Hindawi
Mobile Information Systems
Volume 2017, Article ID 3254908, 13 pages
https://doi.org/10.1155/2017/3254908

Research Article

A Process Mining Based Service Composition Approach for Mobile Information Systems

Chengxi Huang, Hongming Cai, Yulai Li, Jiawei Du, Fenglin Bu, and Lihong Jiang

School of Software, Shanghai Jiao Tong University, Shanghai, China

Correspondence should be addressed to Hongming Cai; hmcai@sjtu.edu.cn

Received 23 September 2016; Revised 29 November 2016; Accepted 18 December 2016; Published 23 January 2017

Academic Editor: Laurence T. Yang

Due to the growing trend in applying big data and cloud computing technologies in information systems, it is becoming an important issue to handle the connection between large scale of data and the associated business processes in the Internet of Everything (IoE) environment. Service composition as a widely used phase in system development has some limits when the complexity of relationship among data increases. Considering the expanding scale and the variety of devices in mobile information systems, a process mining based service composition approach is proposed in this paper in order to improve the adaptiveness and efficiency of compositions. Firstly, a preprocessing is conducted to extract existing service execution information from server-side logs. Then process mining algorithms are applied to discover the overall event sequence with preprocessed data. After that, a scene-based service composition is applied to aggregate scene information and relocate services of the system. Finally, a case study that applied the work in mobile medical application proves that the approach is practical and valuable in improving service composition adaptiveness and efficiency.

1. Introduction

Along with the rapid advancements in big data and cloud computing technologies, connection of everything is emphasized in many information systems. Thanks to the achievements of devices, infrastructure, and applications in mobile computing [1, 2], systems become more powerful and intelligent with the support of connection among devices, people, and business processes. Particularly, according to the recent research [3], mobile technology development has resulted in the creation of up to 1450,000 applications for smart phones in the last few years. More and more information systems rely on service-oriented processes in order to fit the continually changing business environment and to align business strategies with IT systems [4]. With strong interaction with people and social environments, these systems have a great impact in many areas such as health care [5, 6], exploiting indoor location [7], and other scenarios. As a result, it is becoming more and more valuable to deal with the connection among devices and interaction among people especially in the environment of the Internet of Everything (IoE).

Due to the flexible and scalable characteristics of service-oriented computing, more and more systems use web services composition to deal with the complexity of multisource data in mobile information systems. Business processes and associated services become the most significant supports for the connection of everything. They make functions and devices work as expected in well-organized systems. Achieving adaptiveness in process-based service composition is the key to improve efficiency and adaptiveness of mobile systems.

However, as both the scale and the variety of devices are expanding, the complexity of service implementation is increasing. To sum up, challenges exist in keeping the system process adaptive to the changing environment as the following points:

(1) Process execution environment is changing: in the environment of IoE, as users, devices, and services are widely distributed, the execution of the process may be affected by changing device rules, connection situations, and event users' habits. As more complex rules are introduced with the devices, static processes always lack the consideration of execution environment, and they cannot handle the changing environment efficiently. For instance, in mobile systems, different versions of applications are used at the same

time, which will make the processes in the server side suffer from errors if they cannot handle the changing orders of events.

(2) The complexity of relationship in events and services is increasing: since types of devices are increasing, the relationship in events and services is getting more complicated. Current process-based service composition is not flexible enough to support the complex situations. As a result, approaches designed for application execution are usually incomplete and lacking necessary business consideration. For example, in a smart house application, when new devices like new models of air conditioners are introduced, new events and new connections will be introduced and the controlling process should be fixed accordingly in order to keep the devices and services work correctly.

In our previous work [8], the service composition based on process mining approach has been applied to a logistics cloud service platform which supports the users from different companies to customize their functional services. In the example case about the waybill transportation process, a suitable waybill-related composite service is generalized to connect the information sensing devices like radio frequency identification (RFID), infrared sensors, global positioning system (GPS), and laser scanner. And it is proved that service composition based on process mining is suitable for the situation with indefinite requirements and without high performance demand of the result composite service. Considering the expanding scale and the variety of devices in mobile information systems, a process mining based service composition approach is proposed based on our previous work in this paper in order to improve the adaptiveness and efficiency of compositions.

Generally speaking, the main contributions in this paper can be summarized as follows:

(i) Firstly, to solve the problems above, process mining based service composition is proposed to produce adaptive service composition according to real execution information. A three-step framework is presented to cover the whole life cycle of service composition based on process mining.

(ii) Secondly, according to the framework, a set of models is put forward to support the holistic service composition approach which covers both the practical business and the execution effectiveness.

(iii) Then, to apply request-based logs in event-based process mining, a preprocessing algorithm is presented to transfer request-based logs to event-trace-based models so that the execution data can be used in process mining.

(iv) Last but not least, a scene-based service composition algorithm is presented in order to transfer the process mining results to service composition models which can be further used in service generation.

The remaining parts of the paper are organized as follows: in Section 2, an overall description of the proposed approach is provided. After that, the formal analysis and algorithms in context-based service matching is described in Section 3. And then a case study is presented to validate the method in this approach in Section 4, followed by a brief discussion and comparison of the related works in Section 5. Finally, conclusions and future works are given in Section 6.

2. Overview of Process Mining Based Service Composition

In the environment of IoE, large amounts of event-based devices are involved in information systems. Each of them has individual rules due to the differences in types of devices, users, and execution context. In certain situation, they invoke a set of services to provide and retrieve data as well as execute special functions. Behind the devices, the server-side business processes which represent the sequences of service execution and service composition take the role to ensure the functional correctness of the whole system in either explicit or implicit way.

Process mining [11] is a process management technique that extracts information from event logs recorded by an information system to discover, analyze, and enhance process models. Service discovery mining is one of the most potential applications of the state-of-the-art process mining technologies [12]. It includes discovering service behavior, checking conformance of service, and extending service model based on event data. The processes discovered by process mining can provide the best practices during the execution period. The discovered processes with frequently used services can be regarded as composite web service patterns to help the developing of service composition. To improve the fitness of event rules applied in widely spread devices and the business process maintenance in information systems, three concerns are involved, namely, the execution log from IoE environment, control flow analysis for server side, and the service composition.

In order to cover the life cycle, three phases in process mining based service composition are proposed in this paper, as shown in Figure 1.

First, the approach preprocesses the execution data from current system by extracting the service logs and transforms them into valid traces. Then we leverage process mining algorithms to mine the control flow with the result of the previous step. After that, a metamodel is designed to connect the information of execution environment existing business rules, and service deploy model is generated after relocation the service mapping. The description of the steps is as follows:

(i) The first phase is to preprocess device request services:

(a) input: service invocation records, event rules;
(b) output: Trace Model.

(ii) The second phase is to mine process from event traces:

(a) input: Trace Model;
(b) output: Control Flow Model.

