International Journal of Automation and Computing 7(4), November 2010, 584-595
DOI: 10.1007/s11633-010-0544-2

A Service Composition Approach Based on Sequence Mining for Migrating E-learning Legacy System to SOA

Zhuo Zhang[1, 2] Dong-Dai Zhou[1, 3, 4] Hong-Ji Yang[2] Shao-Chun Zhong[1, 3, 4]

[1]Software School in Northeast Normal University, Changchun 130024, PRC

[2]Software Technology Research Laboratory, De Montfort University, Leicester LE1 9BH, UK

[3]Institute of Ideal Information and Technology in Northeast Normal University, Changchun 130024, PRC

[4]Engineering Research Center of E-Learning Technologies, Ministry of Education, Changchun 130024, PRC

Abstract: With the fast development of business logic and information technology, today′s best solutions are tomorrow′s legacy systems. In China, the situation in the education domain follows the same path. Currently, there exists a number of e-learning legacy assets with accumulated practical business experience, such as program resource, usage behaviour data resource, and so on. In order to use these legacy assets adequately and efficiently, we should not only utilize the explicit assets but also discover the hidden assets. The usage behaviour data resource is the set of practical operation sequences requested by all users. The hidden patterns in this data resource will provide users′ practical experiences, which can benefit the service composition in service-oriented architecture (SOA) migration. Namely, these discovered patterns will be the candidate composite services (coarse-grained) in SOA systems. Although data mining techniques have been used for software engineering tasks, little is known about how they can be used for service composition of migrating an e-learning legacy system (MELS) to SOA. In this paper, we propose a service composition approach based on sequence mining techniques for MELS. Composite services found by this approach will be the complementation of business logic analysis results of MELS. The core of this approach is to develop an appropriate sequence mining algorithm for mining related data collected from an e-learning legacy system. According to the features of execution trace data on usage behaviour from this e-learning legacy system and needs of further pattern analysis, we propose a sequential mining algorithm to mine this kind of data of the legacy system. For validation, this approach has been applied to the corresponding real data, which was collected from the e-learning legacy system; meanwhile, some investigation questionnaires were set up to collect satisfaction data. The investigation result is 90 % the same with the result obtained through our approach.

Keywords: Service composition, e-learning, sequence mining algorithm, service-oriented architecture (SOA), legacy system.

1 Introduction

With decades of technological advances and the change of business climates and objectives, today′s state-of-the-art business solutions are tomorrow′s legacy systems. Currently, there exists a number of large legacy systems that have become difficult and expensive to change and cannot sustain the demands of the marketing department for alterations, leading to business opportunities being lost[1]. However, many legacy systems contain accumulated business experience and useful business processes and they are worth reusing in whole or in part. In this case, it is not a good choice to develop a new system from scratch. Rather, a software product is usually developed through extending an existing system by describing new business logic that manipulates an existing repository of data, presenting existing data and transactions through new channels such as Internet browsers or handheld devices, integrating previously disconnected systems supporting overlapping business activities, and so on[2]. Thus, software reengineering gets more attention, with a question: what are the best approaches for building high quality systems that cannot only deploy efficiently the valued legacy components but also be beneficial to application and further evolution?

One way to change a legacy system is to consider it to be composed of a collection of interacting services. The concept of service was born to address reuse issues. Each service provides access to a well-defined set of functionalities. The business requirements can be implemented with a view to services, and the major elements of business processes are services. The business process is designed and implemented as a set of interactions among these services. Packaging functions as services is the key to flexibility. While the services encapsulate the business functionality, the inter-service infrastructure is required to facilitate service interactions and communication. Services can be implemented on a single computer, distributed on a local area network, or distributed on the Internet. When the services use the Internet as the communication mechanism, it allows others to make use of their services in a natural way regardless of their physical location. It provides a convenient way to cross hardware, language, and network boundaries. A system evolves through adding new services. This is so-called service oriented architecture (SOA), which is a business-centric information technology (IT) architectural approach that supports integrating business as linked, repeatable business tasks, or services. The key to the success of SOA is the reuse of existing IT assets such as legacy applications. Obviously, legacy system understanding and reuse is important for SOA. Legacy system understanding approaches include static trace analysis and dynamic trace analysis. Static traces are used to test the program modification in SOA migration. Dynamic traces are used to

Manuscript received October 30, 2008; revised November 4, 2009
The work was supported by E-learning Platform, National Torch Project (No. z20040010)

determine service composition and migration decision strategy. Both of them are used to extract the service interface description.

In order to understand well the legacy applications, some data mining techniques have been applied to software engineering (SE) data. The goals of mining SE data are to transform static record-keeping SE data to active data and make SE data actionable by uncovering hidden patterns and trends. Concretely, mining SE data can gain empirically-based understanding of software development; predict, plan, and understand various aspects of a project; support future development and project management activities[3].

The situation in the education domain is like this in China. Education reform is the first important thing and some new issues and demands on education policies and regulations are being addressed rapidly. Because e-education is a major application field of IT, currently, there exist a number of e-learning legacy systems. Most e-learning systems are not structured in a perfect architecture, and their elements are tightly coupled. Thus, there are a lot of problems such as redundant development, difficulties to integrate with others and maintenance difficulties among education software products.

However, there exist huge amounts of assets with accumulated business practical experience in an e-learning legacy system, such as program resource, usage behaviour data resource, and so on. Thus, they are worth reusing in whole or in part. In order to reuse these legacy assets adequately and efficiently, we should not only utilise the explicit assets but also discover the hidden assets. The usage behaviour data resource is the set of practical operation sequences requested by all users. The hidden patterns in this data resource will provide users′ practical experiences, which can benefit the service composition in MELS. Namely, these discovered patterns will be the candidate composite services (coarse-grained) in the SOA system. Although data mining techniques have been used for software engineering tasks, little is known about how they can be used for service composition in migrating an e-learning legacy system (MELS). MELS has become a main task of some education software development institutes and companies.

In this paper, we propose a service composition approach based on sequence mining techniques for MELS. The core of this approach is to develop an appropriate sequence mining algorithm for mining related data collected from an e-learning legacy system. According to the features of execution trace data on usage behaviour and needs of further pattern analysis, we propose a sequential mining algorithm to mine this kind of legacy data.

This paper is organised as follows: basic knowledge and approaches on service, service migration, service composition, and sequence mining techniques will be introduced in Section 2; in Section 3, a service composition approach based on sequence mining techniques for MELS will be described; in Section 4, we will represent a sequence mining algorithm for finding all maximal and frequent patterns on any min-support numbers. This algorithm can be used to determine service composition in MELS through mining execution traces data of an e-learning legacy system; the details of case study by applying this approach are described in Section 5; and the final section is conclusion.

2 Service composition and sequence mining techniques

2.1 Services and SOA

SOA is not a novelty. A service-oriented architecture is a framework model, which is composed of a set of services. Services are the core of SOA.

From the implementation point of view, a service is a self-contained and logical entity, which includes well-defined interfaces and contracts between services. The services communicate with each other for the data transfer or for coordinating some business process. Web technologies provide a rich set of tools for delivering services. Services are commonly delivered as wrappers for existing applications, and then these services map into well-defined business processes.

