Event Calculus-Based Adaptive Services Composition Policy for AmI Systems

Huibing Zhang[1,2], Jingwei Zhang[1,2,*], Ya Zhou[2], and Junyan Qian[1]

[1] Guangxi KeyLaboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, Guangxi, China
[2] Research Center on Data Science and Social Computing, Guilin University of Electronic Technology, Guilin, Guangxi, China
zhanghuibing@guet.edu.cn, gtzjw@hotmail.com

Abstract. Services composition technology which is used in Ambient Intelligence should have the features of context-aware, partial order and concurrent. It should adaptive to the dynamic change of user preference and context. To meet these requirements, the paper puts forward an adaptive dynamic services composition framework and its implementation mechanism based on event calculus. It studies the basic principles and technologies for descripting domain services, context information and domain rules based on event calculus. On the basis, it details the composition planning mechanism. At last, a prototype system, intelligence application control, is built to verify the effectiveness and availability of the services composition policy.

Keywords: services composition, event calculus, context-aware.

1 Introduction

Ambient Intelligence (AmI) is a "user-centered" system which should provide kinds of services for users adaptively [1]. However, both system resource with feature of heterogeneous, dynamic and distribution and user requirements with feature of personalize, so it's difficult to meet the AmI's requirements [2]. This problem can be solved effectively by using the service-oriented computing (SOC) technology. Any accessible resource in AmI, such as program, sensor, device, can be modeled as Web service by using SOC, so the device-oriented physical space is transmitted into service-oriented information space [3-4]. The user requirement can be meet by compositing some services: according to the dynamic requirement and context, a composite service is created instance and discomposed it after completing the task [5].

Compare with other services composition application system, AmI is more emphasis on context-aware and personalization which should dynamic adjust its behavior to adaptive the special context and user preferences. At the same time, there are lots of concurrent operations and event-triggered activities. The system behavior is partial order in time dimension.So, the services composition framework for AmI should have

* Corresponding author.

R. Huang et al. (Eds.): AMT 2012, LNCS 7669, pp. 155–164, 2012.

the following features: ①It should have the good ability of context modeling which can model the run-time context, preferences and states constraint; ②It should have the good ability of reasoning which can describe states transition and time constraints; ③It should have the good ability of converting the Web service(OWL-S) into other forms which can be used to automatic services composition flexible and efficient. In order to meet these requirements, the paper puts forward an event calculus-based context-aware service composition framework and its implementation methods. It uses event calculus (EC)as formalize logic and adductive logic programming as reasoning technology to implement context-aware services composition [6-7].

2 Instruction of Event Calculus and Related Works

First-order predicate logic-based event calculus is suitable for description and analysis event-based time-varying domain [8-10]. As a programming framework, event calculus includes ontology, predicates and axioms [11]. The ontology contains events (actions), fluents and time-points which define the basic concepts, attributes, relationships and constraints. The fluents can represent anything which value or state may change with time, such as emotion, room temperature, lamp state (open/close), and so on. The actions can represent any operations which can lead to the change of world states. It can initiates, terminates or releases the values of fluents. Event calculus predicates define the ontology properties and its relationships. The 9 predicates and their meaning are shown in Table 1.

Table 1. Predicates of the Event Calculus

	Formula	Meaning
Base predicates	*Initiates(α, β, τ)*	*Fluent β holds after action α at time τ*
	Terminates(α, β, τ)	*Fluent β does not hold after action α at time τ*
	Releases(α, β, τ)	*Fluent β is not subject to the common sense law of inertia after action α at time τ*
	Initially $_P$(β)	*Fluent β holds from time 0*
	Initially $_N$(β)	*Fluent β does not hold from time 0*
	Happens(α, τ_1, τ_2)	*Action α starts at time τ_1 and ends at time τ_2*
	HoldsAt(β, τ)	*Fluent β holds at time τ*
Auxiliary predicates	*Clipped (τ_1, β, τ_2)*	*Fluent β is terminated between times τ_1 and τ_2*
	Declipped(τ_1, β, τ_2)	*Fluent β is initiated between times τ_1 and τ_2*