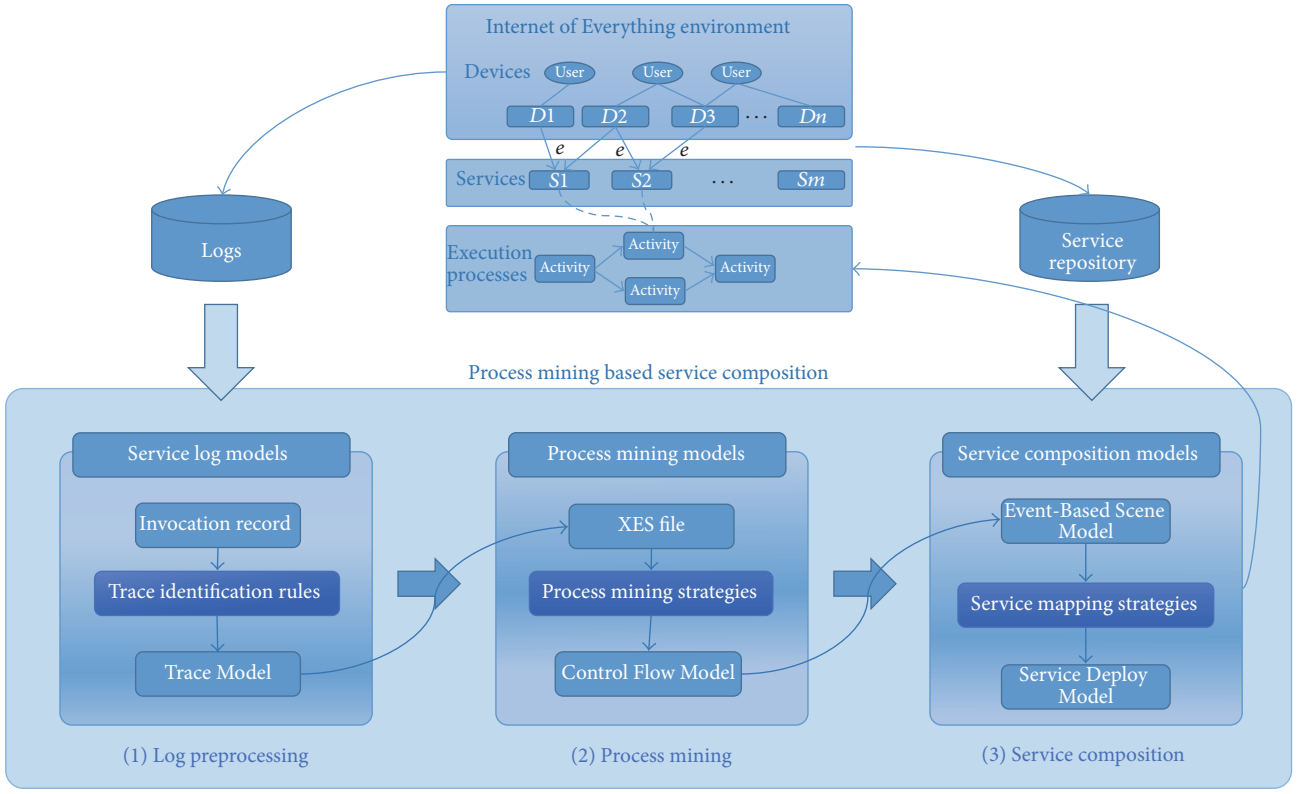


Figure 1: Framework of process mining based service composition, which is divided into three gradual phases: (1) log preprocessing, (2) process mining, and (3) service composition.

(iii) The third phase is to assist service composition with the produced control flow:

(a) input: Control Flow Model, service list;

(b) output: Service Deploy Model.

With execution information retrieved by preprocessing log data, the approach produces a service deploy model for constructing service compositions that is more accurate to the requirement in IoE. Afterwards, new logs will be recorded during the execution of the composite service; therefore the whole life cycle of the service composition procedure becomes a closed loop.

3. Process Mining Based Service Composition

In the following part, the framework mentioned above will be refined to introduce its specifics.

3.1. Models for Process Mining Based Service Composition. A set of models are defined in order to cover the life cycle of process mining based service composition in the three phases of the approach. Figure 2 shows three sets of models and their relationships involved in our approach, including Service Log Models, Process Mining Models, and Service Composition Models.

3.1.1. Service Log Models. Service log models are the set of models that cover preprocessing procedure. The included models are Invocation Log Model, Service Event Model, and Trace Model as the following definitions.

Definition 1. InvocationLogModel (ILM) represents the invocation records that devices executed as event requests. It is a list of service invocation records containing information of devices, users, services and the execution timestamp. The definition of ILM is as equation (1)–(4).

$$ILM \longleftarrow \{Device, Service, User, Timestamp\}, \quad (1)$$

$$Service \longleftarrow \{Description, RequestURL, Action\}, \quad (2)$$

$$Devices \longleftarrow \{DeviceID, DeveiceType, OSType\}, \quad (3)$$

$$User \longleftarrow \{UserID, UserName, RoleSet, UserProperties\}. \quad (4)$$

Definition 2. EventDictionaryModel (EDM) represents the dictionary of the mapping rules between events and the execution services, as shown in (5). The event is defined as in (6), and the service shares the same definition as that in (2):

$$EDM \longleftarrow \{Event, Service\}, \quad (5)$$

$$Event \longleftarrow \{EventName, EventCategory\}. \quad (6)$$

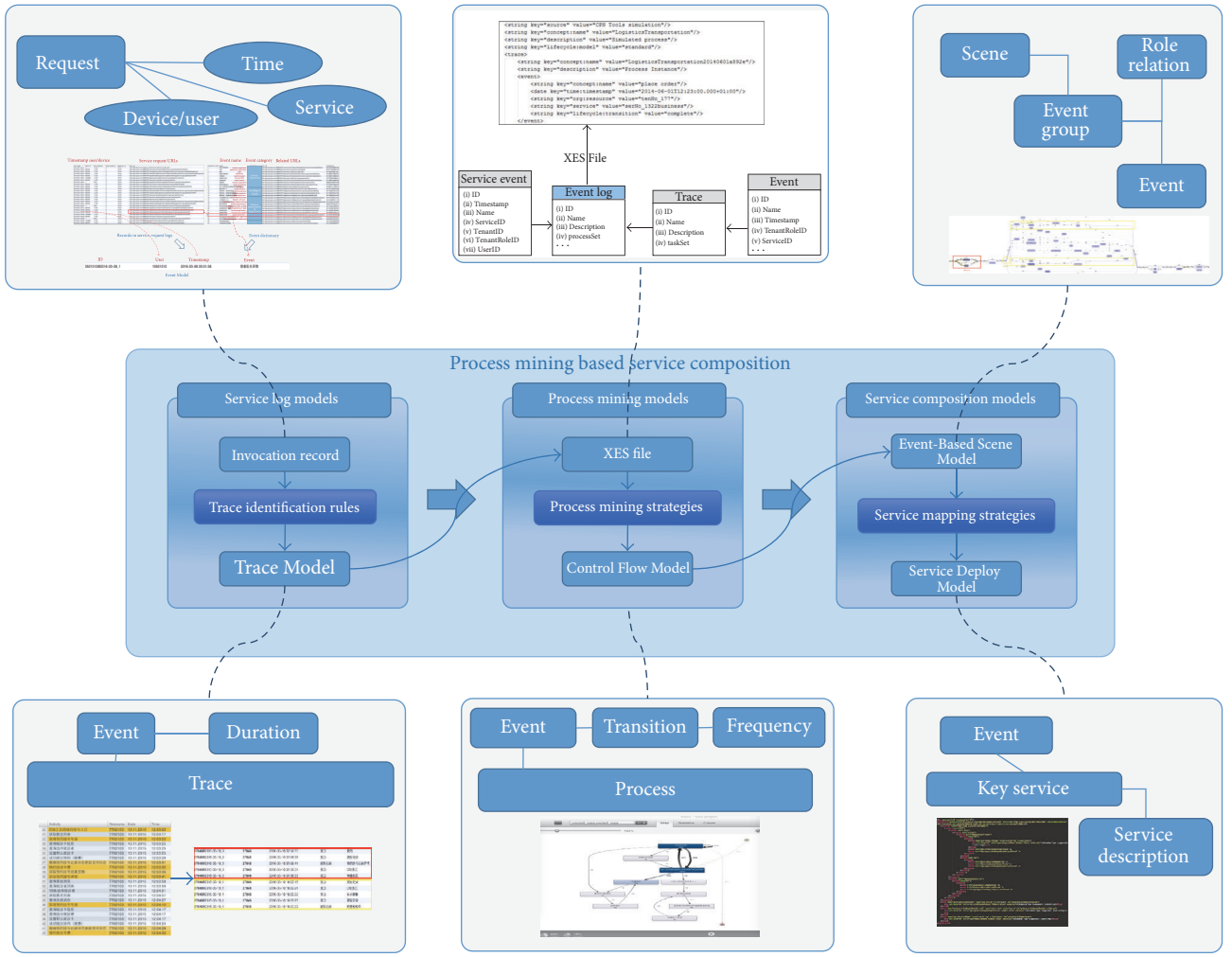


Figure 2: Process mining based service composition models.