From the application point of view, service creation is based on what the business needs but not based on IT capabilities. Business requirements provide service routing information. Services provide a better way to expose discrete business functions. Therefore, services orientation provides an excellent way to build enterprise-scale solutions that support business processes.

2.2 SOA migration

SOA migration tasks can be considered from three viewpoints: service customer, SOA architect, and service provider[4]. Service provider viewpoint is the most common and based on this, roughly, the existing strategies on integrating legacy system to SOA[5] can be divided into the following three categories:

1) Black-box approach

This kind of approach tends to wrap legacy systems to web service. First, it analyses the legacy interfaces, and then, compares them with the new ones required in SOA. The matches will be wrapped into services by service-oriented technology. In this case, services will be composed of the legacy system internals and new web service interfaces[6]. The system built by this kind of approach will be hard to evolve, as it can satisfy a short term need and can actually complicate legacy system maintenance and management over a long term.

2) Business logic approach

This kind of approach reengineer legacy systems to web services by means of business logic[7]. First, the business logic is recovered from a legacy system by reverse engineering techniques, and then, a new web service system is developed in terms of this business logic. This kind of approach can save on maintenance and increase process efficiency. However, one of its cost is entire system comprehension, and sometimes, it is difficult to recover the business logic completely.

3) Grey-box approach

This kind of approach is a combination of black-box and business logic approaches. In this approach, parts of a legacy system are integrated with valuable business logic. Meanwhile, some legacy functionality is extracted as components. Component orientation supports the realisation

of service oriented computing. Web services and SOA technologies make them reusable and the final coordination services benefit more from a legacy system.

At present, the more popular strategy would be the third one, i.e., grey box approach. The combination of functional business processes and existing IT assets provides a potent solution to migrate legacy systems to a modern SOA. How to balance the role of a legacy system and functional business process is the difficulty of the third strategy. The key to the success of SOA is the reuse of existing IT assets such as legacy applications. Obviously, legacy system understanding and reuse are important in SOA.

Currently, most legacy systems are based on two kinds of framework integration techniques: the traditional object-oriented framework[8] and component-based framework[9]. In object-oriented programs, the framework is often referred to the classes in the object-oriented programming, and those classes are integrated into an application framework based on the inheritance and interface mechanism of classes. As to the framework based on components, the components are composed of units that have a well-defined interface and prescribe dependency relationship in the context. Clearly, classes and components are atomic service granularities in SOA.

2.3 Service composition

Services may exist on different levels, such as fine-grained and coarse-grained. Service granularity means the scope of functionality of a service exposed. The usually recognised granularities include technical functions (e.g., logging), business functions (e.g., getUserInfo), business transactions (e.g., openAccount) and business processes (e.g., processOrder)[10]. Sometimes, service composition among these granularities may be required since an application contains services of varying levels of granularity. The illustration of service composition[11] is shown in Fig. 1.

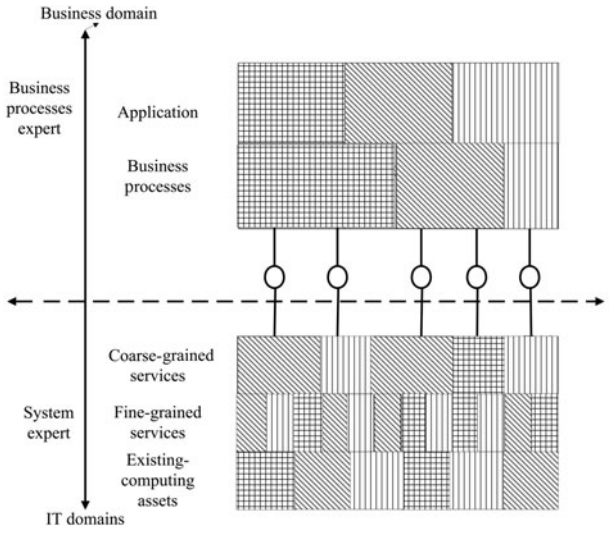

Fig. 1 Service composition

Up to now, there are three major service composition approaches[12], which include workflow-based service composition approach, artificial intelligence based automatic service composition approach, and ontology-based automatic service composition approach[13].

These service composition approaches are usually used by the service composition invoked by service consumers, namely, services composition fulfills service consumer′s requests that require the participation of several component services[14]. However, a project on SOA migration is different from developing a brand-new SOA project. The information obtained from the business logic analysis results is not enough for service composition in SOA migration. In the SOA migration situation, there exist huge amounts of data assets in a legacy system. Also, some practical information on service composition had been hidden in the usage behaviour data asset of legacy systems. Meanwhile, we found that the way that atomic services are discovered, selected, and appended into a composite service is outside these normal service composition approaches′ scope. Therefore, how to do service composition in SOA migration by means of the usage behaviour data asset of legacy systems has become a meaningful research problem.

At present, software engineering data has been collected for field studies. In order to understand well this data, data mining techniques have been used for software engineering tasks[15]. Clearly, sequence mining techniques are the good choices to solve this kind of problem.

2.4 Software engineering data and sequence mining techniques

With the rapid development of the software industry, a huge amount of software application data for understanding well their processes and products has been collected by some software companies and organisations. In order to discover useful information hidden in software engineering data, some data mining techniques have been used on SE data. Moreover, data mining[16] and its applications to SE data[17] have become a hot topic.

According to [3], software engineering tasks helped by data mining techniques include programming, static defect detection, testing, debugging, and maintenance. Data mining techniques used in mining SE data include association rules and frequent patterns, classification, clustering, automaton/grammar learning, searching/matching, concept analysis, template-based analysis, abstraction-based analysis, etc. Software engineering data that can be mined include static code bases, execution traces, deployment logs, software change history, profiled program states, profiled structural entities, bug reports/natural languages, etc.

A trace is a record of the execution of a computer program, showing the sequence of operations executed. Dynamic traces depend upon the input and are obtained by executing the program. Static traces describe potential sequences of operations extracted statically from the source code. Static traces do not depend upon input data[18]. Legacy system understanding approaches include static trace analysis and dynamic trace analysis. Dynamic analysis can obtain a process graph of the program′s execution behaviour. Through further analysis, frequently occurring interaction patterns between classes or components can be found[19]. These frequent patterns can help us understand well the software.

To make sure the dynamic traces reflect software system usages scenarios, dynamic traces should be collected

during a long period of time in a real user environment. Meanwhile, the amount of traces collected should be reasonable to be stored and processed[20]. Approaches for the extraction and compression of any relevant dynamic information from execution traces have been presented by some researchers. Eisenbarth et al.[18] presented an automatic technique to extract operation sequences from source code data. Hamou-Lhadj and Lethbridge[21] presented some compression techniques to simplify the analysis of large execution traces. Autoniol et al.[22, 23] introduced an approach for the collected execution traces reduction. Antoniol and Penta[20] proposed a probabilistic model for representing dynamic information, as well as a web-service based distributed architecture for its collection and compression.

Up to now, many kinds of data mining techniques have been applied to some sort of software engineering data, and great achievements have been made in the development of software engineering and data mining fields. Since the late 1990s, some mining approaches on SE data have been contributed[24].

Sequence mining is one of the important tasks of the data mining field. Sequence mining means to discover knowledge from a sequence database. At present, many sequence mining algorithms have been proposed. Among them, the famous sequence mining algorithms are AprioriAll and AprioriSome algorithms, which have been addressed by Srikant and Agrawal[25, 26]. Some variants based on these two algorithms are also proposed[19, 27−32].

