International Journal of Information Technology Vol. 21 No. 2 2015

User Behavioral Context-Aware Service Recommendation for Personalized Mashups in Pervasive Environments

Wei He, Guozhen Ren, Hui Li and Lizhen Cui

School of Computer Science and Technology,
Shandong University, Jinan, China

hewei@sdu.edu.cn

Abstract

With the rapid development of mobile Internet and increasing amount of smart devices, Internet services have been integrated into peoples' daily lives. Due to the features of end-user-oriented mashups in pervasive environments, new challenges have been presented to conventional mashup approaches, including the complexity of user behaviors, the difficulty of predicting real-time user preference and other dynamic contexts. In this paper, we propose a new paradigm for behavioral context-based personalized mashup provision in pervasive environments by integrating mashup construction and execution into user natural behaviors. In the proposed paradigm, users with similar behavior patterns are identified and then probability distributions of potential behavior selection for user clusters are discovered from historical mashup logs, which provide supports for predicting and recommending user activities for future mashup constructions. Analysis and experiments indicate that our approach can effectively simplify personalized mashup composition, as well as improve the quality of mashup composition and recommendation based on behavioral contexts and personalization in pervasive environments.

Keyword: Pervasive computing; Behavioral context; Mashup; Personalization; Service

I. Introduction

With the rapid development of mobile Internet and the increasing growth of smart devices, more and more Internet-based services are being closely integrated into end-users' daily behaviors, which definitely bring better experiences for users than traditional desktop environments. Analogously, in such pervasive scenarios, the construction and execution processes of end-user-oriented mashups are also integrated into user's natural behaviors with procedure and interaction features [1, 2]. Therefore, user behavior patterns should be considered during the process of constructing and selecting mashup solutions in order to improve user experiences. Together with personalized service provisions and dynamic contexts, these new features have presented great challenges to conventional mashup approaches.

The biggest challenge is the complexity of user behavior processes in pervasive environments. Different users always show various behavior patterns and preferences and it is difficult to perceive and predict real-time user behavior patterns in dynamic contexts. Personalized factors, including user habit and preference, have much impact on the selection of user behaviors. Even in the same context, different users have various preferences for activities and services. Furthermore, mobility of user devices and dynamic contexts increase the difficulties of user preference awareness. Let's consider an example of service mashups for dining out based on available web-based APIs, in which user activities probably include finding restaurant, reserving seats, locating and going to destination, parking and ordering, involving potential services such as LBS, map service, navigation service, reservation service and dish order service etc. Actually, there are multiple potential mashup solutions to meet the requirement and it is important to select a suitable one for the particular user. In conventional approaches, the mashup schema with involved activities is required to be defined and constructed in a mashup tool before it can be executed. However, it is both difficult and unrealistic for unprofessional users to schedule such a complete model in advance due to their limitations of professional knowledge and available information. Even though a mashup solution is pre-defined manually or automatically, probably it cannot achieve satisfactory results during the following execution period. Current context-based approaches for service composition and recommendation rarely consider the influence of user behavioral patterns and service relationships. Therefore, in order to improve user's experience, it is more crucial to help user planning a satisfying personalized mashup solution based on user behavioral contexts and preference, rather than only recommending independent services.

In this paper, we focus on the integrated cycle of both user behavior and mashup execution, instead of only recommending services. We propose a construction approach for personalized mashups based on user behavioral contexts. The main idea of this paper is as follows. First, potential preference-based user behavior patterns are discovered by applying pattern mining tasks to historical mashup logs. Then, based on the probabilistic distribution of user behaviors, according to user goal, user behavior traces and behavior patterns of similar users, target user's upcoming behaviors are predicted and the next activity is recommended. Next, the selected activity will be grounded to concrete Web-based APIs followed by executing the grounded service. By repeating the last two steps of the process, a personalized mashup schema is composed progressively until the final goal is achieved. In this proposed paradigm, mashup composition is simplified and end-user is not required to construct a schema in advance from scratch with necessary professional knowledge.

The rest of this paper is organized as follows: Section 2 gives problem definitions and the proposed system model; In section 3, we describe the detailed pattern mining approach based on historical logs. Section 4 depicts the iterative construction algorithm for end-user-oriented personalized mashups; In section 5, simulation experiments are illustrated; Section 6 gives related work; section 7 summarizes the main contributions of the paper.