Definition 3. EventModel (EM) keeps the operation information from users, including *User, Event,* and *Timestamps*, as in the following equation:

$$EM \longleftarrow \{User, Event, Timestamps\}. \quad (7)$$

Definition 4. TraceModel (TM) contains a group of traces that represent a sequence of continual operation events, including a set of event models and the time duration information, as in the following equation:

$$TM \longleftarrow \{List\{EM\}, User, StartTime, EndTime\}. \quad (8)$$

3.1.2. Process Mining Models. The process mining model restores information for process mining.

The Extensible Event Stream (XES) can be regarded as unification data form between trace models and standard process mining input. The input format of this phase is XES which is a process instance that has integrated multiple Service Events. It contains multiple processes, which are called trace in XES standards, and every trace is related to a trace model that contains multiple events.

Definition 5. ProcessModel (PM) is the output of process mining. Business process is defined as a process that contains events and the control flow between them which is presented as event and transition. And a set of frequency representing the execution frequency of each event is also included for further analysis, as in the following equation:

$$PM \longleftarrow \{Event, Transition, Frequency\}. \quad (9)$$

3.1.3. Service Composition Models. The service composition models restore information from process model, event-role relation, and event service relation. Process model is the process discovered through process mining. Event-role relation includes relations between service events and roles. And Key Service Model is the mapping between services and scene-based events.

Definition 6. EventServiceSceneModel (ESRM) represents the scene based on event analysis:

$$ESRM \longleftarrow \{SceneDescription, ServiceSet, EventSet, RoleSet, KSM\}. \quad (10)$$

Input:
IvoncationLogModel: *ILM*,
EventDictionaryModel: *EDM*,
ValidateUser
TimeDuration
Output:
TraceModelTM
(1) *FilteredLog* ← *ILM*
(2) *TM* ← ∅
(3) *FilteredLog*.remove r if not $User(r) \in ValidateUser$
(4) *FilteredLog*.remove r if not $Time(r) \in TimeDuration$
(5) *FilteredLog*.remove r if $r' == r$
(6) *FilteredLog.sortByTime*()
(7) *trace* ← {*FilteredLog.first*().*event*()}
(8) **while** *FilteredLog.hasNext*() **do**
(9) r ← *FilteredLog.next*().*event*()
(10) **if** r in a short time **then**
(11) trace.add(r)
(12) **else**
(13) TM.add(trace)
(14) *Trace* ← {r}
(15) **end if**
(16) **end while**
(17) **return** *TraceModel*

Algorithm 1: Preprocessing—preprocessing logs for process mining by steps of removing invalid records, eliminating similar request in a short time, picking the successful request and deleting others, connecting service with events, and dividing the events into traces, according to time duration.

Definition 7. KeyServiceModel (KSM) represents the mapping between events and most suited services:

$$KSM \longleftarrow Set\{Service, Event\}. \tag{11}$$

3.2. Execution Log Processing. The log data in IoE is getting more complex with increasing amount of connections, leading to larger scale of events and services. As a result the service logs are not suitable for process mining due to noises and unclear boundaries. Therefore, in the first phase of our method, we extract the execution data from service logs, remove the noise data, and generate traces in trace model.

The preprocessing algorithm is shown as Algorithm 1. Consider the record size of initial logs as data size n. The data cleaning part (line (1) to line (5)) takes a time complexity of $O(n)$, for we only have to travel the data once and remove dirty data by $O(1)$ determinations. And the sorting part (line (6)) is a classic sorting problem which can be optimized to finish in $O(n \log n)$. Finally, the connecting part (the while loop) takes the time complexity of $O(n)$. Because we go through the clean logs (less than n) again and the creating of trace is an $O(1)$ operation, the overall complexity of the algorithm is $O(n \log n)$. As we can see, the preprocessing procedure uses most time in sorting the event records. If the records are already sorted in the initial logs, this algorithm can have a time complexity of $O(n)$. As to space requirement, the cleaning part can be done in place. The sorting part and connecting part each take $O(n)$ space. Because the data size n can be controlled by separating logs by different time periods, this step can be done distributively in acceptable time. Therefore the preprocessing step will not take too much time regarding large scale of logs.

3.2.1. Preprocessing Noise Data. In preprocessing phases, first of all, service invocation logs are used as input of preprocessing step. The original logs keep recordings of service invocation information. Logs contain information for process execution and bridge the gap between service composition and service deployment. However, the logs cannot be used as input of process mining directly as a result of different viewpoints of data organization and different structures of data storage. Therefore, before doing process mining, it is necessary to remove the outdated and incorrect data in logs to extract the required information.

First of all, we manually decide valid users, valid time, and max transaction duration, which means to define *Set*{*ValidateUser*} and *Set*{*TimeDuration*}. Then we remove the invalid records according to the valid configuration. After that, we eliminate the duplicate records that are produced due to connection errors in network.

3.2.2. Generating Event Model. The next step is to transform the records into the event models with the assistance of event dictionary. As mentioned above, the original service invocation logs are restored in the form of *ILM*. And the process mining are based on event data like *EM*. So we transform the ILM into EM by mapping the attribute of *ILM.service.URL* and *EDM.service.URL*, which is presented as *event*() in the algorithm.