The typical sequence mining algorithm does not consider the quantities associated with items. Recently, some researchers have added this factor into the sequence mining algorithm, which is called quantitative sequence mining approaches. Some fuzzy techniques have also been used in this field[33−35].

3 A service composition approach based on sequence mining techniques for MELS

E-learning (also called online learning or distance learning) refers to a virtual teaching/learning platform that uses a wide spectrum of technologies, such as the Internet or computer-based technologies. E-learning is still a relatively new learning technology, which changes learners′ cognitive styles as well as teachers′ teaching modes, strategies, and roles. The teaching goals can be reached by the complementation between traditional face-to-face classes and online learning.

Therefore, an e-learning system must provide complete support for different teaching and learning modes. Normally, different teachers will provide different teaching strategies to the same learners; also, different learners will adopt different learning schemes for the same contents. In this case, during the design and development of an e-learning system, it is a key problem to provide a learner the appropriate learning support environment according to the learner′s preference.

According to the theory of education, a learning process mainly includes the following learning modes: demonstration, read and analysis, interaction, exercise, learn by heart, review, test, feedback, etc. There will be different sequences for different learners for the same learning contents. Therefore, the appropriate learning support environment for each learner means a specific learning sequence (coarse–grained service) composed of existing learning contents and learning modes (fine-grained service).

The usage behaviour data resource is the set of practical operation sequences requested by all users. The practical operation sequence represents specific learning sequence which can be of benefit to the service composition in MELS. Namely, these discovered patterns will be the candidate composite services (coarse-grained) in the SOA system.

Through the above analysis, we conclude a service composition approach based on sequence mining for MELS. The main idea of this approach is to discover useful information for service composition in MELS from the angle of practice. The discovered information will be the complementation of business logic analysis results of MELS. The concrete implementation is through applying a sequence mining algorithm to mine usage behaviour data of an e-learning legacy system.

The architecture of the proposed approach having three phases is shown in Fig. 2. The data preprocess phase is to obtain the appropriate software engineering data for the pattern discovery. The pattern discovery phase is to apply a sequence mining algorithm to discover frequent patterns, which are the candidate composite services. The pattern analysis phase means understanding the results obtained by the sequence mining algorithm and drawing conclusions (service composition schemes). Clearly, the core of this approach is to develop an appropriate sequence mining algorithm for mining related data collected from the e-learning legacy system.

Through analysing the existing sequence mining algorithms, we found that they cannot meet the needs of the e-learning domain. For example, these algorithms are used to find motifs on the fixed min-support number. In order to further analyse the found patterns from the sequence database, it is not enough to just obtain motifs of a fixed min-support number. In addition, the data in e-learning domain should consider the quantities associated with items to some extent. Also, the quantity of each item in e-learning data is small (normally, it is less than 5, for example, a learner can review some learning contents 3 times or 4 times, but it is exceptional if he reviews them 10 times). Thus, if we adopt the existing quantitative sequence mining algorithms, it will increase the costs of time and space greatly, which will affect the practicability of this approach. In this case, according to the features of execution trace data on usage behaviour and needs of further pattern analysis, we propose a sequential mining algorithm to mine execution traces of an e-learning legacy system.

In the following section, we will introduce an algorithm[36] that can find all motifs on any min-support numbers and own quantitative sequence mining function to a certain extent (like a simplified quantitative sequence mining algorithm). First, some related concepts will be described. Second, we show a theorem for finding frequent patterns with min-support $h = 1$. Third, we introduce the algorithm for finding frequent patterns with min-support $h \geqslant 2$. Then, we introduce the algorithm for finding motifs

(namely, the frequent and maximal patterns). Finally, we will run the proposed algorithms on an event trace sequence database.

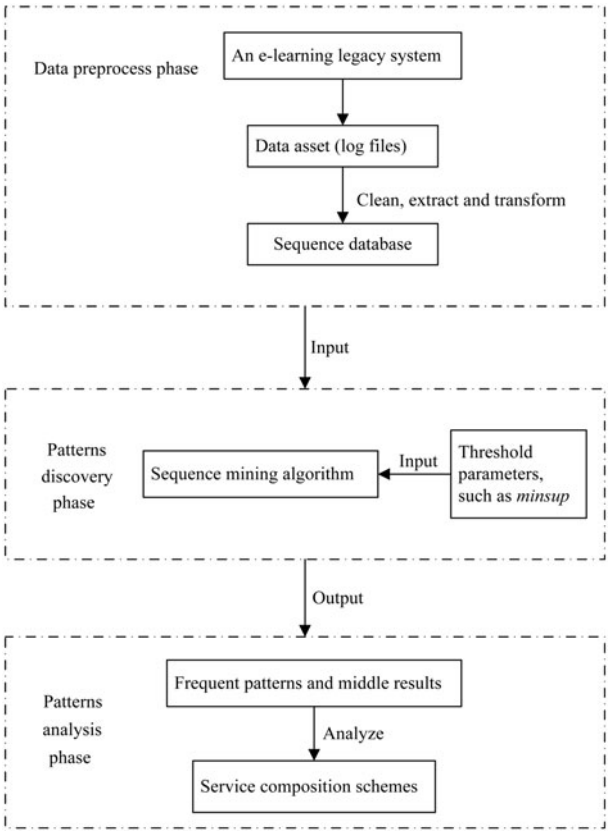

Fig. 2 The architecture of a service composition approach based on sequence mining for MELS

4 Proposed sequence mining algorithm for MELS

4.1 Concepts

For a set Θ, the number of its elements is called its size and denoted by $|\Theta|$. Given a non-empty set T, we call elements in it items and call set T transaction set. A non-empty subset of transaction set T is called itemset.

A sequence s over T is an ordered list of non-empty subsets (terms) of T, expressed as $s = A_1A_2\cdots A_n\cdots A_N$, where $\phi \subset A_n \subseteq T$, $0 < |A_n| \leqslant |T|$, for $n = 1, 2, \cdots, N$ and $N > 0$. 1-itemset is considered to be a sequence with a single term. Given a sequence $s = A_1A_2\cdots A_n\cdots A_N$, we call N the length of sequence s and denote $|s| = N$.

A non-empty set composed of a group of sequences over T is named as sequence database U over T. Given a sequence $a = A_1A_2\cdots A_n\cdots A_N$ of sequence database U, let w be any sub-sequence of a. The sub-sequence w is called a pattern supported by (or occurring in) sequence a. Given a non-empty sequence database $U = \{u_1, u_2, \cdots, u_k, \cdots, u_K\}$, the set $\max M(U, h)$ of maximal patterns with support number not less than min-support number h is called motifs over U.

The work of sequence mining is to find the motifs of a group of sequences, which means the set of frequently occurring and maximal subsequences in this sequence group.

4.2 Theorem for finding patterns with min-support $h = 1$

Motifs with min-support $h = 1$, which is denoted by $\max M(U, 1)$, form the set of maximal sequences of $M(U, 1)$, namely,

$$M(U,1) = M(U,1,1) \cup M(U,1,2)\cdots \cup M(U,1,l)$$

$$\max M(U,1) = \max\{M(U,1,1) \cup M(U,1,2)\cdots \cup M(U,1,l)\}.$$

Theorem 1. Given a sequence database U, motifs with min-support $h = 1$ are the maximal sequences of U, and vice versa. That is $\max(M(U, 1)) = \max(U)$.

