2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Business Process Decomposition based on Service Relevance Mining

Zicheng Huang, Jinpeng Huai, Xudong Liu, and Jiangjun Zhu
School of Computer Science & Engineering, Beihang University, Beijing, China
{huangzc,huaijp,liuxd,zhujj}@act.buaa.edu.cn

Abstract

Reuse is an important mechanism for improving the efficiency of software development. For Internet-scale software produced through service composition, the simple reuse granularity at service is often inefficient due to the large number of available services. This paper proposes a novel architecture which enables efficient reuse of process fragments. In the proposed architecture, services are organized into a network, called Service Composition Network (SCN), based on their co-occurence in the existing composite services. The reusable process fragments are extracted by decomposing existing composite services according to both the structural constraint of the process and the relevance of services in the same process fragment. The design principles and a prototype implementation of this architecture are presented, the performance of the proposed approach is analyzed, and an application is described to demonstrate the effectiveness of it.

1. Introduction

In the open network environment, software requirements are often diverse and constantly evolving, thus are difficult to meet with traditional software development methods. These requirements can be better handled by service-oriented technologies where users can quickly develop software by composing available web services [1].

Recently, process-aware service composition methods, which use process definitions to specify possible interactions and operation invocations between web services, are becoming increasingly popular [2,3]. In these methods, business process models are first created to meet business requirements and facilitate communications between them and developers. Orchestration and refinement are then employed to create business processes based on the pre-defined models in a top-down fashion.

Similar to traditional software development, creating new services in an efficient and low-cost way is an important goal of service composition. However, designing a new composite service through top-down methods is a highly complex and time consuming task because these methods still face some challenging issues. First, with the wide application range of SOA technologies, the number of existing reusable services is increasing rapidly. Meanwhile, the business logic of composite service is becoming more and more complicated. Developers need to learn sufficient query technologies and domain knowledge to discover appropriate services for composition. Secondly, when incorporating new requirements into existing services such as adding new components or adapting existing ones into new ones, the reuse granularity of composite services is low, since the reusable part of such services, which we call *business knowledge*, is embedded in many processes supported by diverse technologies depending on the process modeling languages. This knowledge can only be reused if it can be extracted as an independent and uniform process fragment.

To overcome these shortcomings of traditional top-down approaches, a novel bottom-up fashion for service composition is proposed [4], which aims at exploring the full potential of the service space without prior knowledge of what exactly is in it. This will alleviate the burden of service developers during service discovery and thus increases the automation of service composition. To apply the bottom-up approach in process-aware service composition, we propose a novel architecture of the *Business Knowledge Repository* which enables reuse of business knowledge in the form of process fragments. In the proposed architecture, services are organized into a network, called *Service Composition Network* (SCN), based on their co-occurence in existing composite services. The reusable process fragments are built by decomposing existing composite services according to the structural constraint of the process as well as the relevance of services in the same process fragment. Algorithms for decomposing a business process are described and a prototype system is implemented and evaluated.

This paper makes the following major contributions:

1. We design a *business knowledge repository* for reuse of process fragments in a bottom-up fashion of service composition.
2. We propose a *service composition network* and a service relevance mining method to evaluate the co-occurrence of web services.
3. Unlike existing program decomposition approaches, we propose primitives for decomposing a business process into a hierarchy of reusable process fragments that takes into account both structural constraints and service relevance in the generated fragment.
4. We do various experiments and analyses on a real dataset of bioinformatics workflows. Initial results show the proposed architecture is effective and practical.

The rest of this paper is organized as follows. Section 2 addresses a motivating scenario for business process decomposition, and introduces the basic idea of our architecture. In section 3 some preliminaries are presented. In section 4 we describe the approach of service relevance mining and algorithms for decomposing business processes. Experimental evaluations of the performance and effectiveness of the proposed architecture are presented in section 5, followed by a review of related work in section 6 and a conclusion in section 7.

2. A motivating scenario

For process-aware service composition, business analysts or domain experts usually use process modeling languages such as BPMN [5] to generate abstract business models which describe the business requirement of composite services. These models are then passed to developers. A further orchestration and refinement of the models will usually be needed to finally transform them into low-level executable business processes. During the refinement, a set of reusable process fragments extracted from existing composite services will help to increase the efficiency. Fig. 1 (upper) shows the context of service composition, which includes a business model *M* and a set of reusable process fragments. In this scenario, the abstract fragments F_1, F_2 and F_3 in *M* are refined to concrete fragments F_1', F_2' and F_3' respectively. In comparison to the direct orchestration of these fragments, the development efficiency is obviously improved. The lower part of Fig. 1 illustrates a service repository which includes the business knowledge repository and all the existing composite services.