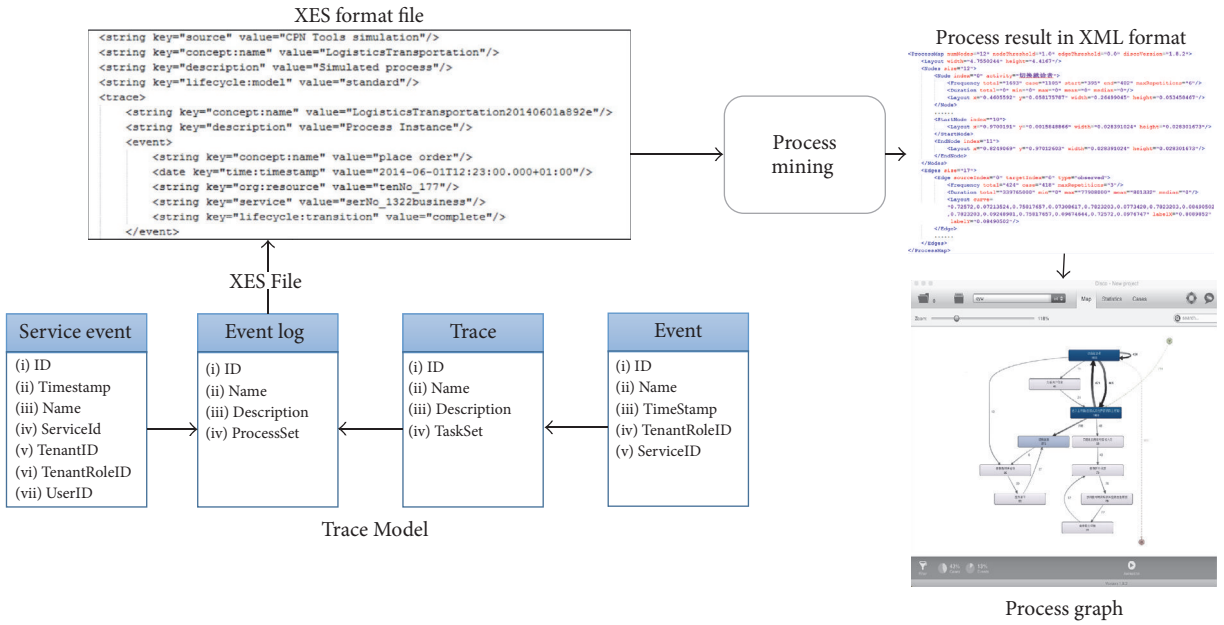


Figure 3: Execution of process mining.

3.2.3. Generating Trace Model. The last step of preprocessing is to reorganize the event models into trace models. Other than the Iterative Expectation-Maximization Procedure method introduced in [13], which takes too much time when confronting large amount of logs, we use the dividing strategy based on time duration separation. First, we group the event models by the attribute of user. That is, for each user, we have a group of (event, timestamps) pairs. By sorting the events on time, the group of events contains sequences of events. Then we separate them into different traces according to the time duration.

3.3. Process Mining. Process mining is a technique that extracts information from event logs recorded by an information system to discover, analyze, and enhance process models. As in Figure 3, the event logs are from the executing network of devices.

3.3.1. Transforming Trace Model to XES. Processing event logs is to convert the information for process mining we got from log processing into the input criterion required by the process mining tool (like ProM [14] and Disco [15]), which requires XES (Extensible Event Stream) as input format. XES file is a process instance that has integrated multiple service events. It contains multiple processes, which are called trace in XES standards, and every trace contains multiple events, as in the left part of Figure 3.

3.3.2. Executing Process Mining. In the part of process mining, the fuzzy mining algorithm [16] is selected. In the case of our implementation, we choose the fuzzy miner module of tool Disco. The miner is based on the significance and correlation of events to produce adaptable process models, as in the right part of Figure 3.

3.4. Scenario-Based Service Composition. After the steps mentioned above, the process model is produced from device-to-service invocation log. The next step is to adjust the process by execution frequency of events and relocate the services to the process. We provide the procedure as Algorithm 2.

Consider the total event size as data size n. Removing less important nodes (line (1) to (5)) takes $O(n)$, because we only have to calculate the result of $\sum_{e_i \in E} F(e_i)$ once. And in the event grouping and scene generalization part (line (7) to line (16)), calculating all the $\text{sim}(e_i, e_j)$ takes $O(n^2)$. And add/remove operation can be done in $O(1)$. Since the while loop iterates at most n times, the worst complexity of the algorithm is $O(n^3)$. As we can see, the most time taken is in generating Composition Model. The iteration time is dependent on specific data. Comparing to other composition approaches, the scenario generation takes extra time to simplify the processes. Since the event size will not be very large in systems, the time consumed is considered acceptable.

3.4.1. Scene-Based Event Analysis. As a process mining result, a mined process is presented as a directed graph with nodes and edges. By analyzing the source and target in process model, we could get the sequence of events in a process graph. In the graph, nodes represent events and edges indicate the transitions of events. Each edge has a weight representing the frequency of transitions.

To simplify the graph, insignificant nodes and edges will be removed. Frequency of an event e is noted as $F(e)$. Then the importance of the event $I(e)$ is defined as

$$I(e) = \frac{F(e)}{\sum_{e_i \in E} F(e_i)}. \tag{12}$$

Input:
$PM = \{V, E, F\}$
Output:
CompositionModel
(1) **for all** $e_i \in E$ **do**
(2) **if** $I(e) < IO(e)$ **then**
(3) $E.remove(e)$
(4) **end if**
(5) **end for**
(6) Composition Model = PM
(7) **while** Last iteration change Composition Model **do**
(8) **for all** $e_i \in E$ **do**
(9) **for all** $e_j \in e_i.to$ **do**
(10) **if** $\text{sim}(e_i, e_j) < threhold$ **then**
(11) $CompositionModel.remove(e_i, e_j)$
(12) $CompositionModel.add(Scene(e_i, e_j))$
(13) **end if**
(14) **end for**
(15) **end for**
(16) **end while**
(17) **return** Composition Model

Algorithm 2: Scene-based service composition.

Thus $I(e)$ is the ratio of its frequency $F(e)$ and the sum of all the event frequencies. The events with much low frequency can be removed from the graph.

And for the edges, we note sum of all the input transition frequencies as $ID(e)$ and sum of all the output transition frequencies as $OD(e)$:

$$IO(e) = \frac{ID(e)}{OD(e)}. \tag{13}$$

The smallest $IO(e)$ is the start node of the process, and the largest is the end node.

For a transition t, and its source event $e = t.sourceEvent$, the importance of the transition $I(t)$ is shown as follows:

$$I(t) = \frac{F(t)}{F(e)}. \tag{14}$$

If $I(t)$ is much lower than normal, the transition hardly happens according to existing logs. So it can be removed:

$$\text{sim}\left(e_i, e_j\right) = \frac{F\left(t_{ij}\right)}{DO\left(e_i\right) + IO\left(e_j\right)}. \tag{15}$$

For the nodes with similarity close to 1, they are normally executed as a patterned sequence. In other words, e_i, e_j are usually executed at the similar situations. We can group (e_i, e_j) as a scene. And this procedure is repeated literately.

3.4.2. Determine Key Services. In this part, services are marked with priorities in order to pick the most suitable service for each event. In the service repository, similar services are existing. However, these services have different influence in a particular process environment. It is necessary to pick out the most suitable services.

After process mining, two factors can be introduced in service selection: relevance of service-to-event and relevance of service-to-scene. For each event, each service has a priority. The same event may not invoke the fixed service every time, and one service may also be provided to multiple events, so we need a method to choose suitable services, that is, the strategy we use to extract Key Service from all the invoked services (in service repository). We calculate the weight of the service for the event to measure its criticality in service mapping. $F(e, s)$ represents the number of execution time from service e. The outdated data is filtered, so $F(e, s)$ can be used to calculate the importance of service s to event e:

$$\begin{aligned}\text{priority}\left(e_i, s_i\right) = u \cdot \frac{F\left(e_i, s_i\right)}{F\left(e_i\right)} + v \\ \cdot \frac{\sum_{e \in \text{scene}(e_i)} F\left(e, s_j\right)}{\sum_{e \in \text{scene}(e_i)} F(e)}.\end{aligned} \tag{16}$$

With the priority, each event can be related to most usually used services, which means $KSM \leftarrow Set\{Service, Event\}$ can be generated. And the combination of Composition Model and KSM Model becomes the Service Deployment Model.

4. Evaluation

4.1. Case Study: An Application in Mobile Medical System. In this section a case study will be presented to demonstrate the approach.

One of the most potential usages of connecting everything is the application of IoE in medical processes.