Shanahan pointed that Partial order planning = event calculus + abduction.Let Γ be a goal, let Σ be a domain description, let$\Delta_0$be an initial situation, and let Ω be the conjunction of a pair of uniqueness-of-names axioms for theactions and fluents mentioned inΣ, and let ψbe a finite conjunction of stateconstraints. A plan for Γ is a narrativeΔsuch that:

$$CIRC[\Sigma; Initiates, Terminates, Releases] \wedge CIRC[\Delta_0 \wedge \Delta; Happens] \wedge \psi \wedge EC \wedge \Omega \models \Gamma$$

Event calculus-based services composition planning is an open study domain. Ozorhan and Okutan introduced event calculus into service composition and implemented 4 types of service composition: Monolithic, Interleaved, Templatebased and Staged [12-13]. They detailed the methods which can transform Web service (OWL-S) into event calculus axioms (ECA). The article [14-15] proposed a method for formal verification of Web service composition by using event calculus. [16] adopted event calculus and workflow to realized services composition. Ishikawa and Chen researched the event calculus-based services composition from other aspects [17-18]. The above mentioned studies achieved good results at Web service modeling and verification based on event calculus. But, as far as we know, there are no study focusing on the AmI's complex context information modeling based on event calculus and using it in AmI system.

3 EC Based Services Composition Framework for AmI

According to the AmI system's special demands, the paper proposed a context-aware dynamic service composition framework based on event calculus. It includes 6 modules, as shown in Figure 1.

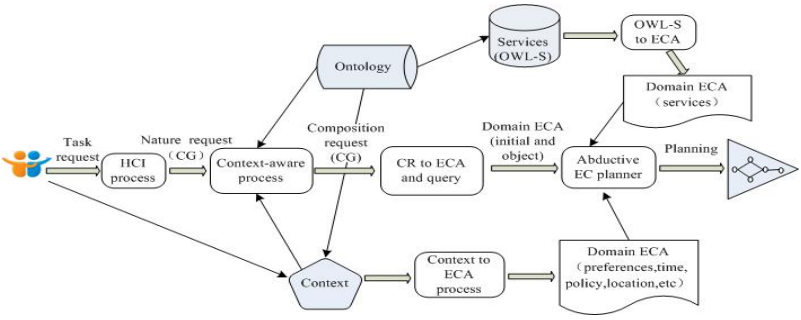

Fig. 1. Event calculus-based context-aware services composition framework

Natural human computer interaction and context-aware processing module implement intelligent interaction and context fusion. The former provides natural langue analysis based on conceptual graph. The latter combines context information into user basic requirement and gets a complete, definite requirement. Figure 2 denotes the processing of requirement "open TV". The module, CR to ECA, submits final requirement to planner.

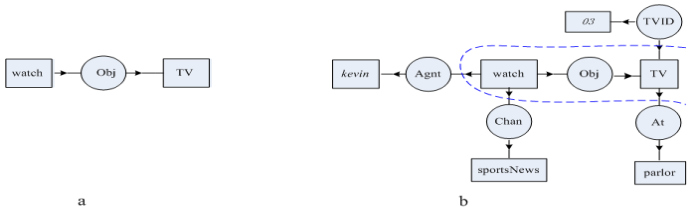

Fig. 2. Conceptual Graph of user's requirement

3.1 OWL-S to ECA

All of the resources deployed in AmI system modeled as Web services can be described as a 4-tuple:*WSFunctional=(Input, Output, Precondition, Effect).* Every service in service space can be mapped as an event in event space: Service invoking corresponds to event happens. Service's input parameters correspond to known parameters of event. Service's precondition corresponds to precondition of event happen. Service's output and effect correspond to fluents which are generated by event happen. The details can be seen inarticles [12-13].

3.2 Context to ECA

Context information is rich and it is the key factor for operating action in AmI system. Moreover, context is dynamically changing: ① All kinds of resource (services) evolve independently, and they may be unavailable momentarily. A better service may substitute the old one or new services are published. ② As user movies, available services are differences. ③Sometime-related context information changes with the time. So it is important to get, model, analyze and use context information in AmI environment. The paper focuses on the rules of each kind of context and their modeling based on event calculus axioms (ECA).