In this paper, we discuss the business process decomposition approach for existing composite services. CS_n in Fig. 1 represents a composite service

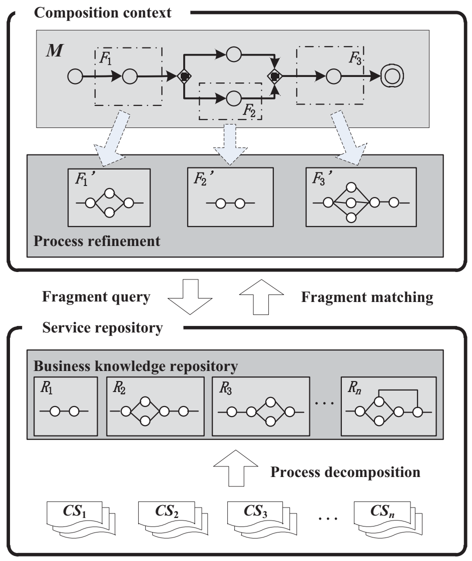


Figure 1. Process decomposition and its application in process-aware service composition

in service repository. R_n is the reusable process fragment derived from it. The fragments are then matched according to the query of developers and the matched ones may provide a wealth of information for the further orchestration of the composite service.

In order to implement the business process decomposition mentioned above, we argue that the following criteria are needed for describing a reusable process fragment. First, it should be independent in structure and connectable for other fragments. Secondly, it should preserve the execution semantics as it does in the original business process. Finally, it should be the representation of a complicated task in a certain domain. So our process decomposing algorithm should take into account both the structural constraints of process and the domain relevance of services.

3. Preliminaries

In this section we give the definition of SCN, and recall a few standard definitions which will be required for understanding the following descriptions of the proposed algorithm.

3.1. Service composition network

We build an overlay network called *Service Composition Network* (SCN) to capture the relevance of web services. A node of SCN represents a web service, and there is at most one undirected weighted edge between two nodes if the corresponding services both exist in at least one composite service. A weight function is assigned to each node to describe the existing times of the service. The weight function is also assigned to each edge to describe the times that the corresponding services of the edge both exist in the same composite service.

The construction of SCN can be easily realized by analyzing all composite services in the service repository. When analyzing a composite service, we obtain a list of web services L_S in it and then check this list. Every time we get a service s_n from L_S, we compare the service endpoint address with those of services which are already in SCN. If there is no service having the same endpoint address as the current checked one in L_S, then it is a new encountered one and a new node representing this service is added to SCN and the weight function denoted as $W(s_n)$ is set to 1. Otherwise, $W(s_n)$ is increased by 1. We then analyze each service pair (s_i , s_j) in L_S. The edges in SCN is adjusted in two cases: if an edge between s_i and s_j exist in SCN, then the weight function denoted as $W(e(s_i , s_j))$ is increased by 1. Otherwise, a new edge $e(s_i , s_j)$ is added to SCN and $W(e(s_i , s_j))$ is set to 1.

3.2. Workflow graph and process fragment

Currently, business process modeling languages are widely used in process-aware service composition, and composite services are described in an inconsistent way due to the use of different modeling languages. In order to facilitate a consistent description for the decomposition, we employ a more basic form named as *workflow graph* [6], which is a directed graph including a set of tasks, coordinators and control flow relations. Our definition of reusable process fragment is based on the workflow graph. And from now on, when we talk about process fragment, it is the fragment of a workflow graph.

The business knowledge is a representation of every independent business task in business process. Thus it should be captured and described by an independent structure. In our decomposing algorithm, we use the *single-entry-single-exit fragment* (SESE fragment for short) in [7] to express the business knowledge. There are two constraints in the definition of SESE fragment: (1) the fragment has only two edges in common with the other part of the same workflow graph, called the entry and the exit edge respectively; (2) each node in the fragment is on a path from the entry edge to the exit edge. These constraints make sure that the fragment is an independent and well-structured representation of the underlying business knowledge.

4. Building reusable process fragments

The building of reusable process fragments is based on the following heuristic: *web services tend to cooperate with each other to complete a complex business task if they occur together and connect with each other in the same composite service*.