II. Problem definitions and system model

A mashup instance represents a historical execution of mashup, which involves multiple components with specific context and execution sequence. The mashup log is the set of finite discrete mashup instances, each of which records a particular execution trace with component invocations and user contexts.

Definition 1 (Mashup Instance). A Mashup Instance is expressed as a tuple: mi = <u, g, ts>, with u denoting the end-user, g denoting user goal: g = <In, Out, Desc>, where In = {in1, in2, ...} is the set of all input parameters, Out = {out1, out2, ...} is the set of all output parameters of the mashup, and Desc = {kw1, kw2, ...} is the set of keywords describing the mashup functionalities. The third attribute ts denotes task sequence: ts = <t1, t2, . . ., tn>, where each task ti (1≤i≤n) is a tuple: ti=<mci, ctxi> with mci denoting mashup component and ctxi denoting the context of current task. In the task sequence, the parameters required by component mci come from the outputs generated by one or more precursor components mci-1, mci-2, …mc1 (1≤i≤n).

In the scenario of pervasive mashups, conventional context is extended with user behavior information. User behavioral context records the behavior trace (or activity sequence) he/she has performed during the period of mashup execution.

Definition 2 (Behavioral Context). A Mashup Context is a prefix subset of the task sequence mi.ts of a mashup instance, which is expressed as: bc = <t1, t2, . . ., tm>, m⩽n, where each task ti is a tuple: ti=<mci, ctxi> with mci denoting mashup component and ctxi denoting the context of current task.

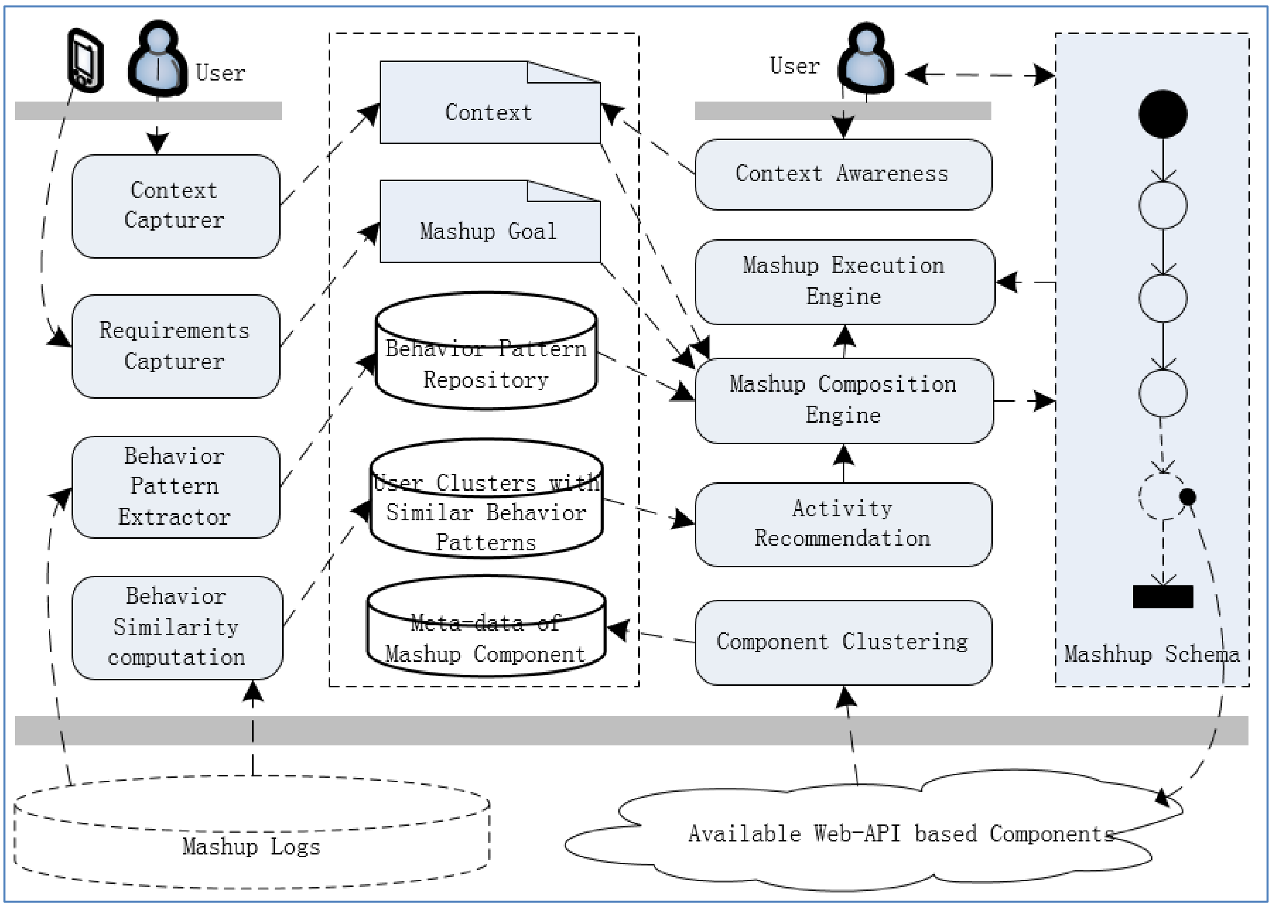
User
Context Capturer
Requirements Capturer
Behavior Pattern Extractor
Behavior Similarity computation
Context
Mashup Goal
Behavior Pattern Repository
User Clusters with Similar Behavior Patterns
Meta-data of Mashup Component
User
Context Awareness
Mashup Execution Engine
Mashup Composition Engine
Activity Recommendation
Component Clustering
Mashhup Schema
Mashup Logs
Available Web-API based Components