For case study, a mobile medical system with large numbers of smart devices (mostly smart phones) in China

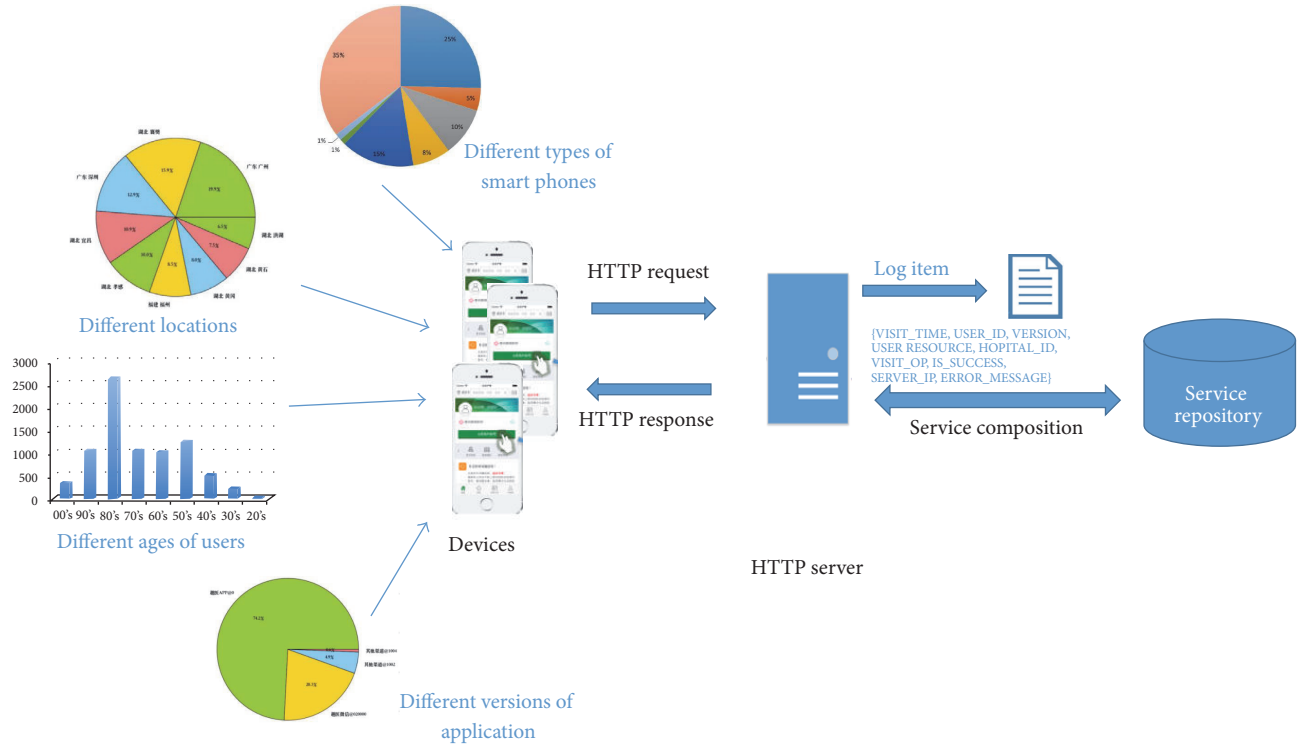


Figure 4: Mobile medical system in IoE.

is used in this evaluation (as in Figure 4). In particular, a registration process is demonstrated in the following part.

As the mobile medical system is getting popular, it is widely used in many provinces over the whole country. The connection network of people, devices, and medical organizations is getting larger recently. With larger scale of usage, the system faces difficulties in optimization of services. The devices have different operation systems and application versions. Due to the variability of operation systems, application versions, and geological locations, the behavior of usage cannot be unified. Unpredictable service usage leads to difficulty in optimization of services. It is inconvenient for updating both mobile applications and server-side systems.

4.2. Preprocessing the Logs. For the case study, five months of logs from the http server of the system is used. The selected logs are from May 2015 to April 2016. Each record includes *visit_time*, *user_id*, *app_version*, *hospital_id* and *visit_op*. The initial log is shown in the left part of Figure 5.

In this log, each record represents a service request. Typical noise of the data includes duplicate operations, invalid operations, and unclear transaction boundary. First, data cleaning is applied to the initial logs. Then, we execute *visit_optobusiness_codemapping*. And the structure of event dictionary is shown in the right part of Figure 5. After mapping service request URL with events, each record is transformed into event model as the bottom part in Figure 5.

To identify traces, the following rules are applied: to ensure over 75% traces are correctly identified, operations that take less than 30 min and 36 seconds are regarded as the same trace. And the result of Trace Model is shown in Figure 6.

4.3. Process Mining. In the process mining phase, the first step is to transfer Trace Model into standard process mining input, that is, to generate XES file with the above method. In the case study, the log is transferred into the log. After preprocessing, we transfer the trace models into XES format, as in Figure 7(a). Disco is chosen to be our process mining platform where the XES can be used directly as standard process mining input. After selecting filters (as Figure 7(b)), we choose the fuzzy miner as the process mining strategy. The tool is used to analyze the interaction records among the business activities in the processes and through mining and reasoning to get the process model. After process discovery, the process model (as in Figure 7(c)) is stored in the form of the XML file (as in Figure 7(d)).

4.4. Scene-Based Composition. Then we combine the service set with the event set. The service is combined with the event according to the corresponding event ID. The similar phase is done to the role set as well. Figure 8 shows the optimization of control flow in this case, which includes start node identification, similar event composition, and less significant event reduction.

Through service selection, a set of key services will be generated. After we import the data of process mining phase to service composition phase, the Service Deployment Model can be generated. And with template technologies, we can generate the service descriptions for service compositions

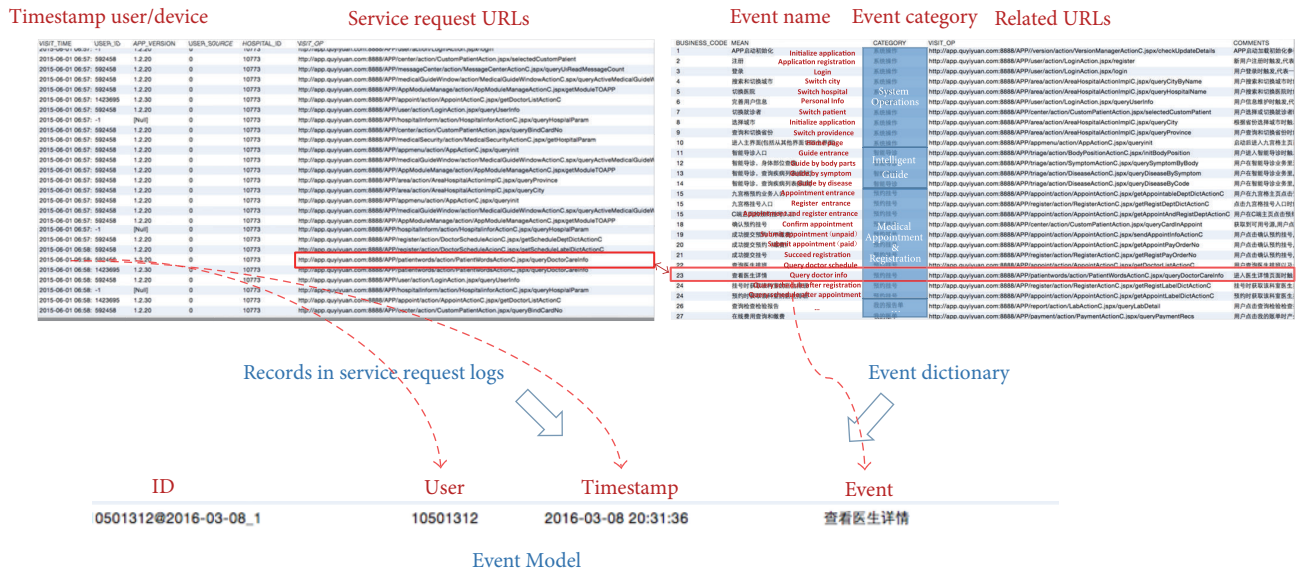


Figure 5: Mapping service request with event dictionary.