1. Location based service

Location information is an important context and it determines available service. Firstly, location determines the environment and spatiotemporal. Therefore, the location changes often leading to other related context change. Secondly, many services directly related to location. There are always different services at different locations. Last, location can influence system's operating action. For one requirement, the system may have different response because of different location. So it needs to update the context and domain services according to the current location. For example, TV available in the parlor or computer available in the study can be expressed follows:

axiom(initially(have(TV)),[holds_at(at(parlor),t)]).
axiom(initially(have(computer)), [holds_at(at(study),t)]).

2. Data input

In many case, context can be used as data input. System can transmit the data into service's input implicitly and user transparently. So it can effective reduce user inputs, improve user experience and reduce mistakes. For example, a user wants to browse a road map from current location to railway station. He only needs to input the phrase: view the map to train station. The system gets user's current location and train station location automatically, and then takes this information as input data. The context information which acts as input data is converted to known or initial state, and is added to event calculus domain axioms in form of fact clauses. It can be expressed as:

axiom(initially(at(person，location,)),[]).
axiom(initially(at(trainStation，location)),[]).

3. User preference and intelligent reasoning

The personalized of AmI system is mainly presented as user preference. Here, the preference refers to anticipatory affect or behavior disposition under special scenario. So it has significant individual difference and closely related to context. In order to meet user's special preference, AmI system dynamically adjusts its behaviors according to sensed context. Therefore, user preference is a key factor for intelligent reasoning which influences operating action. In event calculus based service planning, user preference can be acted as parameters or precondition of action. For example, a user like TV sports news after going home. It can be expressed as:

axiom(initially(known(channnelsportsNews),
[holds_at(at(parlor),t),holds_at(time(cur_time),t)]).
axiom(initially(known(channnel cartoon),
holds_at(at(parlor),t),holds_at(time(cur_time),t)]).
axiom(initiates(e_Open(Device,TVID,sportsNewts),
On(TVID,sportsNews),t),[holds_at(at(parlor),t),holds_at(time(cur_time),t),holds_at(known
(channnelsportsNews),t),
holds_at(known(channelcartoon),t),holds_at(neg(others(son)),t)]).

At last, user preference also can be acted as implicit requirement which expects some objects in a certain states. The implicit and explicit requirement consist a complete task requirement.

4. Trigger events

Some context can trigger related events which make AmI system adapt to environment. For instance, temperature or humidity can trigger events (actions) of air-conditioning in AmI space: While the temperature exceeded high_temperature, the air-conditioning opens refrigeration. While the humidity exceeded high_humidity, the air-conditioning opens dehumidification. It can be expressed as follows:

axiom(initiates(turnOnCool(x),CoolOn(x),t),
[holds_at(temperature(cur_temperature),t)，holds_at(higher
(current_temperature,high_temperature))，holds_at(neg(CoolOn(x)),t)]).
axiom(initiates(turnOnDehumidification (x),
DehumidificationOn(x),t),[holds_at(humidity
(cur_high_humidity),t)，holds_at(higher(cur_high_humidity ,
high_humidity)),holds_at(neg(Dehumidification (x)),t)]).

5. State constraints

State constraints are a kind of constraints relationship that similar to rules of object's fluents in domain. It reflects the mutual restriction among fluents during system running. In event calculus, state constraints are expressed as predicates holds_at and ¬holds_at. For instance, people will feel comfortable while the temperature and humidity maintained at a value. It can be expressed as follows:

axiom(holds_at(Comfortable,t),[holds_at(temperatureValue,t),holds_at(humidityValue,t)])

6. Precondition

Only meet all preconditions, the service can run and get right results. Context can provide some preconditions. In event calculus, it uses holds_at to express preconditions. For instance, TV should close if there is no human around it. It can be expressed as follow:

axiom(initiates(turnOff(TV),Off(TV),t), [holds_at(neg(at(personID,person_location)),t)]).