Directly applying the web service clustering [8] or workflow parsing [7] algorithm in our context will not work well, since two web services in the same cluster may not be able to connect to each other, or otherwise, web services in the same SESE fragment may not express the same business knowledge since the workflow parsing algorithm does not consider business information. Our process decomposition approach is a hybrid of the web service clustering and workflow parsing algorithm. We exploit both the co-occurrence and connectivity of web services in composite services to form the criteria of the ideal process decomposition approach which builds up a hierarchy of *reusable process fragments* (RPF). The connectivity of web services is described by the SESE fragment which is formed by a set of connected web services. The co-occurrence of web services is measured by service relevance according to the clustering analysis of SCN. The two criteria are put together to decompose existing composite services into RPFs. We will first describe the criteria for an ideal business process decomposing algorithm in section 4.1, and then describe the algorithm in detail in following sections.

4.1. Criteria for an ideal decomposition

Ideally, business process decomposition results should have the following characteristics:

1. The process fragments should be reused independently. That is, the RPFs should be well-structured which meet the following two conditions. First, a RPF should be connected so that every node in it can be reached from the entry edge. Second, every RPF should have only the entry edge or the exit edge or both of them in common with another fragment.
2. The *cohesion* of a RPF—the relevance between web services inside the fragment—should be strong; the *correlation* between RPFs of the same business process—the relevance between web services in different RPFs—should be weak. The definitions of these criteria and detailed solutions to support them are presented in section 4.2.

Our process decomposing algorithm takes into account the criteria mentioned above, which is a hybrid of the structural parsing of business process and the service relevance mining of SCN. The algorithm is based on an iterative analysis of the workflow graph, which is described in detail in section 4.3.

4.2. Service relevance mining based on SCN

We use SCN to measure the relevance of two services or services in the same RPF, which is a key step for the decomposition of business process.

According to the previously discussed heuristic of building reusable process fragments, we measure the relevance of services by exploiting their conditional probabilities of occurrence in existing composite services. The service relevance rule we are interested in can be described in the following form:

$$s_i \rightarrow s_j\ (p, c) \tag{1}$$

In this rule, s_i and s_j are two services. The *support*, p, is the probability that s_i and s_j both occur in a composite service; *i.e.*, $p = P(s_i \cup s_j) = \frac{\sigma(s_i \cup s_j)}{N}$, where N is the total number of composite services in service repository, and $\sigma(s_i \cup s_j)$ is the number of composite services that contain both s_i and s_j. The *confidence*, c, is the probability that s_j occurs in a composite service, given that s_i is known to occur in it; *i.e.*, $c = P(s_i|s_j) = \frac{\sigma(s_i \cup s_j)}{\sigma(s_i)}$, where $\sigma(s_i)$ is the number of composite services that contain s_i. These rules can be efficiently computed based on SCN. Here we assume that the total number of composite services N in the service repository has been computed as the result of constructing SCN. According to the definition of SCN, the value of $\sigma(s_i \cup s_j)$ is equal to the weight of the corresponding edge that injects to nodes s_i and s_j in SCN. And the value of $\sigma(s_i)$ is equal to the weight of node s_i that can be obtained directly.

One of our goals is to extract the frequently occurring business tasks represented by process fragments. We thus define the *support* of a process fragment as the probability that all services in it occur in a composite service to measure this property. Given a process fragment F, the support of F is defined as follow:

$$support_F = \frac{\sigma(\cup_{s_i \in F} s_i)}{N} \tag{2}$$

In traditional data mining theory [9], *cohesion* is defined as the sum of squares of Euclidean distances from each point to the center of the cluster it belongs to; *correlation* is defined as the sum of squares of distances between cluster centers. This definition does not apply well in our context given that the center of a RPF is unknown. We hence quantify the *cohesion* and *correlation* of RPF based on service relevance.

Given a process fragment F in a business process P, we define the *cohesion* of F as the average value of *confidence* in relevance rules of all service pairs in the fragment. Formally,

$$coh_F = \frac{\sum\{c_{ij} | s_i, s_j \in F,\ s_i \neq s_j,\ s_i \rightarrow s_j(p_{ij}, c_{ij})\}}{N_F(N_F - 1)} \tag{3}$$

where $s_i \rightarrow s_j(p_{ij}, c_{ij})$ is the relevance rule of s_i and s_j, and N_F is the number of services in F. As a special case, the cohesion of a single-service fragment is 1.