Fig. 1. System structure

The architecture of user behavioral context-based mashup construction and service recommendation is illustrated in figure 1, with multiple components including behavior pattern mining module, context awareness module, knowledgebase of behavior patterns and mashup composer & executer. Historical logs record users' mashup activities, including behavior traces, involved web resources and the related contexts during the execution period, from which quite a few valuable information for future mashup construction can be discovered. User behavior pattern extractor generates the probability distribution for user activity selection. Based on the activity traces in which different users are involved, user behavior similarity computing module evaluates user preferences to candidate activities in particular contexts and partial activity traces, so that users with similar behaviors are identified. Component clustering module generates activities by clustering actual web-based components with similar functionalities.

Context awareness module is responsible for capturing and updating user behavioral contexts in current mashup execution, as well as other real-time contexts which may affect the selection of user activities, such as location and time.

Mashup composer and executer are the core parts of the system including activity recommendation engine, schema composing engine and service execution engine. According to user goal and user's previous behavior traces, activity recommendation engine computes user's preference values for candidate activities based on the extracted probability distribution of user behavior and recommend the next activity. Then, composition engine enhances current mashup schema by combining the recommended activity. Execution engine grounds the activity with available web-based component and perform execution. The result of execution engine will feed back to composition engine as one of the conditions to generate the next steps of the mashup.

The above framework provides an effective approach for constructing a preference-based mashup instance using an iterative paradigm. In the following sections, we will discuss some key issues in the proposed system.

III. Log-based behavior pattern mining for mashups

A mashup instance records a historical execution trace involving user, services with a running sequence and contexts. Note that the mashup instances are independent and isolated with each other in historical logs, even for similar mashup goals and contexts.

Definition 3 (Mashup Activity). A Mashup Activity, also called abstract component, describing the common attributes of a set of concrete web-based components with similar functionalities, which is expressed using a tupole: ma = <Op, In, Out, Desc>, with Op denoting operation name, In = {in1, in2, ...}, Out = {out1, out2, ...} denoting input and output parameters respectively, and each of the parameters is a tuple: p = <name, Desc>, where $p \in In \cup Out$, Desc is the keyword set of parameter annotations: Desc={anno1, anno2, …}.

Definition 4 (Mashup Schema). Assuming A is a set of activities, C is a set of contexts and B is the Cartesian product of A and C: B= A × C. Let B* denotes the set of finite sequences of B. A Mashup Schema σ is a finite sequence of activities in B*,σ∈B*, which can be expressed as: σ = <(a1,c1), (a2,c2),……, (an,cn)>, σ[i]= (ai,ci), |σ|=n is length of the sequence, 1≤i≤n.

A mashup schema is an abstract template describing multiple mashup instances with similar contexts and behavior patterns. As the basic component of mashup schema, activity extraction becomes one of the fundamental tasks in mashup pattern mining. At present, there have been quite a few researches on service clustering and classification. We adopt a component clustering approach based on functionality similarity and interface compatibility, which is described in detail in our previous works [11].