3.3 Abductive EC Planning

As practical execution module of service planning, abductive EC planner uses Prolog as its logic programming platform[1], event calculus and abductive theory prover (ATP) as its reasoning logic. It can deal with kinds of special requirement, such as concurrent, trigger, partial-order, continuous variation and so on, in AmI. Event calculus based service composition planning can be expressed as 4-tuple (*SEA, ECA, ConA, Q*), where *SEA i*s domain event calculus axioms which are converted from domain services. *SEA* is used to describe domain state altering or information transformation. *ECA* is domain independent axioms. It is the core of event calculus meta interpreter. *ConA* is domain axioms which get from context. It acts as initial states or state constrains. *Q* is target states which contain some query clauses and represent user requirement.

SEA, ECA and *ConA*constitute the event calculus based logic reasoning knowledge base*K*. Each clause in *K*represents an action, fact or rule. Using*Q* as the starting point of reasoning, it can get an event sequence or fail to exit by match, resolution, backtracking. The event sequence represents a service composition planning.

4 Experiments and Analysis

We apply the event calculus based service planning to our test-bed AmI-Space [1] and use intelligence application control (IAC) to test availability of the service planning. The physical deployment of IAC is shown in figure 3. Ten services are used in IAC:

TurnOnTV*(TVID, TVOff,TVOn);* ***TurnOffTV****(TVID, TVOn, TVOff);****ChannelSelection****((TVID, Channel), TVOn, Play on the selection channel);* ***TurnOnLight****(LightID, LightOff, LightOn);* ***TurnOffLight****(LightID, LightOn, LightOff);* ***TurnOnAircondition****(AirconditionID, AirConditonOff, AirConditoinOn);* ***TemperatureSet****((AirconditionID, Temperature, CurrentTemperature), ,AirConditionOn, AirConditoin On temperature);****TurnOffAircondition****(AirconditionID, AirConditonOn, AirConditinOff);* ***OpenCurtain****(CurtainID, CurtainClose, CurtainOpen);* ***CloseCurtain****(CurtainID, ,CurtainOpen, CurtainClose).*

4.1 Scenario and Experiments

Event calculus based service composition planning in IAC can be illustrated by following scenario.

[1] `http://www.swi-prolog.org`

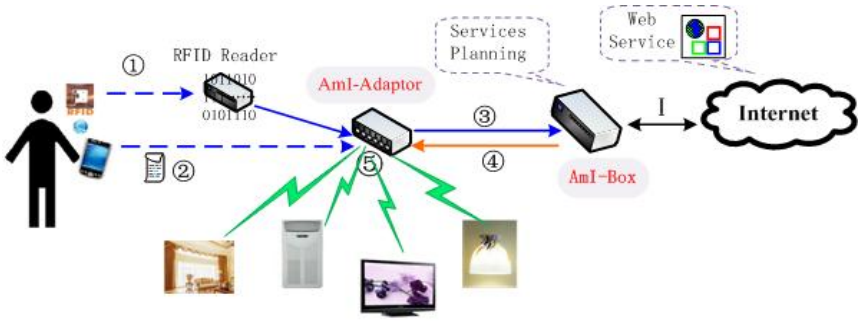

Fig. 3. Architecture of IAC

When comes home, Kevin sits on the sofa and lightly speaks out "open TV" by using his smart phone. Then TV opens automatically and selects the sports news. Curtain closes lightly and gentle light illuminate the parlor. The gentle breeze blows out from air-conditioning. …

The working process of this scenario as follows:

At the services planning side, it needs to convert domain services into domain axioms, as shown in Figure 4. All of the domain event calculus axioms, domain facts and rules composea complete domain axiom.

axiom(initiates(channelselection(Tvid,Channel),tvplay(Tvid,
Channel),T),[holds_at(tvon(Tvid),T)]).
axiom(initiates(turnontv(Tvid), tvon(Tvid), T), [holds_at(tvoff(Tvid),T)]).
axiom(initiates(turnofftv(Tvid),tvoff(Tvid),T), [holds_at(tvon(Tvid),T)]).
axiom(initiates(turnonlight(Lightid),lighton(Lightid),T),[holds_at(lightoff(Lightid),T)]).
axiom(initiates(turnofflight(Lightid),lightoff(Lightid),T),[holds_at(lighton(Lightid),T)]).
axiom(initiates(turnonaircondition(Airconditionid),
airconditionon(Airconditionid),T),[holds_at(airconditionoff(Airconditionid),T)]).
axiom(initiates(temperatureset(Airconditionid,Preferencetemperature),
temperature(Airconditionid,Preferencetemperature, Currenttemperature),T),
holds_at(airconditionon(Airconditionid),T),
diff(Preferencetemperature,Currenttemperature)]).
axiom(initiates(closecurtain(Curtainid),curtainclose
(Curtainid),T),[holds_at(curtainopen(Curtainid),T)]).
axiom(initiates(opencurtain(Curtainid),curtainopen
(Curtainid),T),[holds_at(curtainclose(Curtainid),T)]).
axiom(initiates(turnoffaircondition(Airconditionid),
airconditinoff(Airconditionid),T),[holds_at(airconditonon(Airconditionid),T)]).

