. **19**, 470–477 (2013)
10. Amiri, M.A., Serajzadeh, H.: QoS aware web service composition based on genetic algorithm. In: IEEE Fifth International Symposium on Telecommunications, pp. 502–507 (2010)
11. Ludwig, S.A.: Applying particle swarm optimization to quality-of-service-driven web service composition. In: IEEE 26th International Conference on Advanced Information Networking and Applications, pp. 613–620 (2012)
12. Seghir, F., Khababa, A.: A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. J. Intell. Manuf. **29**, 1–20 (2016)
13. Lee, C., Ko, S., Lee, S., Lee, W., Helal, S.: Context-aware service composition for mobile network environments. In: Indulska, J., Ma, J., Yang, L.T., Ungerer, T., Cao, J. (eds.) UIC 2007. LNCS, vol. 4611, pp. 941–952. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73549-6_92
14. Swain, S., Niyogi, R.: Smartmedicist: a context-aware system for recommending an alternative medicine. Int. J. Pervasive Comput. Commun. **14**(2), 147–164 (2018)
15. Swain, S., Niyogi, R.: Context-aware service composition with functionally equivalent services for complex user requests. In: Barolli, L., Amato, F., Moscato, F., Enokido, T., Takizawa, M. (eds.) AINA 2020. AISC, vol. 1151, pp. 1089–1100. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44041-1_94
16. Swain, S., Niyogi, R.: An ontology based approach for satisfying user requests in context aware settings. In: IEEE 30th International Conference on Advanced Information Networking and Applications, pp. 1130–1137 (2016)
17. Al-Masri, E., Mahmoud, Q.H.: QoS-based discovery and ranking of web services. In: IEEE Proceedings of 16th International Conference on Computer Communications and Networks, pp. 529–534 (2007)
18. Kautz, H., Selman, B.: BLACKBOX: a new approach to the application of theorem proving to problem solving. In: Workshop on Planning as Combinatorial Search (AIPS 1998), vol. 58260, pp. 58–60 (1998)