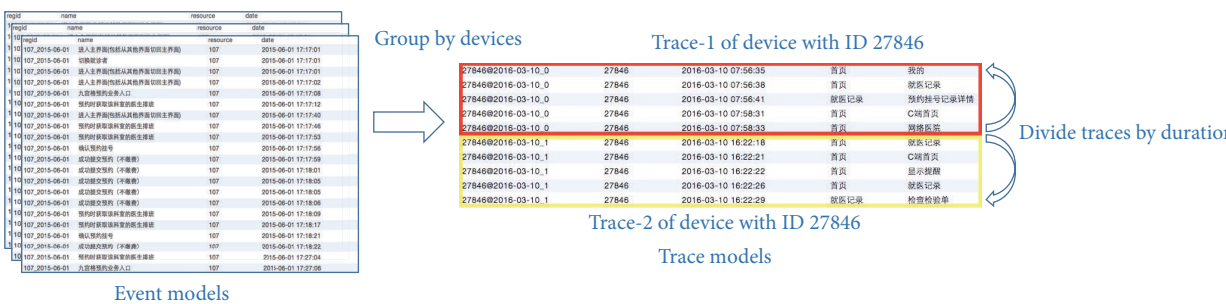


Figure 6: Separating traces from logs.

of scenes. Figure 9 shows examples of result of key services mapping and service generation. In Figure 10(a), the key services are mapped to the events ($u = 0.9, v = 0.1$ for priority calculation). And Figure 10(b) shows one of the examples of generated WSDL descriptions for composited service.

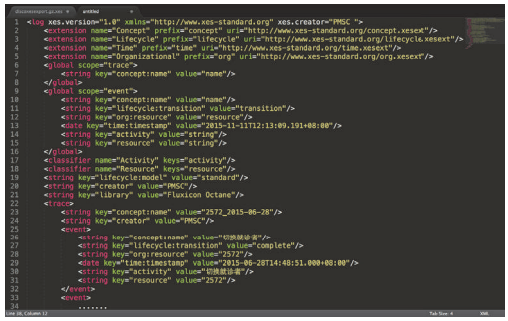
Then the composite service is registered in the service library and enters the service deployment phase. After long-term running, the execution of this service will leave behind service logs which can be used for the new process mining phase of the next generation.

4.5. Result and Discussion. After applying our work to the mobile medical system, the registration process of the system is improved considering two criteria.

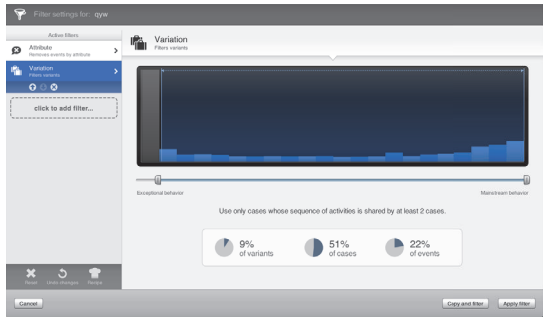
First of all, the simplicity of the new process is improved after we composite the services that invoked as a pattern. Secondly, as services are composited for certain scene, the rules defined in devices can be simplified. And with the discovery of composition, further optimization can be implemented to redeploy the services so that services in the same scene can be physically deployed in the same server to reach a better performance.

We recollect the execution logs after adjustment of event rules to the new service compositions. To evaluate the performance, we compare two log data, one from the week right before redeploying the service composition and the other from the week right after applying our method (see Table 1 and corresponding Figure 10). It is assumed that, in the continuous two weeks, the user behavior and the operation of the application should not change much. As we can see in the result, after reduplicate request and meaningless events are removed, the total amount of the events is reduced owing to the simplification of the process. To complete the same functional requirement, the events of each case are greatly reduced. And the relative percentage of event that may be caused by users' hesitation like "Select City" and "Switch Province" is reduced. Thus the execution of the process is improved by efficiency.

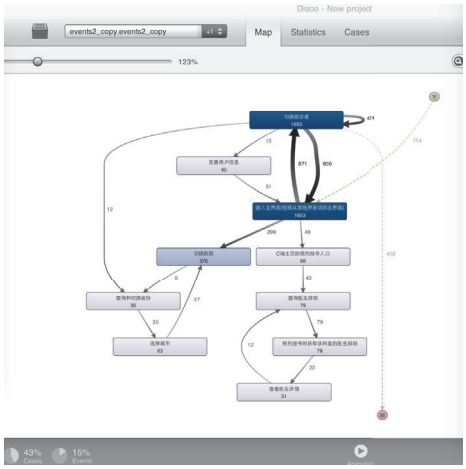
As to privacy issues, first of all, the input of our approach is system log that contains service requests. They do not contain sensitive data such as credit accounts. Our method just uses the necessary data that is usually used for system maintaining. And after process mining, the mining result is a summary of all the behaviors rather than an operation



(a) Snippet of generated XES file



(b) Configuration fuzzy mining



(c) Output of process mining

(d) Output in XML format

Figure 7: Execution of process mining.

Table 1: Comparison of event frequencies before and after our optimization.

Event	Original frequency	Original frequency rate	Improved frequency	Improved frequency rate
Main Entrance	7,777	35.21%	1,853	42.91%
Switch Patient	4,473	20.25%	1,693	39.21%
Check Doctor Schedule	1,825	8.26%	79	1.83%
Query Doctor Info	1,713	7.76%	31	0.72%
Switch Hospital	1,474	6.67%	376	8.71%
Select City	1,114	5.04%	63	1.46%
Appointment Entrance	725	3.28%	69	1.6%
User Profile	621	2.81%	45	1.04%
Switch Province	540	2.44%	30	0.69%

sequence of individual person. So our service composition is based on the summarized result of a group.

5. Related Work

The existing approaches that perform service discovery and service composition will be discussed in this section.

For service selection solutions, in [9], a service selection technique is proposed to select the best potential candidate service from a set of functionally equivalent ones. The approach in [17] takes several aspects such as QoS, user preference, and the service relationship into consideration. And the work [18] proposes an effective approach to extract events and their internal links from large-scale data with predefined event schema.

