Formal Identification of Right-Grained Services for Service-Oriented Modeling*

Yukyong Kim and Kyung-Goo Doh**

Dept. of Computer Science & Engineering, Hanyang University, Ansan, 426791, South Korea
{yukyong,doh}@hanyang.ac.kr

Abstract. Identifying the right-grained services is important to lead the successful service orientation because it has a direct impact on two major goals: the composability of loosely-coupled services, and the reusability of individual services in different contexts. Although the concept of service orientation has been intensively debated in recent years, a unified methodic approach for identifying services has not yet been reached. In this paper, we suggest a formal approach to identify services at the right level of granularity from the business process model. Our approach uses the concept of graph clustering and provides a systematical approach by defining the cost metric as a measure of the interaction costs. To effectively extract service information from the business model, we take activities as the smallest units in service identification and cluster activities with high interaction cost into a task through hierarchical clustering algorithm, so as to reduce the coupling of remote tasks and to increase local task cohesion.

Keywords: Service identification, service granularity, business process model, UML activity diagram, graph clustering.

1 Introduction

The basic idea of Service-Oriented Architecture (SOA) is the restructuring of the information technology (IT) systems or IT landscapes into loosely coupled, independent services. These services should allow the reuse of existing IT functionality in order to shorten the time between design and implementation when business requirements change [1]. The key challenges in developing the service-oriented systems are the refinement and eventually the mapping of business processes to the existing service infrastructure. However, the existing services do not immediately fit the requirements elicited during business process modeling. We consider this problem as the identification of reusable services at the right level of granularity that can bring on a mismatch between the business process models and the

* This work is supported by the Engineering Research Center of Excellence Program of Korea Ministry of Education, Science and Technology (MEST)/Korea Science and Engineering Foundation (KOSEF), grant number R11-2008-007-01003-0.
** Corresponding author.

G. Vossen, D.D.E. Long, and J.X. Yu (Eds.): WISE 2009, LNCS 5802, pp. 261–273, 2009.

available services. In this paper, we focus on how the right level of service abstraction and granularity could be provided based on business process models specifying the business requirements.

A business process is a collection of business functions required to achieve its ultimate business goals or objectives. In practice, the business functions are connected with each other to represent the execution sequence or execution pattern with various control constructors, such as serial, parallel, alternative, and so on, which are used in modeling a business process. A business function, also called a business activity, has its corresponding participant which completely supports its task and is ultimately materialized as a service. Thus a service is essentially a business component, which implements an autonomous business concept or business process.

Note that the coverage of a service depends on the system implementation policy. That is, a single service can support the role of a single business activity, several business activities or even a whole business process. This is the problem of service granularity. Because a business process is composed of a set of business activities realized by a set of services, service granularity is related with determining both the number of services required to fulfill a business process and the coverage of a service. Hence, identifying the right-grained services is to discover and determine needed services to achieve the business processes designed from the business requirements. The problem of identifying appropriate services has not been addressed in the literature. No formal methodology and tools that allow the designer to generate and evaluate alternative designs based on a set of managerial design goals exist [2]. This paper describes a formal approach to identify services of right granularity.

Our goal is to determine the service coverage to fit the business processes and then expose a reasonable number of services. We solve the problem of determining the right coverage of service using the cost metric of interactions between business activities. Suggested cost metric is to evaluate amount of interactions and to determine which activities are covered by a service. Several researches have suggested as the service modeling approach that can identify and specify service components [3–5]. However, because they provide only descriptive guidelines to define services, it is less obvious and objective to apply those approaches, and then it much relies on experience and intuition. The major contribution of our work is a formal and systematical approach using metrics which are expected to be clear, objective and efficient for identifying services of the right abstraction level. This article is organized as follows: First, we briefly mention the related work in Section 2. Section 3 describes terms and basic principles used in our paper. Section 4 presents our service identification method along with a running example. After analyzing and discussing the results of validation in Section 5, we summarize and conclude the paper including some future works in Section 6.

2 Related Works

The development and integration of services resemble Component Based Software Development (CBSD), where pre-built parts, known as business components, are assembled into larger scale applications [6, 7]. In 2005, T. Erl introduced a service modeling process to produce service and operation candidates. This process provides

steps and guidelines for the modeling of a SOA consisting of application, business, and orchestration service layers [3]. IBM introduced Service-Oriented Modeling and Architecture (SOMA) [4]. SOMA illustrates the activities for service modeling and the importance of activities from service consumer and provider perspectives. It is an enhanced, interdisciplinary service modeling approach that extends existing development processes and notations such as object-oriented analysis and design, Enterprise Architecture (EA) frameworks, and Business Process Modeling (BPM). In [5], Web Services Modeling Framework (WSMF) was defined to provide a rich conceptual model for the development and the description of Web services. The philosophy of WSMF is based on the maximal de-coupling principle. A model in WSMF consists of four main elements: ontology that provide the terminology used by other elements; goal repositories that define the problems that should be solved by web services; web services descriptions that define various aspects of a web service; and mediators which bypass interoperability problems.