A. *Similarity of user activity traces*

Let σ be a complete mashup instance in historical logs, σt denotes the activity sequence of σ. We define the activity adjacency relation of σt as: AAR(σt) = {<ti,ti+1>}. Also, Let ADR(σt) denotes the activity dependence relation: ADR(σt) = {<ti,tj>}, i<j. Therefore, the activity adjacency relation and activity dependence relation are two types of decomposition for an activity sequence with different rigorous levels.

Definition 5 (Activity Trace Similarity). Letσ1, σ2 represent two arbitrary activity sequences: σ1=<t1, t2, . . ., tn>,σ2=<t'1, t'2, . . ., t'm>, the Activity Trace Similarity ofσ1 and σ2 is defined as

$$Sim_t(\sigma_1,\sigma_2) = \frac{|ANR(\sigma_1) \cap ANR(\sigma_2)|}{|ANR(\sigma_1) \cup ANR(\sigma_2)|} * w + \frac{|ADR(\sigma_1) \cap ADR(\sigma_2)|}{|ADR(\sigma_1) \cup ADR(\sigma_2)|} * (1-w) \quad (1)$$

Where 0≤w≤ 1, is a weight value for measuring the importance of activity adjacency relation and activity dependence relation.

B. *Similarity of user behavior patterns*

Collaborative filtering recommendation approaches based on similar users have been proven significant effectiveness in multiple domains. Similarly, introducing the factor of user preference in personalized mashup construction will definitely improve the quality of activity recommendation for end-users. In this scenario, users are considered similar if they have similar behavior patterns for the same goal.

Definition 6 (User Preference for Activity Trace). The preference of user U for an activity trace is expressed as a tuple: EP=(P, δ) with P={P1, ……,Pn} denoting the finite set of the activity traces that user U has selected, δ(Pi) denoting the preference value for activity trace Pi which is the times that U has performed Pi in historical logs.

Based on the matrix of user preference for activity traces, the behavior pattern similarity between any users can be measured. Among current approaches for measuring user similarity, Pearson correlation coefficient [4] has been proved to be effective enough in multiple domains. In our approach, Pearson correlation coefficient is used to measure the preference similarity for activity traces between user u and v:

$$r_{uv} = \frac{\Sigma_{i} \in I_{uv}(P_{u,i} - \overline{P_u}) \times (P_{v,i} - \overline{P_v})}{\sqrt{\Sigma_{i} \in I_{uv}(P_{u,i} - \overline{P_u})^2} \sqrt{\Sigma_{i} \in I_{uv}(P_{v,i} - \overline{P_v})^2}}$$

Where ruv denotes the preference similarity of user u and v, Iuv means the common activities in the traces they selected, Pu,i, Pv,i is the preference value of user u, v for activity trace Pi, and $\overline{\text{P}}$u, $\overline{\text{P}}$v denotes the average preference value of user u, v for their common activities Iuv.

Definition 8 (Behavior Pattern Similarity). Assuming the preferences of user Ui and Uj for activity traces are EPi=(Pi,δi) and EPj=(Pj,δj) respectively, the behavior pattern similarity between user Ui and Uj is defined as

$$A_{ui \Leftrightarrow uj} = \begin{cases} r_{ij}, & r_{ij} \geq 0 \\ 0, & else \end{cases} \qquad (2)$$

Where rij is the Pearson correlation coefficient of the preference value for the common activity traces of user Ui and Uj.

IV. Incremental construction algorithm for mashup schema

The mashup construction is a progressive procedure involving two phases in each iteration: schema construction phase and service execution phase.

A. *Support Function for activity trace*

Based on user partial activity traces, historical mashup instances provide different supports for recommendation of the user's future activities. That is, the more similar the trace fragment and the historical mashup instances are, the more valuable the historical patterns are to support the construction of current mashup.

Let σ be the activity trace of a mashup instance in historical logs, σp denotes the prefix sub-sequence of σ, ρ is a partial execution trace (i.e. mashup fragment), G(σ) and G(ρ) denote the goal of σ and ρ respectively. Then we define the **Support Function** of mashup instance σ for partial execution trace ρ, i.e. the probability that activity trace evolves into mashup instance σ:

$$S(\rho,\sigma) = Sim_t(\rho,\sigma) * w + S_g(\rho,\sigma) * (1-w) \qquad (3)$$

Where $Sim_t(\rho,\sigma)$ is the activity trace similarity defined in formula (1), and Sg(ρ, σ) denotes mashup goal similarity which is defined as

$$S_g(\rho,\sigma) = \frac{|g(\sigma).\mathrm{Out} \cap g(\rho).\mathrm{Out}|}{|g(\sigma).\mathrm{Out} \cup g(\rho).\mathrm{Out}|}$$

The support function for activity trace is used to filter the historical mashup logs, so that the instances supporting the partial execution trace can be identified and the other unrelated mashup instances are excluded.