We use the *correlation* between F and its complement according to P to measure the relevance between services inside and outside of F. The complement of F denoted by F^c consists of services in P but not in F and the corresponding control flow relation between them. We define the *correlation* between F and F^c as the average value of *confidence* in relevance rules of all service pairs cross them. Notice that the rule $s_i \rightarrow s_j\ (p_{ij}, c_{ij})$ and $s_j \rightarrow s_i\ (p_{ji}, c_{ji})$ may have different support and confidence values. So,

$$cor_{F,F^c} = \frac{W_1 + W_2}{2 \cdot N_F \cdot N_{F^c}} \tag{4}$$

where

$$W_1 = \sum\{c_{ij} | s_i \in F,\ s_j \in F^c, s_i \rightarrow s_j(p_{ij}, c_{ij})\}, \tag{5}$$

$$W_2 = \sum\{c_{ij} | s_i \in F^c, s_j \in F, s_i \rightarrow s_j(p_{ij}, c_{ij})\}, \tag{6}$$

and N_{F^c} is the total number of services in F^c.

Based on the *Modularization Quality Function* which measures the quality of decomposing source code components and relations into subsystem clusters [10], we define the *cohesion/correlation score* as the quality function of a fragment to evaluate the effectiveness of building it. Formally, we have:

$$quality_F = \frac{coh_F}{cor_{F,F^c}} \tag{7}$$

Our goal is to obtain high $quality_F$, which indicates a tight relevance inside a fragment and a loose relevance between the fragment and other parts of the same business process. We say that F is a RPF if its $quality_F$ is greater than a threshold q_r. The threshold q_r is chosen manually to be the value that best separates reusable and insignificance process fragments. To measure the quality of a process decomposition $\mathcal{D}$, the average value of qualities of all RPFs derived by $\mathcal{D}$ is defined as the overall quality. Formally,

$$quality_D = \frac{\sum_{F \in D} quality_F}{N_D} \tag{8}$$

where $N_{\mathcal{D}}$ is the total number of RPFs derived by $\mathcal{D}$.

4.3. The decomposition algorithm

Based on the service relevance mining proposed in section 4.2, we design an algorithm to automatically decompose a business process into a hierarchy of RPFs

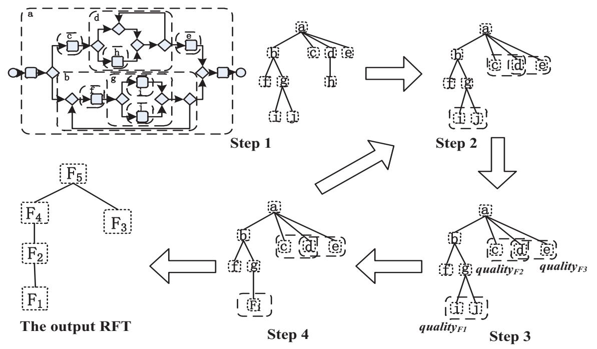


Figure 2. High-level steps of the decomposing algorithm

which we call the *Reusable Fragments Tree* (RFT). The algorithm iteratively analyzes a workflow graph to build new fragments. It has four high-level steps that are illustrated in Fig.2. In Step 1, the tree of all SESE fragments named as SFT is computed, using the linear-time algorithm presented in [11].

In Step 2, all connectable SESE fragment pairs are derived from SFT. This can be done by analyzing SFT bottom-up—all child fragments before a parent fragment. During the analysis, a minimum support p_r is chosen to filter the fragments that do not occur frequently. At each pass of this step, we firstly mark the leaf node fragments which have a $support_F$ score (s_F for short) greater than p_r, a $quality_F$ score (q_F for short) greater than q_r and at least two services in it as new RPFs. Then the solitary leaf node which has no brother node is removed from SFT and the leaf node set *LN* of SFT is analyzed. We check each node pair (F_1, F_2) in *LN* and if they obey the following conditions, they are defined as a connectable SESE fragment pair: (1) F_1 and F_2 have the same parent fragment; (2) F_1 and F_2 have a control flow in common which will be the connecting edge of them; (3) the support of $F_1 \cup F_2$ is greater than p_r.

In Step 3, for each connectable SESE fragment pair, we compute the q_F of the merged fragment of them and sort the scores in descending order.

