A Goal-Driven Context-Aware Architecture for Distributing Cognitive Service Group

Siyuan Lu(✉)

School of Software and Microelectronics, Peking University, Beijing, China
larrylu0426@icloud.com

Abstract. Cognitive service is an emerging service paradigm in service-oriented computing. It can comprehend data in the same way as the human. Cognitive services require sufficient information to understand service scenarios. Actually, to achieve a goal sometimes requires multiple services with order dependencies and prerequisites to work collaboratively. When an exception event occurs during the service group working, the conventional approach is to restart or stop the service based on the exception type. If the environment information is changed much fast and retrieved unpractically, the exception event can cause the delayed response of the service group. If the goal of the service group is time-aware and service result is preferred, the regular policy is hard to match the requirement. In this paper, we address the problem of delayed response caused by exception events raised from the distributing cognitive service group. A novel architecture is proposed to ensure the overall consistency and real-time reaction of distributing cognitive service group.

Keywords: Service-oriented computing · Cognitive service · Context awareness

1 Introduction

Cognitive services require sufficient information to understand service scenarios. To obtain sufficient information, cognitive services often require multiple data sources. Furthermore, multiple resource (e.g., computing, storage, and cache) nodes are often required to achieve the desired result. Because each data source has different device characteristics and physical locations, only part of the data from the service environment can be obtained, and the service environment cannot be fully represented. Therefore, it is necessary to integrate the data from all channels to represent the service environment. Actually, to achieve a goal sometimes requires multiple services with order dependencies and prerequisites to work collaboratively. The functions and features of each service are different to a large extent and all have the independent operational capability. These individuals need work together for an ultimate goal while communication and consistency of each other. We gather these services as a cluster and name it "distributing cognitive service group".

When an exception event occurs during the service group working, the regular approach is to restart or stop the service based on the exception type. Restarting the service need to abandon the current service progress and then retrieve the previous data

X. Liu et al. (Eds.): ICSOC 2018 Workshops, LNCS 11434, pp. 400–406, 2019.
https://doi.org/10.1007/978-3-030-17642-6_34

to execute again. Even, it needs to choose another available resource node due to the current invalid one. In any case, all manners need a broad temporal and resource cost. If the environment information is changed much fast and retrieved unpractically, the broken-down service accumulates large amount of context data required to handle in time. Other related services need to wait for the invalid service to be valid. Further, the exception event can cause the delayed response of the service group. If the goal of the service group is time-aware and service result is preferred, the regular policy is hard to match the requirement. This is a severe challenge about service high availability and instantaneity.

In this paper, we propose a novel service architecture for multiple cognitive services to promote the adaptability to the environment and robustness of service group. We introduce the concept of context-aware to achieve the perception, understanding, and decision of the exception and deviation across the service collaboration and service working then assess the impact of both on the achievement of goals. Depending on the degree of impact, the influenced service decides to ignore or handle the exception, further, to continue or restart even stop the service. The significance of context is to provide an overall consistent basis for the service group and a prerequisite for the execution of the service. When the exception and deviation rise, the relationship between the goal and the current service working progress state is considered according to the overall status of the service group at that time, and the high availability of the service is guaranteed under the condition of ensuring the overall consistency and real-time response. The distributing cognitive service group based on this architecture can acquire the expected result with the unexpected exception.

To achieve it, this paper will first give the overview of cognitive service and research on combing service computing and contextual computing. Second, give a high-level design of the proposed architecture and some related concept explanations in short. Next, we enumerate some challenge points and give some insight from us about how to solve them. The paper closes with a conclusion.

2 Related Work

Cognitive service is an emerging service paradigm in service-oriented computing. It's able to comprehend data in the same as the human. For instance, the human face recognition service not only can find where the human faces are but also can know who they are, even mind the human character information (e.g., affect, sex) from the face shape, skin color, expression and so on. The cognition process doesn't need the human intermediary. In a word, cognitive service makes service more intelligent. So far, there are few relevant studies in this field. There is no clear clarification of cognitive service across the academia and industry. Microsoft is working on this field even has published several cognitive services with the API such as Emotion, LUIS, Speaker Recognition [1].

From the perspective of cognitive service, Context is a powerful, and longstanding concept to depict the interaction environment which is usually complexing and variety [10]. Interaction with service is by explicit the information given by users, and the context is implicit, e.g., the environment information surrounding the service and user. Context is useful to interpret explicit interaction [2, 4], giving the input information

more rich implications. According to Hong et al. “to provide adequate service for the users, applications and services should be aware of their contexts and automatically adapt to their changing contexts-known as context-awareness” [5].