As to context-aware dynamic service composition approach and AI planning techniques in addition, [10, 19, 20] use models at runtime to guide the dynamic evolution of context-aware web service compositions to cope with unexpected situations. Reference [21] proposes a service granularity space for multitenant service composition, which provides a semantic basis for multitenant service composition. In [22], a methodology based on process mining is proposed to do business process analysis in health care environments to

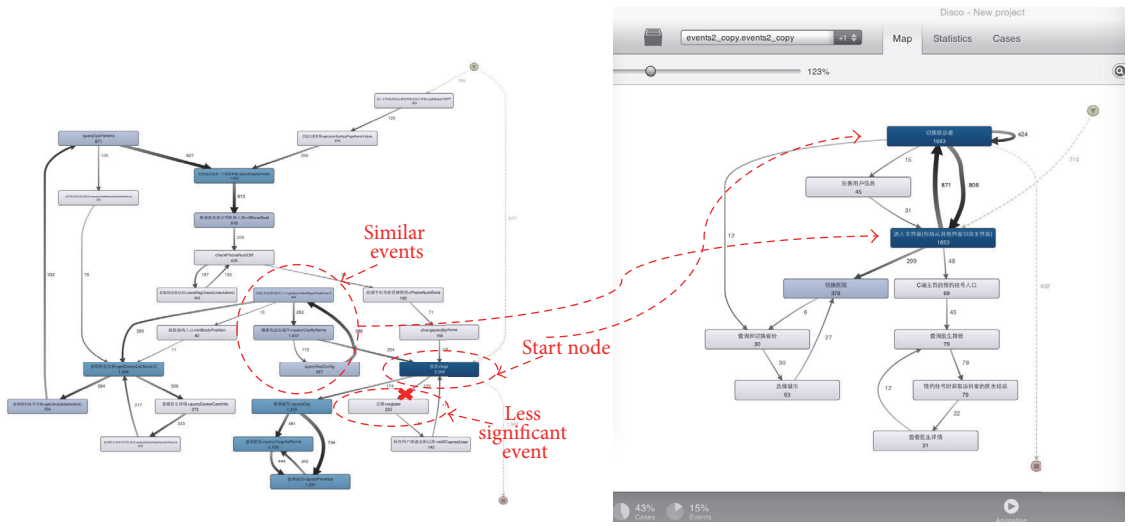
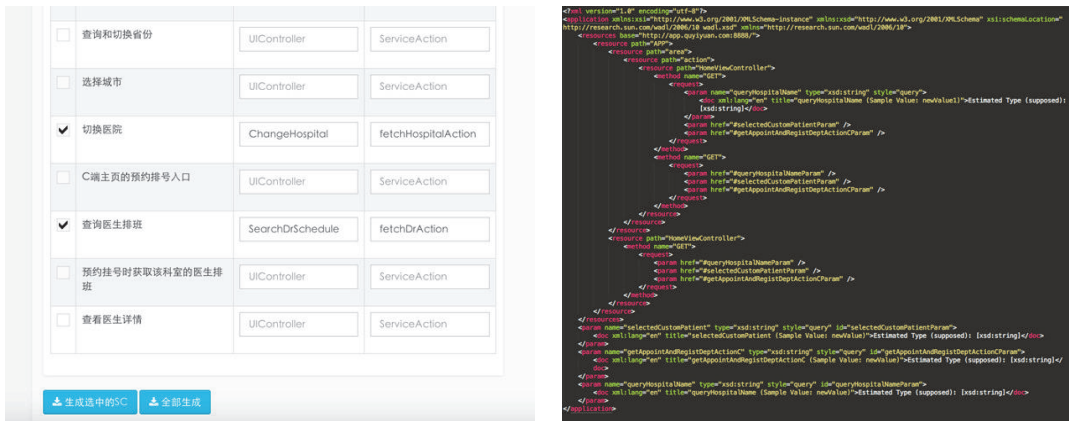


Figure 8: Scene-based control flow optimization.



(a) Service mapping

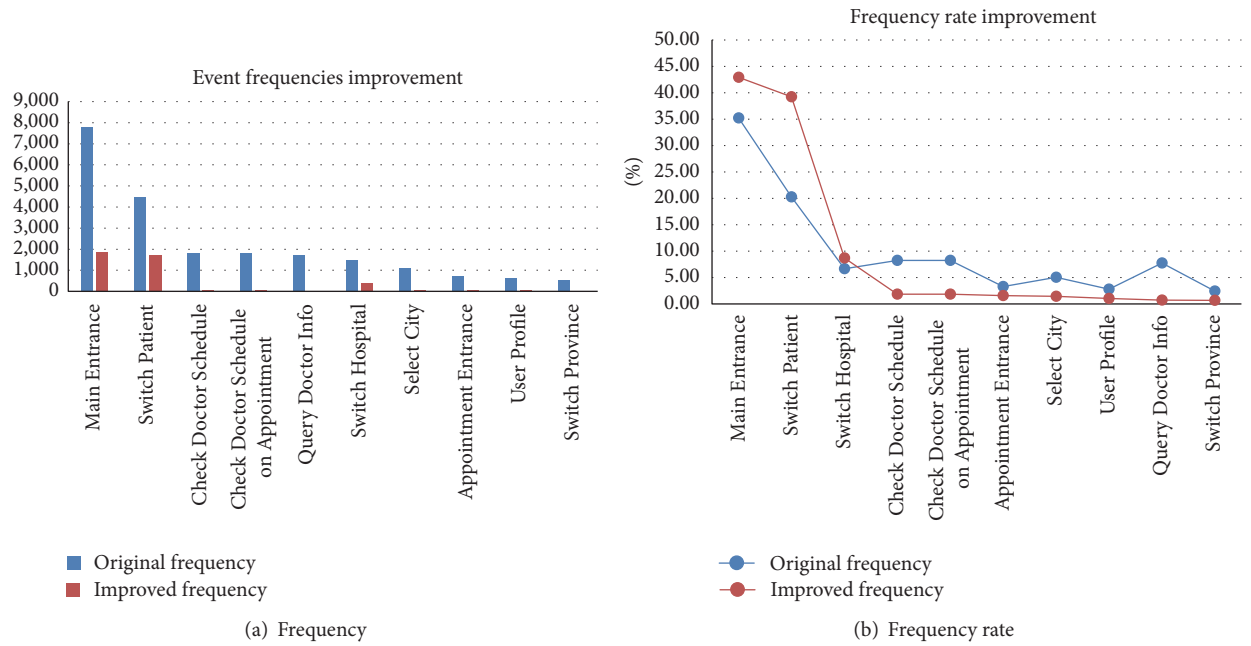
(b) Service generation

Figure 9: Mapping and deploy services.

identify regular behavior, process variants, and exceptional medical cases.

For optimizing the existing service approaches, there are few approaches about service composition in the area of service mining, such as service composition analysis and optimization. The following works are devoted to optimizing the existing service composition based on mining patterns from existing data. A mining algorithm based on statistical techniques to discover composite web service patterns from execution logs is proposed by [23] to better understand, control, and eventually redesign the composite services while [24] proposed an approach to generate service composition pattern for cloud migration from a set of service composition solutions by a graph similarity analysis approach. In [25], an event-based monitoring approach for service composition infrastructures is presented to provide a holistic monitoring approach by leveraging Complex Event Processing techniques. In summary, the works [23–25] use data mining instead of process mining.

Our work proposed a service composition approach based on process mining, which is aimed at improving the adaptiveness and efficiency of compositions considering the expanding scale and the variety of devices in mobile information systems. In terms of the main objectives of these three approaches, our service composition approach is based on process mining and can select services according to the result of the process mining while the other approaches either focus on performance or on context environment. We compare our work with the recent service composition approaches in service composition research area, that is, QoS-based service composition approach [9] and context-aware dynamic service composition approach [10] in Table 2. Although it is hard to execute the data with existing approaches, our approach is more suitable in some cases. Our approach



(a) Frequency

(b) Frequency rate

Figure 10: Comparison of event frequency before and after our optimization.

Table 2: Comparison to existing approaches.