B. *Mashup construction algorithm*

In the following we describe the schema construction phase of the iterative mashup composing process, i.e. the algorithm for activity recommendation based on user past behavior trace.

```
**Algorithm** : Behavioral context-based activity recommendation
**Input**: mashup fragment $f$, mashup goal $g$, user $u$, threshold of mashup similarity $h$
**Output**: enhanced mashup fragment $\hat{f}$.
Let $f$ = ($<s_1, c_1>, <s_2, c_2>, <s_3, c_3>$ ……, $<s_k, c_k>$);
$ssi$ = {}; //Initialize the set of similar mashup instances of $f$;
For each mashup instance $mi$ in the historical logs
  Compute similarity $S(f, mi)$ using formula (4);
  If ($S(f, mi) \geq h$)
```

```
    $ssi = ssi \cup \{mi\}$;
  End if
End for
Identify user cluster $o$ for user $u$ based on formula (2); //$u \in o$
$r$[0..$c$]=0; //The selection times of candidate activities by users in $o$;
For each user $\hat{u}$ in user cluster $o$
  For each candidate activity $ca[i]$
    $r[i] = r[i]$ + the times of $ca[i]$ that user $\hat{u}$ has selected in logs;
    End for
  End for
 Assign $ca[k]$ to $na$ where $r[k]$ is the greatest value in $r$[0..$c$];
$\hat{f} = f \cup <na>$;
Return $\hat{f}$;
```

The algorithm describes schema construction phase of the iterative mashup composition process, which aims to enhance the mashup fragment by computing and recommending the next activity. The complete mashup can be constructed progressively by performing the algorithm repeatedly. Once the recommended activity is confirmed, it should be grounded to a particular Web-API based component registered in the repository. Currently, there have been quite a few researches focusing on mashup service selection and recommendation.

V. Simulation Experiments

According to the effectiveness and efficiency of the mashup composition approach, an application scenario of service mashups are constructed, in which dining-out related services are provided for end-users. In this application scenario, a user goal may be described as “Finding a restaurant within 10 miles, getting there, parking and having dinner with my friends” with probably multiple web-based services.

A. *Experimenting data*

1. Web-based components

The detailed construction process involve 2 steps. In the first step, real-world services on Internet are searched and extracted to generate the primary components in the registry from the sources of general service providers (such as Google Place APIs, Baidu mapping APIs etc.) and specific platforms(such as Yelp APIs, Dianping.com APIs etc.). The descriptive information for both functionalities and interfaces of these public services are extracted, and then annotations are generated for each component based on its native descriptions. Besides, some QoS attributes are randomly generated, including usability, performance and reliability etc. In the second step, we also created and annotated virtual services with a random number for each primary component based on its meta-data, so that the registered components come to a certain quantity. The basic statistic info of the constructed components is listed in table 2.

Table 1. The statistics of components

Total number	372		
Clusters based on functionality similarity	15		
Number of annotations	1, 3, 5		
Component classification	**Number of components**	**Primary components**	**Virtual components**
Total	372	51	321
LBS-based service	19	3	16
Navigating service	18	3	15
Restaurant finding	44	9	36
Reservation service	26	5	21
…………	……	……	……

2. Historical mashup instances

Based on the meta-data of generated components, we constructed simulation data for historical user behavior traces, i.e. mashup instances. The mashup instances were created by randomly selecting components following the constraints of compatible interfaces and functionalities. Then, the generated instances were adjusted from two aspects. One adjustment is to increase clustering property for the behavior traces of similar users. On the other side, a percentage value measuring component popularity was introduced for each component and let the appearance frequency of components in the generated logs roughly follow their popularity distribution, so that the actual situation could be reflected as much as possible. The statistics of generated mashup instances is listed in table 3.