The proof of this theorem can be found in [36]. In fact, for a transaction database, this theorem is also correct[37].

For example, by theorem 1, we can find the motifs $\max(M(U, 1))$ from sequence database U shown in Table 1 immediately. Since $u_3 < u_1$, $u_4 < u_2$, $u_5 < u_2$, we have $\max(M(U, 1)) = \max(U) = \max\{u_1, u_2, u_3, u_4, u_5\} = \{u_1, u_2\}$.

Table 1 (Sequence database U) This database can be expressed as a group of software trace sequences

ID	Sequence s_i	Length $\lvert s_i\rvert$
1	{1,5}{2}{3}{4}	4
2	{1}{3}{4}{3,5}	4
3	{1}{2}{3}{4}	4
4	{1}{3}{5}	3
5	{3}{3}	2

4.3 Algorithm for finding patterns with min-support $h \geqslant 2$

Algorithms for finding patterns with min-support $h = 2$, which is denoted as $M(U, 2)$[1], are shown in Algorithm 1.

Algorithm 1.

Step 1. (Find $M(U, 2, 1)$[2]) Find all 1-length (or unit) patterns with min-support $h \geqslant 2$, denoted as $M(U, 2, 1)$, from sequence database U.

Step 2. (Find $M(U, 2, 2)$) Find $M(U, 2, 2)$ from $M(U, 2, 1)$.

for each $x \in (U, 2, 1)$ do
  for each $z \in (U, 2, 1)$ do
    x right concatenates z to generate $w = xz$,
    and put w to the candidate set // Obtain
    candidate set with length 2
if ($|w^U|$[3] $\geqslant 2$) then $w \in M(U, 2, 2)$ //Check support number of the candidate patterns and get $M(U, 2, 2)$.

Step 3. (Find $M(U, 2, l)$) Find $M(U, 2, l)$ from $M(U, 2, l-1)$, $l > 2$.

for ($l = 3$; $M(U, 2, l-1) \neq \{\}$; $l++$) do
  for each $x \in M(U, 2, l-1)$ do

[1] $M(U, h)$ represents the set of patterns with min-support h.

[2] $M(U, h, l)$ represents the set of patterns with min-support h and length l.

[3] $|w^U|$ represents the number of sequences in sequence database U in which sequence w is contained. Usually, $|w^U|$ is called support number and w^U is called support group.

for each $z \in M(U,2,l-1)_{\text{last term}}$[4] do
x right concatenates z to
generate $w = xz$, and put w to
the candidate set // Obtain
candidate set with length l
if ($|w^U| \geqslant 2$) then $w \in M(U,2,l)$
//Check the support number of the
candidate patterns and get $M(U,2,l)$.

Step 4. Answer
Output the patterns set $M(U,2)$.

$$M(U,2) = M(U,2,1) \cup M(U,2,2) \cup \cdots \cup M(U,2,l).$$

Algorithms for finding patterns with min-support $h > 2$ are shown in Algorithm 2.

Algorithm 2.
Step 1. Find $M(U,2)$
Call Algorithm 1.
Step 2. Find $M(U,h)$
For ($h = 2$; $M(U; h+1) \neq \{\}$; $h++$)
do
$M(U,h+1) = M(U,h) - G(U,h)$[5] //Delete
pattern w with $|w^U| = h$ from $M(U,h)$, the
remainder belongs to $M(U,h+1)$.

4.4 Algorithm for finding motifs

Let V be a non-empty partially descending ordered set. An element $w$of set V is a maximum in V if there exists no $v \in V - w$ such that $v > w$.

Algorithm 3. For finding $\max(V)$ from V.
$\max(V) = \{\}$
for each $w \in V$
{for each $v \in V - w$
{ counter=0
if $v > w$ then counter++ }
if counter $= 0$ then $\max(V) = \max(V) \cup \{w\}$
//There exist no $v \in V - w$ such that $v > w$.
So w is a maximum in $V$and let
$w \in \max(V)$.
}
Return $\max(V)$.

This algorithm can be run by feeding SE data. In our application domain, sequences will be composed of event traces collected from source code or executed existing program. For example, a sequence database U is shown in Table 1. Items of this sequence database are classes or components.

By theorem 1, we have $\max(M(U,1)) = \max(U) = \max\{u_1, u_2, u_3, u_4, u_5\} = \{u_1, u_2\}$.

According to Step 1 of Algorithm 1, we can obtain $M(U,2,1)$ shown in Table 2 from sequence database U.

Table 2 $M(U,2,1)$ in sequence database U

1-patterns w	Support group w^U	Support number $\lvert w^U \rvert$
{1}	{u_1,u_2,u_3,u_4}	4
{2}	{u_1,u_3}	2
{3}	{u_1,u_2,u_3,u_4,u_5}	5
{5}	{u_1,u_2,u_3}	3
{5}	{u_1,u_2,u_4}	3

[4] $M(U,2,l-1)_{\text{last term}}$ represents the set of the last term of each element of $M(U,2,l-1)$, for example, if $M(U,2,2) = \{\{1\}\{2\},\{1\}\{3\},\{2\}\{3\},\{3\}\{4\}\}$, then $M(U,2,2)_{\text{last term}} = \{\{2\},\{3\},\{4\}\}$. If $M(U,2,3) = \{\{1\}\{2\}\{3\},\{1\}\{3\}\{4\},\{2\}\{3\}\{4\}\}$, then $M(U,2,3)_{\text{last term}} = \{\{3\},\{4\}\}$.

[5] $G(U,h)$ represents the set of patterns over U with support number h. $G(U,h)=\{w|\,|w^U| = h\}$.

According to Step 2 of Algorithm 1, we can obtain $M(U,2,2)$ shown in Table 3 from sequence database U.

Table 3 $M(U,2,2)$ in sequence database U

2-frequent patterns$w$	Support group w^U	Support number $\lvert w^U \rvert$
{1}{2}	{u_1,u_3}	2
{1}{3}	{u_1,u_2,u_3,u_4}	4
{1}{4}	{u_1,u_2,u_3}	3
{1}{5}	{u_2,u_4}	2
{2}{3}	{u_1,u_3}	2
{2}{4}	{u_1,u_3}	2
{3}{3}	{u_2,u_5}	2
{3}{4}	{u_1,u_2,u_3}	3
{3}{5}	{u_2,u_4}	2

According to Step 3 of Algorithm 1, we can obtain $M(U,2,2)_{\text{last term}} = \{\{2\},\{3\},\{4\},\{5\}\}$ from $M(U,2,2)$; and then, we can obtain $M(U,2,3)$ shown in Table 4.

Table 4 $M(U,2,3)$ in sequence database U

3-frequent pattern w	Support group w^U	Support number $\lvert w^U \rvert$
{1}{2}{3}	{u_1,u_3}	2
{1}{2}{4}	{u_1,u_3}	2
{1}{3}{4}	{u_1,u_2,u_3}	3
{1}{3}{5}	{u_2,u_4}	2
{2}{3}{4}	{u_1,u_3}	2

Similarly, we can have $M(U,2,3)_{\text{last term}} = \{\{3\},\{4\},\{5\}\}$ from $M(U,2,3)$; and then, we can obtain $M(U,2,4)$ shown in Table 5.