Fig. 4. Services described by event calculus

At services requester side:

1. When Kevin enters AmI-Space, the system can get complete context information, such as time, location, preference, and so on. All these information are transmitted to the AmI-Box for modeling, analysis and storage by AmI-Adaptor.

2. Some of context information, for example TV or lamp's state and its attribute value are expressed as initial states or state constraints. The following event calculus axioms represent this information.
   axiom(initially(curtainopen(Curtainid)),[]).
   axiom(initially(tvoff(Tvid)),[]).
   axiom(initially(lightoff(Lightid)),[]).
   axiom(initially(airconditionoff(Airconditionid)),[]).
3. User speaks out his request "watch TV", then speech- to- text system converts the voice to text and sends to the CGGenerator in AmI-Box.
4. CGGenerator converts the user requirement to conceptual graph, and then generates a complete service request conceptual graph by using domain ontology and context information, as shown in figure 2.
5. Convert the conceptual graph based service request to event calculus query clause which is submitted to service planner. The figure 2.b is expressed as :
   ?-abdemo([holds_at(tvplay(01,sportsnews),t)],R).
6. In the AmI environment, preferences contain user's requirements which are the expected target states. So some preference information should be expressed as query of event calculus.
   ?-abdemo([holds_at(curtainclose(02),t),holds_at(temperature(04,22,29),t), holds_at(lighton(03),t)],R).
7. Combining query of step 5) and step 6), we can get a complete service request query which are submitted to planner. When planner receives the query, it returns an event sequences. Each event corresponds to a service. The system will reach at target state after executing each service according to the time sequence ordered in event sequence.

4.2 Results and Analysis

The planner returns a partial-order event sequences after it receives the query. Figure 5 shows the query and its planning results.

Query: *?-abdemo([holds_at(tvplay(01,sportsnews),t),holds_at(curtainclose(02),t), holds_at(temperature(04,22,29),t), holds_at(lighton(03),t)],R).*

Planning Results:

R=[[happens(turnonlight(lightid),t6,t6), happens(turnonaircondition(airconditionid), t5, t5), happens(temperatureset(airconditionid, preferencetemperature),t4,t4), happens(closecurtain(curtainid),t3,t3),happens(turnontv(tvid),t2,t2), happens(channelselection(tvid, channel), t1, t1)], [before(t6, t), before(t5, t4), before(t4, t), before(t3, t), before(t2, t1), before(t1, t)]] .

Fig. 5. Query and planning results

The planning results include 6 atomic services. The time sequence among these services is shown in Figure 6. It has 4 concurrent processes: process P2 and P3 are both single atomic services. Process P1 and P4 are both include two atomic services

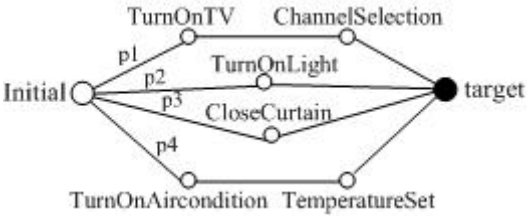

Fig. 6. Services composition corresponding to the planning results

which are sequence structure. From the planning results and its planning procedure we can see that event-based service composition planning method can meet AmI's demands:① It suitable for three control logic: concurrent, sequence and partial-order. ②It suitable both event logic based and state constraints services composition. Many input/output match based services composition algorithms can only suitable for information transmit service composition. But there are many states altering in AmI environment, the services often only have effects instead of output. Hence, it needs the event logic based services composition. ③Services composition procedure can naturally reflect user preferences and context constraints.