Table 2. The statistics of generated mashup instances

Set of mashup instances	
Number of mashup instances	3500
Number of involved users	60
Number of involved components	317

Estimated number of activities	29
Number of mashup goals	12
Number of contexts	5

B. *Experimental Results*

First, experiments are performed to verify the component clustering approach based on functionality similarity and interface compatibility. The experimental results of component clustering in different conditions are described in our previous works [11]. In this paper, we also compared our component clustering approach, referred to “FSandIC-based clustering” with other popular clustering algorithms. In literature [6], the authors proposed two service similarity computing approaches: Euclidean-distance and Cosine-distance measurement based on vector space for multi-dimensional properties, which is referred to “ED-based clustering” and “CD-based clustering” respectively. For similar purpose, Literature [7] proposed functionality-based and process-based similarity measurements for component clustering which is referred to “FS-based clustering”. The difference degree between our clustering method and the other 3 approaches were computed based on service similarity matrices generated by the 4 methods, including difference value for service pair $<s_i, s_j>$ defined as:

$$D(s_i, s_j) = d_1(s_i, s_j) - d_2(s_i, s_j),$$

and the overall difference value define as

$$D(M_1, M_2) = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(d_1(s_i, s_j) - d_2(s_i, s_j))^2}{n(n-1)/2}}$$

where $d_1(s_i, s_j) \in M_1,\ d_2(s_i, s_j) \in M_2$ denotes the similarity value between service s1 and s2 in matrix M1 and M2 respectively. The difference degrees of clustering between our method and the other 3 approaches, as well as the combined comparison, are shown in figure 2. The results indicate that the clustering result of our method is close to FC-based clustering with an overall difference of 1.9, and has much bigger difference with the other approaches. This is because the two closer clustering methods considered both functionality and interface compatibility, and the other 2 methods focus on multi-dimensional spaces including functional semantic, location info and QoS attributes.

The next experiment was performed to verify the effect of mashup construction. The generated mashup instances were divided into 2 parts: sample instances and benchmark instances. Experiments were carried based on sample data to generate recommended mashups, then the results were compared with the benchmark part of mashup instances with similar goal and contexts.

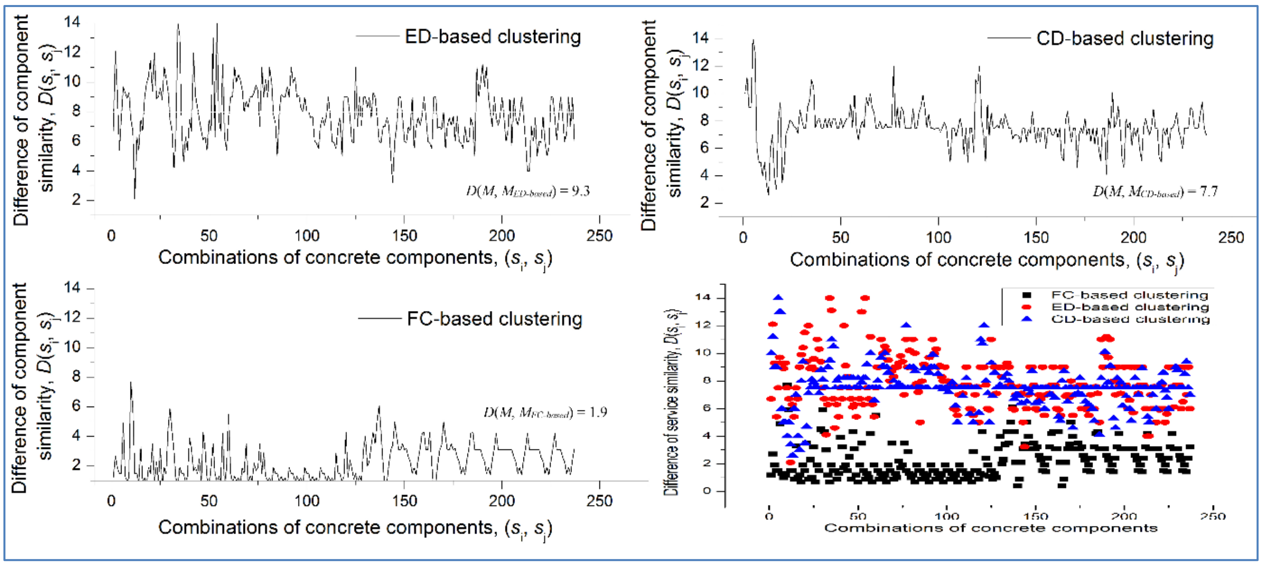

Fig. 1. Clustering differences among the approaches

In order to measure the results of mashup construction, we define matching rate according to benchmark data with mashup goal g and user u:

$$mr(g,u) = \frac{\sum_{c_i \in C} |S_b(g,r,c_i) \cap S_r(g,r,c_i)|}{\sum_{c_i \in C} |S_b(g,r,c_i)|},$$

where Sr(g, r, c) denotes the set of recommended components generated in the experiments, Sb(g, r, c) is the components in the benchmark instances.