Table 5 $M(U,2,4)$ in sequence database U

4-frequent pattern w	Support group w^U	Support number $\lvert w^U \rvert$
{1}{2}{3}{4}	{u_1,u_3}	2

Step 3 of Algorithm 1 stops since M(U, 2, 5)={}.

According to Step 4 of Algorithm 1, we can obtain $M(U,2)$ shown in Table 6.

Table 6 $M(U,2)$ in sequence database U

Patterns w	Support group w^U	Support number $\lvert w^U \rvert$
{1}	{u_1,u_2,u_3,u_4}	4
{2}	{u_1,u_3 }	2
{3}	{u_1,u_2,u_3,u_4,u_5}	5
{4}	{u_1,u_2,u_3}	3
{5}	{u_1,u_2,u_4}	3
{1}{2}	{u_1,u_3}	2
{1}{3}	{u_1,u_2,u_3,u_4}	4
{1}{4}	{u_1,u_2,u_3}	3
{1}{5}	{u_2,u_4}	2
{2}{3}	{u_1,u_3}	2
{2}{4}	{u_1,u_3}	2
{3}{3}	{u_2,u_5}	2
{3}{4}	{u_1,u_2, u_3}	3
{3}{5}	{u_2,u_4}	2
{1}{2}{3}	{u_1,u_3}	2
{1}{2}{4}	{u_1,u_3}	2
{1}{3}{4}	{u_1,u_2,u_3}	3
{1}{3}{5}	{u_2,u_4}	2
{2}{3}{4}	{u_1,u_3}	2
{1}{2}{3}{4}	{u_1,u_3}	2

After obtaining $M(U,2)$, according to Algorithm 2, we can obtain $M(U,3)$, $M(U,4)$ and $M(U,5)$ through $M(U,2)$.

According to Algorithm 3, we can obtain $\max(M(U,2))$ shown in Table 7, $\max(M(U,3))$, $\max(M(U,4))$, and $\max(M(U,5))$.

Table 7 $\max(M(U,2))$ in sequence database U

Motif	Support group w^U	Support number $\lvert w^U \rvert$
{3}{3}	$\{u_2,u_5\}$	2
{1}{3}{5}	$\{u_2,u_5\}$	2
{1}{2}{3}{4}	$\{u_1,u_3\}$	2

5 Case study

According to the stated approach, the found patterns capture some high level domain concepts that can be helpful for program understanding and further service composition. To further explain this approach, in the following sub-sections, we will take as examples the service compositions of learning schemes in the target SOA system.

5.1 An e-learning legacy system

An e-learning system for middle school students shown in Fig. 3 has been developed by Ideal Information Technology Institute of Northeast Normal University in China[38, 39]. This e-learning system consists of well-designed object-oriented programs that include some key classes. Some classes work together to complete some functionality. However, the structure of this system cannot meet the user′s needs; hence we want to migrate this system to a service-oriented architecture shown in Fig. 4[40].

In MELS, service composition can benefit from mining execution trace data on usage behaviour from this e-learning legacy system. Namely, some coarse-grained services can be added into the new system from the point of view of users′ practices.

In the following example, we take service grains of learning schemes in the target SOA system as examples. According to the suggestions of educational experts and excellent teachers, there exist 8 learning modes (items), such as demonstration (denoted by {1}), read and analysis (denoted by {2}), interaction (denoted by {3}), exercise (denoted by {4}), learn by heart (denoted by {5}), review (denoted by {6}), test (denoted by {7}) and feedback (denoted by {8}). An ordered list of these learning modes is named as a learning scheme, which can be taken as a sequence. For example, the sequence of {read and analysis}{interaction}{exercise} (for short, {2}{3}{4}) is a learning scheme, which means a learner first reads and analyses related learning materials, and then communicates with the system to get more detailed explanations on his questions, and then does some exercises to strengthen his learning. The users can organise a learning scheme by themselves in this e-learning legacy system. We collect learning schemes from different learners to form a sequences database. In order to discover which learning patterns (supported by most learning schemes) are popular for most learners, the proposed algorithm is used to mine this learning scheme sequences database. The mined frequent and maximal learning patterns (motifs) can be the candidates of coarse-grained services on the learning scheme module in the target SOA system.

5.2 A new software architecture

In Fig. 4, the software architecture is a flexible and hierarchical reusable architecture based on domain-specific software architecture (DSA) and software product lines (SPL). Under this architecture, we encapsulate business logic into the software component first, and then weave the unchanged or uneasy changing parts of the e-learning domain logic into respective domain frameworks, and implement the easy changing parts as pluggable user interfaces in order to insert or delete or replace the components according to different requirements in the future.