The constructing of reusable process fragments is proceeded in a greedy fashion. In Step 4, the constructing is performed iteratively until a new reusable process fragment is formed. At each pass of this algorithm, the connectable SESE fragment pair with the highest ranked q_F is chosen and removed from the list. The two fragments are then merged to a new RPF if the q_F of the merged one is greater than the quality threshold q_r. After merging we replace the connectable SESE fragment pair with a tree node in SFT which represents the new RPF. If a new RPF has been found, the constructing algorithm is terminated and a reconstruction of the SFT is performed. The procedures from Step 2 to Step 4 of this algorithm are

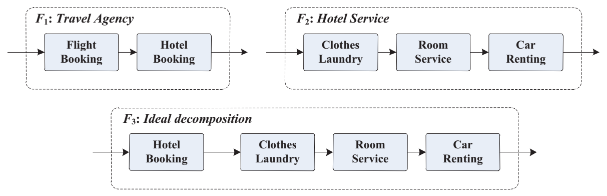


Figure 3. Problem of the greedy algorithm

iteratively performed until no more connectable SESE fragment pairs can be derived from SFT or no new reusable process fragment can be formed.

A greedy algorithm always pursues local optimal solutions at each step, but usually cannot obtain the global optimal solution. During the construction of reusable process fragments, an inappropriate construction decision at an early stage may prevent subsequent appropriate constructing. Consider the situation showed in Fig. 3. There is a reusable process fragment F_1 for *Travel Agency* TA which includes two sequential services: a flight booking service FB followed by a hotel booking service HB. This fragment is formed because of the frequent occurrences of TA in all existing composite service. But in the situation showed in Fig. 3, a new reusable process fragment F_2 for *Hotel Service* HS, which includes three sequential services: a clothes laundry service CL, a room service RS and a car renting service CR, has just been constructed. Now we need to decide whether or not to merge F_1 with F_2, where the service HB is closely associated with CL, RS and CR, but service FB is not because it also occurs often in other fragments such as *Airline Ticket Agent* and *Airline Company*, which typically do not co-occur with CL, RS or CR. Consequently, the two fragments F_1 and F_2 cannot be merged. And the ideal constructing result is the RPF F_3 including four sequential services: HB, CL, RS and CR. The solution to this problem is to split already-formed RPFs to obtain a better set of RPFs with higher q_F scores. Here we also use the quality threshold q_r to determine which RPF should be split and how the splitting and merging are done.

Alg. 1 shows the details of a single pass of the refined RPF constructing function in Step 4. Based on Step 3, all connectable SESE fragment pairs are sorted according to their q_F in descending order. At the start of this function, the connectable SESE fragment pair with the highest ranked q_F is chosen. The result of comparing q_F with q_r yields two branches of this algorithm. If q_F is greater than q_r, then the fragment pair is merged directly. Otherwise, we have to decide whether to split one of them or both of them to achieve a better constructing. Given a connectable SESE fragment pair (F_1, F_2), our algorithm makes splitting decision based on which of the following three cases occurs:

Algorithm 1. ConstructingRPF

Input: The sorted connectable SESE fragment pair list *sesesList*.
Output: The formed *RPF*. If no new *RPF* can be formed, return *null*.

```
begin
  while $sesesList \neq \emptyset$ do
    // Get the fragment pair with the highest $q_F$
    $(F_1, F_2)$ = HighestQualityPair(sesesList);
    // Remove this pair from the list
    sesesList.remove($(F_1, F_2)$);
    // Determine whether to split or not
    if $q_F((F_1, F_2)) > q_r$ then newRPF = $F_1 \cup F_2$;
    else if $F_1 = F_{11} \cup F_{12}$ and $F_2 = F_{21} \cup F_{22}$ then
      // Both of the two fragments are merged fragments
      if $q_F(F_{12} \cup F_2) > q_F(F_1)$ and
         $q_F(F_{12} \cup F_2) > q_F(F_2)$ and
         $q_F(F_{12} \cup F_2) > q_F(F_1 \cup F_{21})$ then
        $(F_{11}, F_{12})$ = SplitMergedFragment($F_1$);
        newRPF = $F_{12} \cup F_2$;
      else if $q_F(F_1 \cup F_{21}) > q_F(F_1)$ and
         $q_F(F_1 \cup F_{21}) > q_F(F_2)$ and
         $q_F(F_1 \cup F_{21}) > q_F(F_{12} \cup F_2)$ then
        $(F_{21}, F_{22})$ = SplitMergedFragment($F_2$);
        newRPF = $F_1 \cup F_{21}$;
    else if $F_1 = F_{11} \cup F_{12}$ then
      // $F_1$ is a merged fragment
      if $q_F(F_{12} \cup F_2) > q_F(F_1)$ and
         $q_F(F_{12} \cup F_2) > q_F(F_2)$ then
        $(F_{11}, F_{12})$ = SplitMergedFragment($F_1$);
        newRPF = $F_{12} \cup F_2$;
    else if $F_2 = F_{21} \cup F_{22}$ then
      // $F_2$ is a merged fragment
      if $q_F(F_1 \cup F_{21}) > q_F(F_1)$ and
         $q_F(F_1 \cup F_{21}) > q_F(F_2)$ then
        $(F_{21}, F_{22})$ = SplitMergedFragment($F_2$);
        newRPF = $F_1 \cup F_{21}$;
    // If a new fragment is formed, terminate the iterating
    if newRPF $\neq$ null then break;
  return newRPF;
end
```