Features	Our approach	QoS-based [9]	Context-aware [10]
Main objective	Improve processes efficiency	Improve performance	Improve adaptabilities
Service selection criteria	Execution info from process mining	QoS goals from SaaS providers	Context collected from equipment
Requirement accuracy	High	Low	High
Nonfunctionality	Medium	High	Low
Adaptability accuracy	High	Low	High
Performance accuracy	Improved	Improved	Medium
Nonfunctionality	Medium	High	Low
Time cost	Lower	Lower	High
Flexibility	Medium	Medium	High
Optimization support	Yes	No	Yes

has advantages that other approaches do not have. Firstly, our work can handle the comprehensiveness from business rules. Rather than focusing on execution time selection as in work [9], service invoking pattern discovery is also considered in our work. As a result, service execution relation can be optimized rather than optimized single request time. Secondly, rather than taking information from equipment context in [10], our method is based on server-side data. Though our offline computing is not as flexible as dynamic perdition, our method can handle a system that different versions of devices rule execute together.

In conclusion, our service composition approach based on process mining is outstanding in comprehensiveness with acceptable time cost and flexibility. However there is currently no standard benchmark to evaluate the performance of each work, due to the different focus of area. It can be concluded that our method can improve both the adaptiveness of functional requirement and the efficiency of process executing. So, it is more suitable than other approaches when there are different types for devices that use services in a different way.

6. Conclusions

In the area of the Internet of Everything, service composition is widely used for the development of applications. In this paper, in order to improve both execution effectiveness and comprehensiveness of existing service compositions, we propose a service composition approach based on process mining, considering both the practical business and the execution information in environment with large amount of connection between devices and users. It is shown that our approach can improve the adaptiveness of process by combining the execution information with service composition. And the efficiency of compositions can be further optimized by redeploying the services in the same scene on the same physical server, which is planned as our further work.

Competing Interests

The authors declare that there are no competing interests.

Acknowledgments

The authors would like to acknowledge the support provided by the National Natural Science Foundation of China under nos. 71171132 and 61373030.

References

[1] H. Hoehle and V. Venkatesh, "Mobile application usability: conceptualization and instrument development," *MIS Quarterly*, vol. 39, no. 2, pp. 435–472, 2015.

[2] F. F. Ntawanga, A. P. Calitz, and L. Barnard, "A context-aware model to improve usability of information display on smartphone apps for emerging users," *The African Journal of Information Systems*, vol. 7, no. 4, p. 3, 2015.

[3] V. V. S. M. Chintapalli, W. Tao, Z. Meng, K. Zhang, J. Kong, and Y. Ge, "A comparative study of spreadsheet applications on mobile devices," *Mobile Information Systems*, vol. 2016, Article ID 9816152, 10 pages, 2016.

[4] H. Cai, C. Xie, L. Jiang, L. Fang, and C. Huang, "An ontology-based semantic configuration approach to constructing Data as a Service for enterprises," *Enterprise Information Systems*, vol. 10, no. 3, pp. 325–348, 2016.

[5] B. Xu, L. D. Xu, H. Cai, C. Xie, J. Hu, and F. Bu, "Ubiquitous data accessing method in iot-based information system for emergency medical services," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1578–1586, 2014.

[6] C.-W. Shen, C.-H. Hsu, C.-C. Chou, and T.-C. Tsai, "Toward a nationwide mobile-based public healthcare service system with wireless sensor networks," *Mobile Information Systems*, vol. 2016, Article ID 1287507, 11 pages, 2016.

[7] H.-Y. Noh, J.-H. Lee, S.-W. Oh, K.-S. Hwang, and S.-B. Cho, "Exploiting indoor location and mobile information for context-awareness service," *Information Processing & Management*, vol. 48, no. 1, pp. 1–12, 2012.

[8] Y. Li, H. Cai, C. Huang, and F. Bu, "Leveraging process mining on service events towards service composition," in *Advances in Services Computing—9th Asia-Pacific Services Computing Conference (APSCC '15)*, pp. 195–209, 2015.

[9] T. Ahmed and A. Srivastava, "Minimizing waiting time for service composition: a frictional approach," in *Proceedings of the IEEE 20th International Conference on Web Services (ICWS '13)*, pp. 268–275, IEEE, Santa Clara, Calif, USA, July 2013.

[10] G. H. Alférez and V. Pelechano, "Facing uncertainty in web service compositions," in *Proceedings of the IEEE 20th International Conference on Web Services (ICWS '13)*, pp. 219–226, IEEE, Santa Clara, Calif, USA, July 2013.

[11] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros et al., "Process mining manifesto," in *Proceedings of the International Conference on Business Process Management*, pp. 169–194, Springer, 2011.

[12] W. V. D. Aalst, "Service mining: using process mining to discover, check, and improve service behavior," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 525–535, 2013.

[13] D. R. Ferreira and D. Gillblad, "Discovering process models from unlabeled event logs," in *Proceedings of the International Conference on Business Process Management*, pp. 143–158, Springer, Ulm, Germany, September 2009.

[14] W. M. P. Van Der Aalst, B. F. Van Dongen, C. W. Günther et al., "Process mining with ProM," in *Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence (BNAIC '07)*, pp. 453–454, Utrecht, The Netherlands, November 2007.

[15] C. W. Günther and A. Rozinat, "Disco: discover your processes," *BPM (Demos)*, vol. 940, pp. 40–44, 2012.

[16] C. W. Gunther and M. P. Wil Van Der Aalst, "Fuzzy mining-adaptive process simplification based on multi-perspective metrics," in *Business Process Management: 5th International Conference, BPM 2007, Brisbane, Australia, September 24–28, 2007. Proceedings*, vol. 4714 of *Lecture Notes in Computer Science*, pp. 328–343, Springer, Berlin, Germany, 2007.

[17] L. Cui, J. Li, and Y. Zheng, "A dynamic web service composition method based on viterbi algorithm," in *Proceedings of the IEEE 19th International Conference on Web Services (ICWS '12)*, pp. 267–271, Honolulu, Hawaii, USA, June 2012.

[18] Y. Sun, H. Yan, C. Lu, R. Bie, and Z. Zhou, "Constructing the Web of Events from raw data in the Web of Things," *Mobile Information Systems*, vol. 10, no. 1, pp. 105–125, 2014.

[19] B. Heinrich and L. Lewerenz, "Decision support for the usage of mobile information services: a context-aware service selection approach that considers the effects of context interdependencies," *Journal of Decision Systems*, vol. 24, no. 4, pp. 406–432, 2015.

[20] S. Wang, L. Sun, Q. Sun, X. Li, and F. Yang, "Efficient service selection in mobile information systems," *Mobile Information Systems*, vol. 2015, Article ID 949436, 10 pages, 2015.

[21] H. Cai, L. Cui, Y. Shi, L. Kong, and Z. Yan, "Multi-tenant service composition based on granularity computing," in *Proceedings of the 11th IEEE International Conference on Services Computing (SCC '14)*, pp. 669–676, Anchorage, Alaska, USA, July 2014.

[22] Á. Rebuge and D. R. Ferreira, "Business process analysis in healthcare environments: a methodology based on process mining," *Information Systems*, vol. 37, no. 2, pp. 99–116, 2012.

[23] W. Gaaloul, K. Baïna, and C. Godart, "Log-based mining techniques applied to web service composition reengineering," *Service Oriented Computing and Applications*, vol. 2, no. 2-3, pp. 93–110, 2008.

[24] Z. Wan, F. J. Meng, J. M. Xu, and P. Wang, "Service composition pattern generation for cloud migration: a graph similarity analysis approach," in *Proceedings of the 21st IEEE International Conference on Web Services (ICWS '14)*, pp. 321–328, Anchorage, Alaska, USA, July 2014.

[25] O. Moser, F. Rosenberg, and S. Dustdar, "Event driven monitoring for service composition infrastructures," in *Proceedings of the International Conference on Web Information Systems Engineering*, pp. 38–51, Springer, 2010.