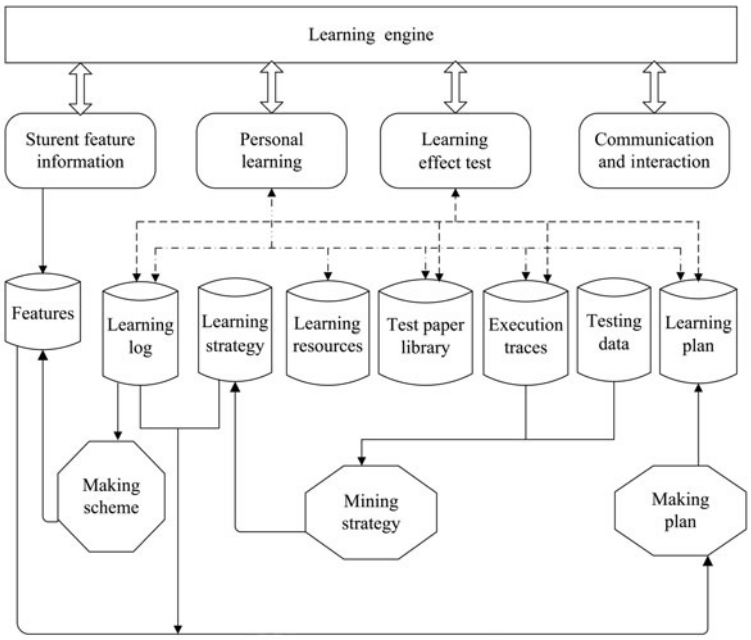

Fig. 3 The e-learning system structure

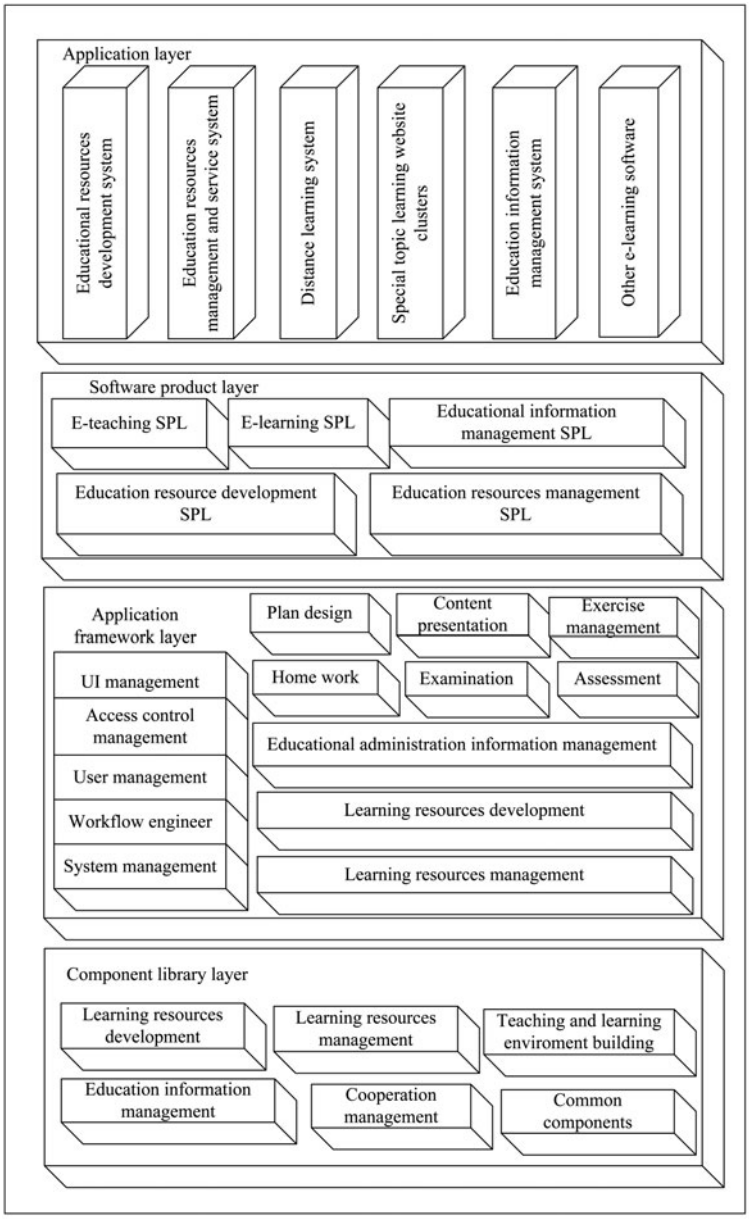

Fig. 4 The e-learning platform software architecture

Subsequently, we build e-learning software product lines based the domain frameworks. Consequently, we achieve the goal of fast-building and integration of e-learning systems via those e-learning software product lines. By using this software architecture, instructors can build customised application software systems in a visual studio just like building blocks.

This architecture is divided into four levels: the application layer, the product line layer, application framework layer, and component library layer.

1) Product line layer

It mainly includes five SPL, which are e-teaching supporting SPL, e-learning supporting SPL, education information managing SPL, education resource development SPL and education resources management and service SPL. This layer is responsible for providing reusable domain solution of e-learning, describe the functional and non-functional requirements of components that compose the various product line architectures and the associated relation between these components, and realise the building of specific software application systems. This layer is the core of the platform software architecture. Each product line is designed for a specific goal of domain application and can support a group of buildings of similar application systems. Through selecting and assembling different frameworks and components, instructors can quickly instantiate a customised software system.

2) Application layer

This layer is a specific group of e-learning software systems that are built via different product lines.

3) Application framework layer

This layer is the collection of the public frameworks and

domain specific frameworks. Each framework is composed of a series of related components, component associated relations, and restrictions. It is the collection of associated components that solve certain sub problems of the product line. The public framework mainly includes user interface (UI) management, access control management, user management, workflow engineer, and system management, all of which are common frameworks to the product architecture. Meanwhile, the domain specific frameworks are designed according to different teaching and learning activities design. It covers teaching and learning plan design, contents presentation, exercise management, homework management, examination management, question and answer management, assessment management, learning resource development, learning resource management, and educational administration information management. By introducing the application framework layer, we reduce the complexity of the architecture of the product line and increase the reusable efficiency greatly.

4) Component library layer

This layer is the foundation of the platform software architecture. It is a reusable collection of software units that is already validated by other projects. It mainly includes the component library of educational resources development, the component library of educational resources management, the component library of building teaching and learning environment, the component library of educational information management, the component library of cooperation, and the common component libraries for software system management. Moreover, the component library layer is extensible and distributed. As long as the third-party components meet with the certain agreement of the application framework, they can be added to the component library layer.

In order to integrate components into different application frameworks, we also present a framework weaving approach based on Web Services and XML data bus. The main principle is based on the standard protocols of UDDI/WSDL/SOAP to encapsulate e-learning objects into reusable Web Services components. Meanwhile, we build an XML data bus for the coordination and communication between components to realise components plugging in or binding dynamically. The XML data bus is a pool of data entities, and is composed of both XML and XML schema. In this data pool, the XML structures represent an existing set of classes (which deal with business data in an object model), and the XML schema is used to build the object models for data from XML structures. Based on the XML data bus, each component (encapsulated into web service) only reads and writes the XML data to accomplish its business logic without the need of communicating with other components.

5.3 Application of proposed approach

The preparation, collection and extraction of execution trace data are the basis for the application of the proposed approach.

Developments of education software and resource[41] have made it possible for educational institutes to collect and store huge amounts of teaching and learning execution trace data[42]. Execution scenario refers to which functionality of the program gets executed. It is very important since it influences the results of the technique[19]. Thus, we should establish an execution scenario for collecting trace data according to our purposes. For example, in order to discover popular learning patterns, we should take the learning scheme module in legacy systems as execution scenario to collect and extract the related trace data.

To ensure the quality of data, we collected the related data from 20 middle schools that are using this e-learning system. In each school, we chose 5 classes of 50 students as test objects. We organised 48 tests during four months. The students can adopt a learning scheme during one test, that is, each student can provide one record in one test. In total, we can get about 240 000 records. The size of each record is about 0.1 k. Thus, the data volume is about 24 M, that is,

$$20 \times 5 \times 50 \times 48 \times 0.1\,\mathrm{k} = 24000\,\mathrm{k} = 24\,\mathrm{M}.$$

The log sub-system of this e-learning system collected these records and stored them in table format in an SQL server database. The collected original data with the main columns is shown in Table 8.

In order to reduce the data volume, we should extract the related columns to our mining purposes from the original data. The related columns are extracted and stored in a new structural file. For example, in order to discover popular learning patterns, the ID column and learning scheme column were extracted from the structural file shown in Table 8. A sequence database on learning schemes trace data shown in Table 9 can be obtained. The size of one record in this structural file is 14 bytes. Thus, the data volume of the mined sequence data is 3.2 M, that is,

Table 8 The collected original data

ID	Learning content	Starting time	Knowledge node code	Learning scheme*	Learning source code	Score
...	...	...	...	...	...	...
7021	Math in grade 7	2007-05-04, 14:03	2.2.3	1,2,3,4,7,8,37	172020,172118,172002,172010, 172102, ···	*B*
...	...	...	...	...	...	...
7043	Math in grade 7	2007-05-04, 14:10	2.2.2	1,4,3,5, 7,8	172031,172118,172015,172011, 172112, ···	A
...	...	...	...	...	...	...
7101	Math in grade 7	2007-05-04, 14:05	2.2.1	2,5,3,4,6,7,8	172020,172118,172002,172036, 172108, ···	C
...	...	...	...	...	...	...