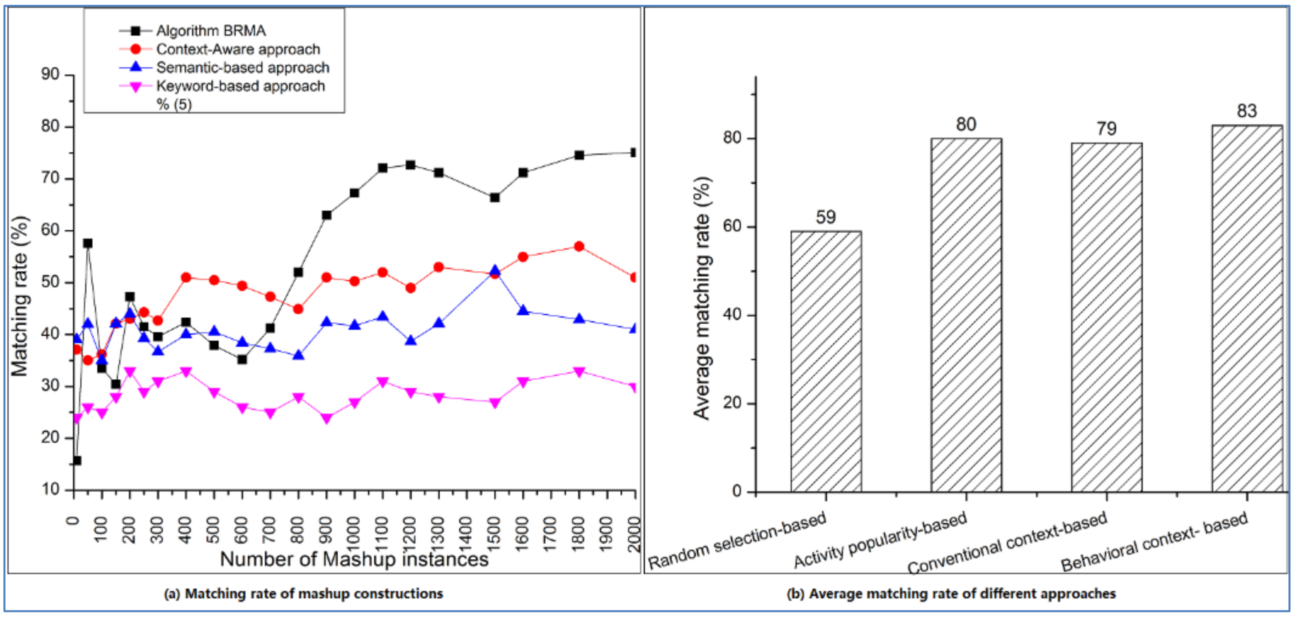

Fig. 2. Matching rate of mashup constructions

Based on the definition of matching rate, we compared our algorithm with other service composition approaches according to different numbers of mashup instances. We investigated some representative approaches of service composition, including context-awareness based [5], pervasive-environments based [3] and planning-based[8] composition. We simplified these methods and implemented the core ideas, then compared the results with our algorithm. Figure 3(a) illustrates the matching rate of the algorithms with different number of generated mashups. With the increasing number of generated mashup instances to be evaluated, the matching rate with benchmark data becomes stable. Another experiment in figure 3(b) shows the average matching rate for different approaches.