1. If F_1 and F_2 are both original SESE fragments that are not the merging of other fragments, then do nothing.
2. If only one of the two fragments, for example F_1, is the merging of a connectable SESE fragment pair (F_{11}, F_{12}), then there are two options: one is to split F_1 into F_{11} and F_{12} and then merge F_{12} with F_2; the other is not to split or merge. We compute the q_F for F_1, F_2 and $F_{12} \cup F_2$, and choose the first option if $quality_{F_{12} \cup F_2}$ is the higher one.
3. If both of the two fragments are merged fragments. That is, F_1 is the merging of (F_{11}, F_{12}) and F_2 is the merging of (F_{21}, F_{22}). Then there are three options: one is to split F_1 into F_{11} and F_{12} and then merge F_{12} with F_2; another one is to split F_2 into F_{21} and F_{22} and then merge F_1 with F_{21}; the last one is not to split or merge. We compute the q_F for F_1, F_2, $F_{12} \cup F_2$ and $F_1 \cup F_{21}$, and choose the option with the highest quality score.

5. Experiments and evaluation

We have implemented a prototype system to evaluate the proposed approach and algorithms. We first evaluate the performance of building SCN with different total number of composite services. Then the performance of decomposing workflow graph into RFT with different quality threshold is evaluated. Finally, we give some empirical properties of the reusable bioinformatics process fragment repository.

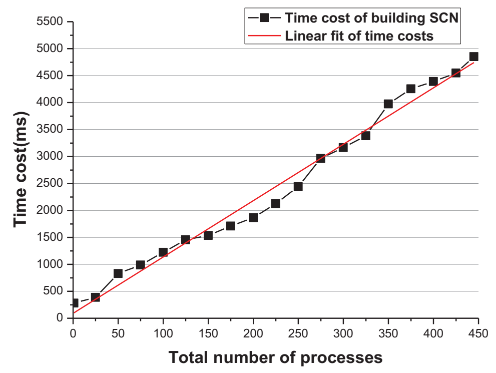
The data set used in the experiments is a workflow repository that contains 445 bioinformatics workflows taken from myexperiment[1], which is a collaborative environment where scientists can publish and share their workflows and experiment plans [12]. These workflows contain 2655 web services and the average occurrence of them is 2.58. According to this context, we say that a fragment which has a set of services with more than 5 times of occurrence is a frequent fragment. Thus the value of minimum support threshold p_r is set to be 0.01 in our experiments. The experiments were conducted on a Windows machine with two 3GHz Intel Xeon CPUs and 4G main memory.

Firstly, we evaluate efficiency of building SCN with different total number of business processes. We use CPU time as the standard measure of the performance. The time costs of the building procedure are tested with the increase of the number of bioinformatics workflows and the results are showed in Fig. 4(a). We can see that the building time increases in a linear way with respect to the number of workflows. For a large workflow repository, there may be a huge number of workflows and it will take a long time to build SCN. But as a preparation work of process decomposition, the building of SCN can be done beforehand and an incremental approach will make it more efficient.

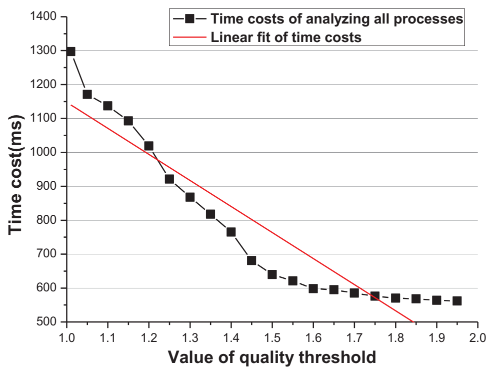
The second evaluation case was to evaluate the workflow graph decomposing time in the light of increasing the value of quality threshold. We performed tests of decomposing all the 445 bioinformatics workflows with quality threshold values from 1.01 to 1.95 by increasing 0.01 each time. The results are showed in Fig. 4(b). We can also see that the decomposing algorithm is linear with respect to the value of quality threshold. And at the worst situation in the evaluating scenario, the time of decomposing 445 workflows is 1.31 seconds. The proposed process decomposing algorithm is effective and practical.