5 Conclusions

In this paper, we put forward a context-aware automatic service composition framework based on event calculus. It can improve the intelligence of the AmI system and reduce the complexity for AmI developers. And by using event calculus, the services composition mechanism can meet concurrent, partial-order, event state logic. Event calculus based context modeling technology can provide context-aware adaptive services composition. At last, it shows that the planning technology is available and effective by the IAC system. In the future, we will focus on the mechanism of domain axiom automatic generating technology. Another question is time efficiency of event calculus based reasoning.

Acknowledgments. The authors would like to thank the Foundation of Guangxi Key Laboratory of Trusted Software (kx201214, kx201203, kx201114),Nature Science Foundation of Guangxi (No. 2012GXNSFBA053171), National Nature Science Foundation of China (No.61063002,No.61063038),and the Education Department of Guangxi (No. 201010LX154)for their support incurrent research.

References

1. Chen, R., Hou, Y., Huang, Z., He, J.: Modeling the ambient intelligence application system: concept, software, data, and network. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 39(3), 299–314 (2009)

2. Lindenberg, J., Pasman, W., Kranenborg, K., et al.: Improving service matching and selection inubiquitous computing environments: a user study. Personal Ubiquitous Computing 11(1), 59–68 (2006)
3. Benazzouz, Y., Sabouret, N., Chikhaoui, B.: Dynamic Service Composition in Ambient Intelligence Environment. In: 2009 IEEE International Conference on Services Computing, pp. 411–418 (2009)
4. Martin, D., Burstein, M., et al.: OWL-S: Semantic Markup for Web Services (2010), http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.5.10
5. Medjahed, B., Bouguettaya, A., Elmagarmid, A.: Composing web services on the semantic web. The VLDB Journal 12(4), 333–351 (2003)
6. Shanahan, M.: An abductive event calculus planner. The Journal of Logic Programming 44, 207–239 (2000)
7. Shanahan, M.: The Event Calculus Explained (June 10, 2011), http://www.doc.ic.ac.uk/~mpsha/ECExplained.pdf
8. Kowalski, R.A., Sergot, M.J.: A Logic-Based Calculus of Events. New Generation Computing 4, 67–95 (1986)
9. Shanahan, M.P.: Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia. MIT Press (1997)
10. Barros, L.N.D., Santos, P.E.: The nature of knowledge in an abductive event calculus planner. In: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management, pp. 328–343. Springer, London (2000)
11. Shanahan, M.: The Event Calculus Explained. In: Veloso, M.M., Wooldridge, M.J. (eds.) Artificial Intelligence Today. LNCS (LNAI), vol. 1600, pp. 409–430. Springer, Heidelberg (1999)
12. Okutan, C., Cicekli, N.K.: A monolithic approach to automated composition of semantic web services with the Event Calculus. Knowledge-Based Systems 23, 440–454 (2010)
13. Ozorhan, E.K., Kuban, E.K., Cicekli, N.K.: Automated composition of web services with the abductive event calculus. Information Sciences 180(19), 3589–3613 (2010)
14. Rouached, M., Perrin, O., Godart, C.: Towards Formal Verification of Web Service Composition. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 257–273. Springer, Heidelberg (2006)
15. Rouached, M., Godart, C.: An event based model for web service coordination. In: Second International Conference on Web Information Systems and Technologies, pp. 81–88 (2006)
16. Aydın, O., Cicekli, N.K., Cicekli, I.: Automated Web Services Composition with the Event Calculus. In: Artikis, A., O'Hare, G.M.P., Stathis, K., Vouros, G. (eds.) ESAW 2007. LNCS (LNAI), vol. 4995, pp. 142–157. Springer, Heidelberg (2008)
17. Ishikawa, F., Yoshioka, N., Honiden, S.: Developing consistent contractual policies in service composition. In: Proceedings of the Second IEEE Asia-Pacific Service Computing Conference, pp. 527–534 (2007)
18. Chen, L., Yang, X.: Applying AI planning to semantic web services for workflow generation. In: The First International Conference on Semantics, Knowledge and Grid, pp. 323–325 (2005)