To combine the service with context, one popular approach existed in many studies is to make context as a middleware embedded into the existed service system to provide the context cognition capacity for automatically adapt to extremely dynamic computing environments. From the context side, the combination involves the definition of context lifecycle divided into four necessary stages [9]: context acquisition, context modeling, context reasoning and context disposition. From the service side, the combination involves the condition to trigger services and the relationship/dependency between them. Gu et al. [3] presented a service-oriented context-aware middleware (SOCAM) architecture for the building and rapid prototyping of context-aware mobile services. The core of this work is to define an ontology-based approach to model various contexts for supporting semantic representation, context reasoning and context knowledge sharing and provide an original insight about context-aware service system architecture. However, this proposal only considered the service automation rather than service efficiency and availability. Other related works about the context-aware service-oriented middleware such as Hydra [6], SCIMS [7] and CAMEO [8] have given an enormous contribution on the context side for providing more comprehensive context representation to promote more effective service response capacity. Nevertheless, considering the capacity of context, service can exploit the change context information to perceive the exceptions raised from the service environment and make a right decision to reduce the negative impact. Further, on the other hand, context is meaningful to promote the service robustness and high availability.

3 Proposal

In this section, we mainly introduce the high-level design of proposed architecture which involves two main topics as bellow (related discussion detail is in the subsection):

- Which concepts are essential to define a cognitive service and how the functional cognitive services work together to achieve the specific goal.
- How to use known information to define the service environment context and use context for ensuring the goal achievement punctually during the service working period.

3.1 Overview on Distributing Cognitive Service Group

According to the functional requirement, cognitive service need use known data to satisfy user’s expected result, for instance, customers can use Meeting Record service to extract the speech content of attendees from the audio data and video data which can be acquired from meeting room monitor device and phone. The attendee information can be an optional data to help the service to comprehend the speech content deeper.

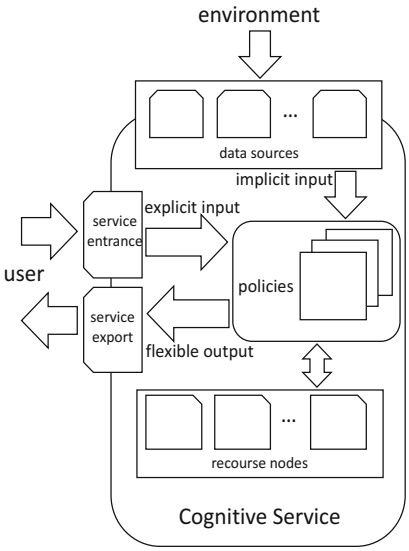

Fig. 1. Cognitive service structure.

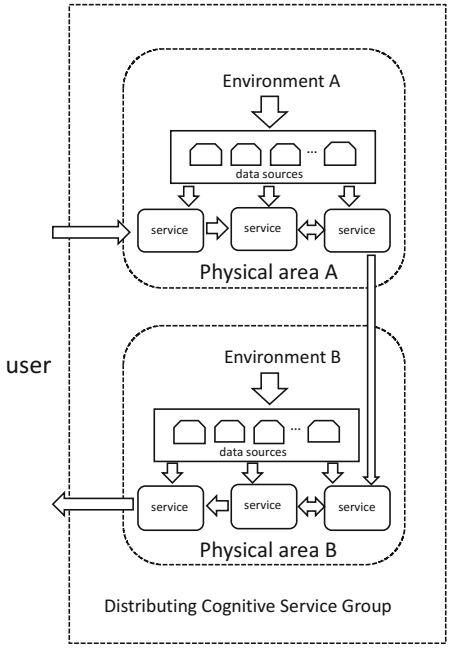

Fig. 2. DCSG topology

Based on this kind of service workflow, we define some essential concepts for cognitive service to match the service functional requirement as below and Fig. 1 illustrates the cognitive service structure we defined, further, based on this structure, we consider the data source location to divide the service area and form the distributing cognitive service group (DCSG) illustrated in Fig. 2.

Explicit Input: User need to provide the satisfactory input data, which is the main part of data used to comprehend the in-depth information for obtaining expected result.

Data Source and Implicit Input: Cognitive service can access and acquire the necessary and additional service context data from data source then fusion and form the implicit input to assist the reasoning process for enhancing the depth for service result.

Resource Node: Cognitive service need to be deployed on several resource nodes for using the computing and storage capability. Resource nodes popularly can be physical or virtual machines (e.g., server) and also can be containers (e.g., Docker). These nodes are usually managed by a cluster manager service.

Policy: Policy is the processing operation to obtain the expected result from the input data.

Flexible Output: Due to the quality variety of input information, cognitive service can conclude different result depth. So, the service output is very flexible.

3.2 A Goal-Driven Context-Aware Architecture for Distributing Cognitive Service Group

To achieve the goal, the service group needs to perceive the overall working progress. We use service context to recognize the exception and make a corresponding decision to handle it. The overall architecture is illustrated in Fig. 3.

The most important consideration about service context is the service environment definition and corresponding service context representation. Context modeling is a regular method to build a context space for representing the interaction environment. The range of service environment is limited to involve two fields: **service outside environment** and **service inside environment**. Service outside environment represents the service input data including explicit input, and implicit input, service inside environment describes the service runtime status and dependent resource status.