Finally we present some empirical properties of the reusable bioinformatics process fragment repository generated by our algorithm. The quality threshold q_r is set to be 1.35 in this case. Fig. 5 presents the

[1] www.myexperiment.org



(a) Time costs of building SCN



(b) Time costs of process decomposition

Figure 4. Performance evaluation

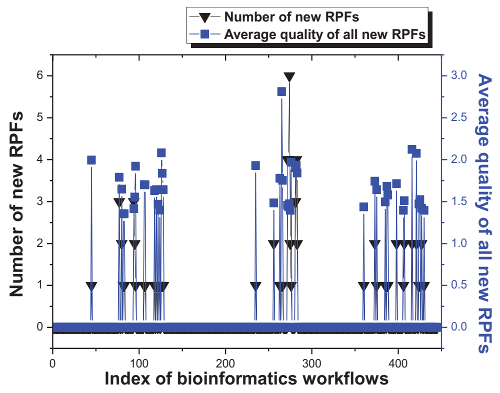


Figure 5. Case study and empirical analysis

distribution of RPF numbers formed in each bioinformatics workflow and the distribution of average quality score of them. In the 445 bioinformatics workflows there are 76 reusable process fragments. The maximum number of all workflows is 6, and the average number is 0.17. The maximum quality score of the 76 reusable process fragments is 2.81 and the average quality score is 1.68. It can be seen that our process decomposing algorithm can produce a number of high quality reusable process fragments.

6. Related work

Several research approaches have considered decomposing existing software system into reusable software components to reduce the duplicate work in software development. These approaches can be classified into three main categories:

(1) *Knowledge-matching based approaches*. These approaches are based on the domain knowledge obtained by doing domain analysis in some business domains. The software components are checked with the domain knowledge and the matched ones are extracted into reusable software components. The component mining process [13], the pattern identifying approach for recovering software architecture [14] and the feature-oriented reuse method [15] are of this type. However, these approaches rarely have the ability to obtain reusable software components from business models automatically because the domain analysis is not an automatic work and should be accomplished with the aid of experiences of domain analyzers.

(2) *Cohesion-coupling based clustering analysis approaches*. In these approaches, researchers try to cluster business models according to "high cohesion and low coupling" principle and encapsulate each cluster into a component [16]. The graph clustering method for software components capture [17] and the spectral methods for software clustering [18] are of this type and [19] gives a summary of these approaches. Compared with these approaches that target at the decomposition of procedural or object-oriented program and are not fit well with the autonomous nature of services [2], our process decomposing algorithm is much more focused on the relevance of web services in a composite service which is a loose-couple, collaborative relation between web services and is not consider by previous approaches.

(3) *Business process parsing approaches*. These approaches decompose a business process into sub-processes according to some structural constrains such as well-structure, execution semantics preservation and so on. The building of the program structure tree [7,11] and the refined process structure tree [20] are of this type. The first step of our process decomposing algorithm is based on the work in [10].

In summary, current software system decomposition methodologies do not bridge the gap between software analysts and developers effectively as few approaches take into account both the structural constrains of business process and domain relation between web services in process-aware service composition. The presented work combines the structural constrains and service relevance as the criteria of decomposing a business process and provides an effective and practical algorithm.

7. Conclusions and future work

In this paper, we have presented a novel business process decomposition mechanism for discovering and building reusable process fragments automatically, in a bottom-up fashion. We have introduced criteria for the building of fragments and presented an algorithm for decomposing a business process into a hierarchy of reusable process fragments, which catches not only structural constraints, but also service relevance information of the business process. Experimental results show that the proposed business process decomposition approach is effective and practical.

As future work, we are interested in improving the description of a reusable process fragment and providing an automatic query method for it, since currently the query and discovery of those fragments are done almost manually.

Acknowledgments. We would like to thank Jue Wang of University Of Washington in Seattle for his fruitful suggestions and kind help in the writing of this paper. We would also like to thank the anonymous reviewers for their constructive comments and suggestions. This research was supported in part by China 863 High-tech R&D Program (No.2007AA010301 and No.2009AA01Z419) and China 973 Fundamental R&D Program (No.2005CB321803).