VI. Related Work

In recent years, context-based service recommendation for mashup provisions have been attracted much attention. Some researchers performed role mining and service recommendation based on users' historical selection records in various physical environments [9]. Hussein et al. capture a service's requirements as two sets of scenarios, and then deal with the dynamic changes of contexts using adaptation requirement [10]. Due to the successful effects in many other domains, personalized recommendation was widely introduced into service discovery and selection. Meng et al. proposed a similar user-based CF approach to recommend services by annotating users' preferences with keywords [12]. Chen et al. adopted a visualized technology to improve recommendation comprehensibility and implemented personalized recommendation based on a CF algorithm [13]. To improve recommendation qualities in pervasive environments, quite a few researches combine contexts into collaborative filtering methods. Shin et al. proposed a CF-based recommendation approach with aggregated contexts [14]. Karatzoglou et al. constructed a multi-dimensional matrix of "user-item-context" by extending conventional "user-item" matrix for a context-aware recommendation approach [15].

VII. Conclusion

In this paper, we propose a new paradigm for behavioral context-based personalized mashup provision in pervasive environments by combing user behavior and mashup instance execution into an integrated process. Analysis and experiments indicate that our approach can effectively simplify personalized mashup composition without depending on end-users' professional knowledge, as well as improve the quality of mashup composition and recommendation based on behavioral contexts and personalization in pervasive environments.

VIII. Acknowledgement

This work is partially supported by NSFC (No.61572295), the Innovation Method Fund of China (No.2015IM010200), SDNSFC (No.ZR2014FM031), the Science and Technology Development Plan Project of Shandong Province (No.2014GGX101019, 2014GGX101047, 2015GGX101007, 2015GGX101015), the Shandong Province Independent Innovation Major Special Project (No.2015ZDXX0201B03), the Fundamental Research Funds of Shandong University (No.2015JC031).

References

[1] Daniel F, Koschmider A, et al. Toward process mashups: key ingredients and open research challenges. Proceedings of the 3rd and 4th International Workshop on Web APIs and Services Mashups, p.1-8, 2010

[2] Fisichella M, Matera M. Process flexibility through customizable activities: A mashup-based approach. 2011 IEEE 27th International Conference on Data Engineering Workshops, 2011:226 - 231.

[3] Zhou J, Gilman E, Palola J, et al. Context-aware pervasive service composition and its implementation. Personal and Ubiquitous Computing, 2011, 15(3):291-303

[4] Good N, Schafer JB, Konstan JA, et al. Combining collaborative filtering with personal agents for better recommendations. In: Proc. of the 16th National Conf. on Artificial Intelligence. Menlo Park: AAAI Press, 1999. 439−446

[5] Medjahed B, Atif Y. Context-based matching for Web service composition. Distributed and Parallel Databases, 2007, 21:5-37

[6] Platzer C, Rosenberg F, Dustdar S. Web service clustering using multidimensional angles as proximity measures. ACM Transactions on Internet Technology. 2009, 9(3): 1-26

[7] Sun P, Jiang C. Using service clustering to facilitate process-oriented semantic web service discovery. Chinese Journal of Computers, 2008, 31(8): 1340-1353

[8] Hatzi O, Vrakas D, Nikolaidou M, et al. An Integrated Approach to Automated Semantic Web Service Composition through Planning[J]. Services Computing, IEEE Transactions on, 2012, 5(3):319 - 332.

[9] Wang J, Zeng C, He C, et al. Context-aware role mining for mobile service recommendation. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. New York: ACM, 2012. 173-178

[10] Hussein M, Han J, Yu J, et al. Scenario-Based Validation of Requirements for Context-Aware Adaptive Services. In: Proceedings of the IEEE International Conference on Web Services. New York: IEEE Press, 2013. 348-355

[11] He W, Li Q, Cui L, et al. A Context-Based Autonomous Construction Approach for Procedural Mashups[C]. //Web Services (ICWS), 2014 IEEE International Conference on. IEEE, 2014:487 - 494.

[12] Meng, S, Dou W, Zhang X, et al. KASR: A Keyword-Aware Service Recommendation Method on MapReduce for Big Data Application. IEEE Transactions on Parallel and Distributed Systems, 2014

[13] Chen X, Zheng Z, Liu X, et al. Personalized QoS-aware web service recommendation and visualization. IEEE Transactions on Services Computing. 2013, 6(1): 35-47

[14] Shin D, Lee J, Yeon J, et al. Context-aware recommendation by aggregating user context. In: IEEE Conference on Commerce and Enterprise Computing. New York: IEEE, 2009. 423 - 430

[15] Karatzoglou A, Amatriain X, Baltrunas L, et al. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In: Proceedings of the fourth ACM conference on Recommender systems. New York: ACM, 2010. 79-86