*Learning mode "demonstration" is denoted as 1, "read and analysis" is denoted as 2, "interaction" is denoted as 3, "exercise" is denoted as 4, "learn by heart" is denoted as 5, "review" is denoted as 6, "test" is denoted as 7, and "feedback" is denoted as 8.

$$\frac{20 \times 5 \times 50 \times 48 \times 14}{1024} = 3281\,k = 3.2\,M.$$

Table 9 (Sequence database) This database can be expressed as a group of software trace sequences

ID	Sequences s_i	Length $\lvert s_i \rvert$
1	{1}{2}{3}{4}{7}{8}{3}{7}	8
2	{1}{4}{3}{5} {7}{8}	6
3	{5}{1}{2}{4}{3}{7}{5}{8}	8
4	{1}{2}{4}{3}{7}{8}{5}	7
5	{3}{2}{4}{3}{1}{6}{7}{8}	8
6	{7}{8}{3}{5}{4}{6}{7}{8}	8
7	{2}{1}{2}{4}{3}{7}{8}	7
8	{4}{3}{2}{7}{8}	5
9	{2}{1}{2}{4}{3}{7}{8}	7
10	{4}{3}{2}{3}{7}{8}	6
...	...	...

This sequence data of 3.2 M is the input of the proposed sequence mining algorithm described in Section 4. The time complexity of this algorithm is subject to the length of each sequence, the number of items and the size of records in sequence database. In our application, the average length of each sequence is about 12 since the elements of each sequence can be repeated such as {2}{1}{4}{4}{2}{4}; the number of items is 8; the size of records in structural file is 3.2 M. In our experiment, we adopted hash-tree data structure. In this case, the time cost of this algorithm is about one minute. Clearly, the performance of this algorithm is acceptable.

After obtaining the frequent patterns and middle results returned from the pattern discovery phase, we should analyse and refine the results. Normally, this kind of work will be done by project architects and domain experts together. The analysed results will form service composition schemes, which are the complementation of business logic analysis results. These complementary services can provide information for further improving this e-learning system, such as, some services can become the recommended e-learning patterns to learners; some services can direct the adding, deleting, updating or combining of some classes or functionality; etc.

For example, through analysing the results, we found that 68 % of the users like to adopt the learning scheme like {demonstration}{analysis and read}{exercise} {interaction} {test}{feedback}, namely, sequence {1}{2}{4}{3}{7}{8} is a motif of the sequence database shown in Table 9. However, in the existing system, this pattern was not included in the recommended learning patterns. In the new system, this pattern has been identified as a composite service and stored in a service library in the SOA system.

For this execution scenario, we set up some investigation questionnaires to collect satisfaction data. Our method includes an online survey of 20 middle schools (the users of this e-learning system) and in-depth interviews with 30 education experts and excellent teachers. The investigation result is 90% the same with the result obtained from this approach. After doing many similar validations, we conclude that our approach is promising.

6 Conclusions

In the education domain, there are a number of e-learning legacy assets with useful practical experiences, such as usage behaviour data, etc. The usage behaviour data is the set of practical operation sequences requested by all users. The hidden patterns in this data will provide users′ practical experiences, which can be of benefit to the service composition in MELS. These discovered patterns will be the candidate composite services (coarse-grained) in the SOA system. In order to discover these coarse-grained services in MELS, we proposed a service composition approach based on sequence mining algorithm. The services discovered by this approach are the necessary complementation of normal service composition approaches such as artificial intelligence (AI) based and ontology-based approaches. The core of this approach is to develop an appropriate sequence mining algorithm for mining the usage behaviour data. According to the features of execution trace data on usage behaviour and needs of further pattern analysis, we improved a sequential mining algorithm to mine this kind of legacy data. This improved algorithm can find all motifs on any min-support numbers and provide quantitative sequence mining function to a certain extent. To validate this approach, some investigations were done. The investigation result is 90 % the same with the result obtained through our approach. In the future work, we will try to do more research on business process choreography and SOA migration decision scheme for MELS by using data mining techniques.

References

[1] K. Bennett. Legacy systems: Copying with success. *IEEE Software*, vol. 12, no. 1, pp. 19–23, 1995.

[2] A. Brown, S. Johnston, K. Kelly. Using Service-oriented Architecture and Component-based Development to Build Web Service Applications, A Rational Software White Paper, Rational Software Corporation, pp. 11–15, 2002.

[3] T. Xie, A. E. Hassan. Mining software engineering data. [Online], Available: http://ase.csc.ncsu.edu/dmse/dmse-icse07-tutorial.ppt#256, August 30, 2008.

[4] Z. Zhang, H. Yang. Incubating services in legacy systems for architectural migration. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, IEEE, pp. 196–203, 2004.

[5] S. Inaganti, G. K. Behara. Service identification: BPM and SOA handshake. [Online], Available: http://www.bptrends.com, March 2007.

[6] G. Canfora, A. R. Fasolino, G. Frattolillo, P. Tramontana. A wrapping approach for migrating legacy system interactive functionalities to service oriented architectures. *Journal of Systems and Software*, vol. 81, no. 4, pp. 463–480, 2008.

[7] D. S. Linthicum. *Next Generation Application Integration: From Simple Information to Web Services*, Addison Wesley Press, 2003.

[8] M. Mattson. *Evolution and Composition of Object-oriented Frameworks*, Kaserntryckeriet AB, Karlskrona, Sweden, 2000.

[9] W. H. Hu, W. Zhao, S. K. Zhang. Study of application framework meta-model based on component technology. *Journal of Software*, vol. 15, no. 1, pp. 1–8, 2004. (in Chinese)

[10] A. Papkov. Develop a migration strategy from a legacy enterprise IT infrastructure to an SOA-based enterprise architecture. IBM Developer Works, [Online], Available: http://www.ibm.com/developerworks/library/ws-migrate2soa/index.html?S_TACT=105AGX20&S_CMP=EDU, November, 2007.

[11] J. Hanson. Coarse-grained interfaces enable service composition in SOA. JavaOne, [Online], Available: http://articles.techrepublic.com.com/5100-10878_11-5064520.html August 29, 2003.

[12] S. G. Deng, J. Wu, Y. Li, Z. H. Wu. Automatic web service composition based on backward tree. *Journal of Software*, vol. 18, no. 8, pp. 1896–1910, 2007.

[13] M. DiBernardo, R. Pottinger, M. Wilkinson. Semiautomatic web service composition for the life sciences using the BioMoby semantic web framework. *Journal of Biomedical Informatics*, vol. 41, no. 5, pp. 837–847, 2008.

[14] Z. Maamar, D. Benslimane, P. Thiran, C. Ghedira, S. Dustdar, S. Sattanathan. Towards a context-based multi-type policy approach for Web services composition. *Data and Knowledge Engineering*, vol. 62, no. 2, pp. 327–351, 2007.

[15] J. Z. Li, Z. P. Zhang, B. Qiao, H. J. Yang. A component mining approach to incubate grid services in object-oriented legacy systems. *International Journal of Automation and Computing*, vol. 3, no. 1, pp. 47–55, 2006.

[16] J. Han, M. Kamber. *Data Mining: Concepts and Techniques*, China Machine Press, 2006.

[17] M. Mendonca, N. L. Sunderhaft. Mining Software Engineering Data: A Survey, A DACS State-of-the-Art Report, No. SPO700-98-D-4000, Data & Analysis Center for Software, 1999.