Two core service components which are **controller service** and **daemon service** provide the capacities of capturing the exception and deviation and making decisions to solve them. Daemon services deploy on the host platform to acquire the service inside environment context for perceiving and control the functional service. Controller service responds to all daemon service and has a global perception of the whole service group. If a service inside exception raises, the daemon service deployed on host platform locating the broken-down service will firstly respond and try to solve it following some predefined rules and reports the impact to controller service for judging the impact globally.

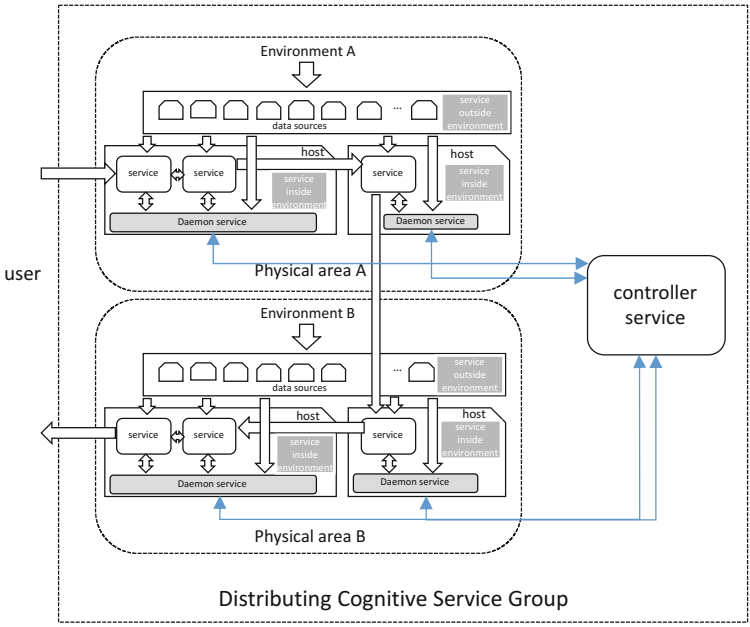

Fig. 3. A goal-driven context-aware architecture for distributing cognitive service group

Considering the encapsulation of third-party service, we decouple the dependency between perception capability and functional service inside. In this way, the existed service system can be easy to transfer to this novel architecture.

4 Challenge

The service architecture defined in our proposal may bring some challenges to our research. The most pressing challenge is when the controller service is down, the functional services will disconnect with the controller service. It means that functional services will lose the ability to perceive each other and controller service will lose the memory of service progress during its downtime. We need to present a progress retrieving approach to decrease this impact. Further, we are ready to evaluate the architecture performance by comparing with different state-of-the-art ones. So, the evaluation work is considered as the future work to be thought and finished.

5 Conclusion

In this paper, we address the problem of delayed response caused by exception event raised from the distributing cognitive service group. A novel architecture is proposed to ensure the overall consistency and real-time response of distributing cognitive service group. In the future, to evaluate the effect of it, we will develop a simple smart home system based on this architecture. Moreover, to solve a series of problems about controller service doesn't work, we will present a progress retrieving approach.

References

1. Microsoft Cognitive Service. https://azure.microsoft.com/en-us/services/cognitive-services/. Accessed 4 June 2018
2. Chan, A.T., Chuang, S.-N.: MobiPADS: a reflective middleware for context-aware mobile computing. IEEE Trans. Softw. Eng. **29**(12), 1072–1085 (2003)
3. Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. J. Netw. Comput. Appl. **28**(1), 1–18 (2005)
4. Santos, L.O., Poortinga, R., Vink, P.: A service-oriented middleware for context-aware applications. In: Proceedings of 5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (2007)
5. Jong-yi, H., Eui-ho, S., Sung-Jin, K.: Context-aware systems: a literature review and classification. Expert Syst. **36**(4), 8509–8522 (2009)
6. Eisenhauer, M., Rosengren, P., Antolin, P.: HYDRA: a development platform for integrating wireless devices and sensors into ambient intelligence systems. The Internet of Things, pp. 367–373. Springer, New York (2010). https://doi.org/10.1007/978-1-4419-1674-7_36

7. Kabir, M.A., Han, J., Yu, J., Colman, A.: SCIMS: a social context information management system for socially-aware applications. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 301–317. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31095-9_20
8. Khan, A.J., Jayarajah, K., Han, D., Misra, A., Balan, R., Seshan, S.: CAMEO: a middleware for mobile advertisement delivery. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, pp. 125–138. ACM (2013)
9. Charith, P., Arkady, Z., Christen, P., Dimitrios, G.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. **16**(1), 414–454 (2014)
10. Cabrera, O., Franch, X., Marco, J.: Ontology-based context modeling in service-oriented computing: a systematic mapping. Data Knowl. **110**, 24–53 (2017)