References

[1] M.P., Papazoglou, W-J., van den Heuvel: Service-Oriented Architectures: Approaches, Technologies and Research Issues. In: VLDB Journal, vol. 16, pp. 389--415 (2007)

[2] W.M.P., van der Aalst, M., Pesic: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Proceedings of International Conference on Web Services and Formal Methods (WS-FM), LNCS, vol. 4184, pp. 1--23. Springer-Verlag, Berlin (2006)

[3] B., Benatallah, Q., Sheng, M., Dumas: The Self-Serv Environment for Web Services Composition. In: IEEE Internet Computing, January / February, pp. 40--48 (2003)

[4] G., Zheng, A., Bouguettaya: Service Mining on the Web. In: IEEE Transactions on Services Computing, vol. 2, pp. 65--78 (2009)

[5] ObjectManagementGroup. Business Process Modeling Notation, V1.1. OMG Available Specification, January (2008)

[6] W.M.P., van der Aalst, A., Hirnschall, H.M.W., Verbeek: An alternative way to analyze workflow graphs. In: Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE), LNCS, vol. 2348, pp. 535--552. Springer (2002)

[7] J., Vanhatalo, H., Völzer, F., Leymann: Faster and More Focused Control-flow Analysis for Business Process Models though SESE Decomposition. In: Proceedings of the 5th International Conference on Service-Oriented Computing (ICSOC), LNCS, vol. 4749, pp. 43--55. Springer, Heidelberg (2007)

[8] X., Dong, A., Halevy, J., Madhavan, E., Nemes, J., Zhang: Similarity Search for Web Services. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB), vol. 30, pp. 372--383 (2004)

[9] D., Hand, H., Mannila, P., Smyth: Principles of Data Mining. The MIT Press, Cambridge, MA, USA (2001)

[10] B.S., Mitchell, S., Mancoridis: Comparing the Decompositions Produced by Software Clustering Algorithms using Similarity Measurements. In: Proceedings of IEEE International Conference on Software Maintenance (ICSM), pp. 744--753. IEEE Computer Society Press, Florence (2001)

[11] R., Johnson, D., Pearson, K., Pingali: The Program Structure Tree: Computing Control Regions in Linear Time. In: Proceedings of the ACM SIGPLAN'94 Conference on Programming Language Design and Implementation (PLDI), pp. 171--185 (1994)

[12] D., Roure, D., Goble, C., Bhagat, J., Cruickshank, D., Goderis, A., Michaelides, D., Newman, D.,: myExperiment: Defining the Social Virtual Research Environment. In: Proceedings of 4th IEEE International Conference on e-Science, pp. 182--189. Indianapolis, Indiana, USA (2008)

[13] D., Spinellis, K., Raptis: Component Mining: A Process and Its Pattern Language. In: Information and Software Technology, vol. 42(9), pp. 609--617 (2000)

[14] M., Pinzger, H., Gall: Pattern-supported Architecture Recovery. In: Proceedings of the 10th International Workshop on Program Comprehension, pp. 53--61. IEEE Computer Society Press, Paris (2002)

[15] K.C., Kang, S., Kim, J., Lee, K., Kim, E., Shin, M., Huh: FORM: A Feature-oriented Reuse Method with Domain-specific Reference Architectures. In: Annals of Software Engineering, vol.5, pp.143--168 (1998)

[16] J.K., Lee, S.J., Jung, S.D., Kim: Component Identification Method with Coupling and Cohesion. In: Proceedings of 8th Asia–Pacific Software Engineering Conference, pp.79--86. Macau, China (2001)

[17] Y., Chiricota, F., Jourdan, G., Melancon: Software Components Capture using Graph Clustering. In: Proceedings of the International Workshop on Program Comprehension (IWPC), pp. 217--226 (2003)

[18] A., Shokoufandeh, S., Mancoridis, M., Maycock: Applying Spectral Methods to Software Clustering. In: Proceedings of the Working Conference on Reverse Engineering (WCRE), pp. 3--10 (2002)

[19] K., Rainer: Atomic Architectural Component Recovery for Program Understanding and Evolution. Ph.D. dissertation, Institut für Informatik, Universität Stuttgart (2000)

[20] J., Vanhatalo, H., Völzer, J., Koehler: The Refined Process Structure Tree. In: Proceedings of the 6th International Conference on Business Process Management, LNCS, vol. 5240, pp. 100--115. Springer, Heidelberg (2008)