[18] T. Eisenbarth, R. Koschke, G. Vogel. Static trace extraction. In *Proceedings of IEEE Working Conference on Reverse Engineering*, IEEE, Richmond, VA, USA, pp. 128–137, 2002.

[19] A. Zaidman, T. Calders, S. Demeyer, J. Paredaens. Applying webmining techniques to execution traces to support the program comprehension process. In *Proceedings of the 9th European Conference on Software Maintenance and Reengineering*, IEEE, pp. 134–142, 2005.

[20] G. Antoniol, M. D. Penta. A distributed architecture for dynamic analyses on user-profile data. In *Proceedings of the 8th European Conference on Software Maintenance and Reengineering*, IEEE, Tampere, Finland, pp. 319–328, 2004.

[21] A. Hamou-Lhadj, T. C. Lethbridge. Compression techniques to simplify the analysis of large execution traces. In *Proceedings of the 10th International Workshop on Program Comprehension*, IEEE, Paris, France, pp. 159–168, 2002.

[22] G. Antoniol, M. D. Penta. Library miniaturization using static and dynamic information. In *Proceedings of IEEE International Conference on Software Maintenance*, IEEE, Amsterdam, The Netherlands, pp. 235–244, 2003.

[23] G. Antoniol, M. D. Penta, M. Neteler. Moving to smaller libraries via clustering and genetic algorithms. In *Proceedings of the 7th European Conference on Software Maintenance and Reengineering*, Benevento, Italy, pp. 307–316, 2003.

[24] T. Xie. Mining software engineering data bibliography. [Online], Available: http://ase.csc.ncsu.edu/dmse/setasks, October, 2008.

[25] R. Agrawal, R. Srikant. IBM Research Report RJ9910: Mining Sequential Patterns, IBM Research Division, Almaden research center, CA, USA, 1994.

[26] R. Srikant. *Fast algorithms for Mining Association Rules and Sequential Patterns*, Madison, USA: University of Wisconsin, 1996.

[27] R. Srikant, R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology, Lecture Notes in Computer Science*, Springer, vol. 1057, pp. 1–17, 1996.

[28] M. El-Ramly, E. Stroulia, P. Sorenson. From run-time behaviour to usage scenarios: An interaction-pattern mining approach. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 315–324, 2002.

[29] A. Hamou-Lhadj, T. C. Lethbridge. An efficient algorithm for detecting patterns in traces of procedure calls. In *Proceedings of ICSE Workshop on Dynamic Analysis*, Portland, OR, USA, pp. 33–36, 2003.

[30] T. Denmat, M. Ducassé, O. Ridoux. Data mining and cross-checking of execution traces: A re-interpretation of Jones, Harrold and Stasko test information visualization. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, ACM, Long Beach, California, USA, pp. 396–399, 2005.

[31] M. Christodorescu, S. Jha, C. Kruegel. Mining specifications of malicious behaviour. In *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, ACM, Dubrovnik, Croatia, pp. 5–14, 2007.

[32] S. Thummalapenta, T. Xie. PARSEWeb: A programmer assistant for reusing open source code on the web. In *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, ACM, Atlanta, Georgia, USA, pp. 204–213, 2007.

[33] C. Kim, J. H. Lim, R. T. Ng, K. Shim. SQUIRE: Sequential pattern mining with quantities. *Journal of Systems and Software*, vol. 80, no. 10, pp. 1726–1745, 2007.

[34] Y. L. Chen, T. C. K. Huang. A new approach for discovering fuzzy quantitative sequential patterns in sequence databases. *Fuzzy Sets and Systems*, vol. 157, no. 12, pp. 1641–1661, 2006.

[35] Y. L. Chen, T. C. K. Huang. A novel knowledge discovering model for mining fuzzy multi-level sequential patterns in sequence databases. *Data & Knowledge Engineering*, vol. 66, no. 3, pp. 349–367, 2008.

[36] Z. Zhang, L. Zhang, S. Zhong, J. Guan. A new algorithm for mining sequential patterns. In *Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery*, IEEE, Jinan, PRC, pp. 625–629, 2008.

[37] Z. Zhang, L. Zhang, S. Zhong, J. Guan. Improving algorithm Apriori for data mining. In *Proceedings of the 8th International FLINs Conference on Computational Intelligence in Decision and Control*, Madrid, Spain, pp. 17–22, 2008.

[38] Ideal Information Technology Institute of Northeast Normal University in China, [Online], Available: http://www.dsideal.net/.

[39] NorthEast Normal University (NENU) in China, [Online], Available: http://www.nenu.edu.cn/.

[40] D. Zhou, Z. Zhang, S. Zhong, P. Xie. The design of software architecture for e-learning platforms. In *Proceedings of the 3rd International Conference on E-learning and Games, Lecture Notes in Computer Science*, Springer, Nanjing, PRC, vol. 5093, pp. 32–40, 2008.

[41] S. Zhong, J. Li, Z. Zhang, Y. Zhong, J. Shang. Methods on educational resource development and application. In *Proceedings of the 3rd International Conference on E-learning and Games, Lecture Notes in Computer Science*, Springer, Nanjing, PRC, vol. 5093, pp. 290–301, 2008.

[42] C. W. Dai, S. H. Yang, R. Knott. Data transfer over the Internet for real time applications. *International Journal of Automation and Computing*, vol. 3, no. 4, pp. 414–424, 2006.

Zhuo Zhang graduated from Jilin University, PRC in 1986. She received the M. Sc. degree from Jilin University, PRC in 1999, and from Montreal University, Canada in 2003. She is currently a professor at Software School in Northeast Normal University, PRC and a Ph. D. candidate of De Montfort University, UK.

Her research interests include data mining, software architecture, and intelligent tutoring, especially the service-oriented architecture (SOA) migration from the legacy systems such as migrating the educational software to SOA.

E-mail: zzhang@nenu.edu.cn

Dong-Dai Zhou graduated from Changchun University of Science and Technology (CUST), PRC in 1992. He received the M. Sc. degree from CUST in 1997 and the Ph. D. degree from the Jilin University, PRC in 2001. He is currently a professor at School of Software, Northeast Normal University, PRC.

His research interests include software architecture and software code auto-generation, especially the architecture design and codeless development of e-learning software.

E-mail: ddzhou@nenu.edu.cn

Hong-Ji Yang received the B. Sc. and M. Phil. degrees from Jilin University in 1982 and 1985, respectively, and the Ph. D. degree from Durham University, UK in 1994. Currently, he is a professor at the Software Technology Research Laboratory, Faculty of Technology, De Montfort University, UK and leads the Software Evolution and Reengineering Group. He served as a program co-chair at *IEEE International Conference on Software Maintenance* in 1999 and the program chair at *IEEE Computer Software and Application Conference* in 2002.

His research interests include software engineering and pervasive computing.

E-mail: hyang@dmu.ac.uk

Shao-Chun Zhong graduated from Northeast Dianli University, PRC in 1982. He received the M. Sc. degree from Jilin University, PRC in 1989 and the Ph. D. degree from Jilin University, PRC in 1994. He is a professor at the Northeast Normal University, PRC and he is the director of Engineering Research Center of e-learning Technologies, Ministry of Education, PRC.

His research interests include artificial intelligence, intelligent tutoring, and education technology.

E-mail: sczhong@sina.com (Corresponding author)