In [2], H. Jain et al. described a formal approach to web services identification, which takes an analysis level object model, representing a business domain, as input and generates potential web service designs, in which the classes in the object model are grouped into appropriate web services based on static and dynamic relationships between classes. An initial hierarchical grouping of classes is derived using a maximum spanning tree algorithm. In [8], M. Bell suggested the conceptual service identification process based on six best practices that can assist with identifying organizational concepts, establishing conceptual services, founding service associations, and forming service structures. Those six best practices are concept attribution, concept classification, concept association, concept aggregation, concept generalization, and concept specification. To discover abstractions and derive conceptual services, he used the decision tree. The attribution analysis process yields sets of recommended attributes that are essential inputs into most categorization activities. Conceptual services are derived from those attributes by applying their corresponding business rules.

These approaches have key limitations: One is that they lack details about how to identify and define services from the business domain. Even though the decision tree is used in [8], they provide only descriptive guidelines instead of using a formal approach to modeling services. The other is that they are based on object models or component models. To remedy these limitations, they usually employ classes and component grouping techniques to identify services. To resolve these limitations in this paper, we formalize the service identification problem on the graph, and directly derive services from the business requirements rather than object or component models.

3 Preliminaries

As shown in Figure 1, the service model in between business model and implementation model is a key factor that can help achieve the service orientation. The service model provides a logical place to define the contracts that ensure that the business side of the organization is aligned with the IT side from a requirement

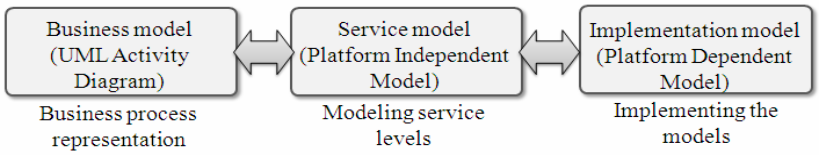


Fig. 1. Three part model of service orientation

perspective [9]. Service models play an important role during service-oriented analysis and service-oriented design phases. We aim at identifying services from the business model at the proper abstraction level. The identified services are the base for building a service model written in WSDL.

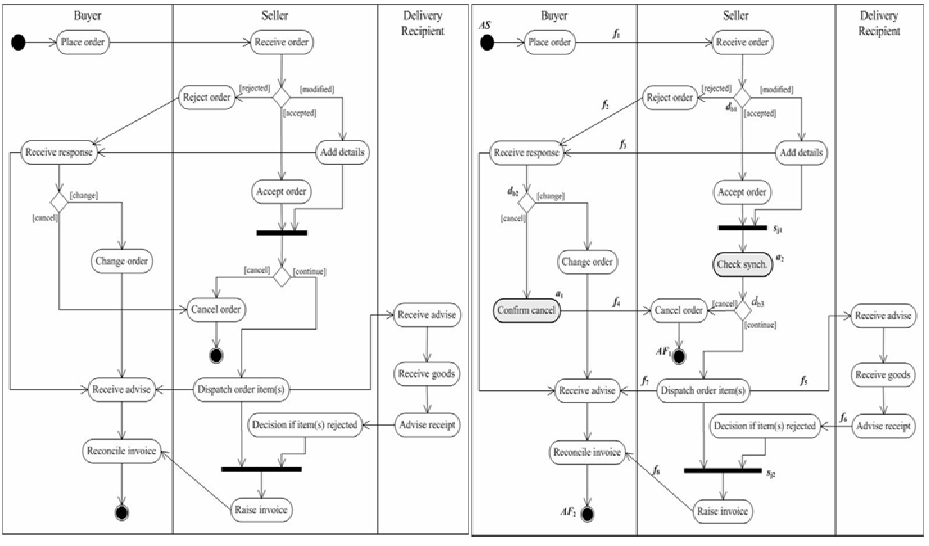
Once the business process is defined and modeled, and the business functionality is understood, then the service identification step starts. As previously stated, a business process is a sequence of activities that are ordered according to a set of procedural rules. Activity diagrams of UML (Unified Modeling Language) have been widely adopted into the business process modeling [10]. Based on activity diagrams, the modeling for communication specification and process logic of inter-organizational processes can be unified to the same modeling language. An activity diagram is used to define the execution logic of business process. Difference roles in business process and their communication structures can be defined based on the swim-lane notation.

Definition 1. An activity diagram $\boldsymbol{G}$ is a tuple (A, D, S, R, F, AS, AF, SL), where

1. A is a finite set of activity nodes.
2. D is a finite set of decision nodes. D_b is the set of branch nodes and D_m is the set of merge nodes. D_b and D_m is a partition of D, i.e., $D_b \cap D_m = \varnothing$ and $D_b \cup D_m = D$.
3. S is a finite set of synchronization nodes. S_f is the set of fork nodes and S_j is the set of join nodes. S_f and S_j is a partition of S, i.e., $S_f \cap S_j = \varnothing$ and $S_f \cup S_j = S$.
4. A_S is the set of start nodes of $\boldsymbol{G}$ and A_F is the set of end nodes of $\boldsymbol{G}$.
5. R is the set of business entities or organizational units of $\boldsymbol{G}$.
6. SL: $A \cup D \cup S \rightarrow R$ is the mapping function from diagram elements to business entities or organizational units.
7. $F \subseteq (A_S \times A) \cup ((A \cup D \cup S) \times (A \cup D \cup S)) \cup (A \times A_F)$ is a set of edges. F is partitioned into two subsets F_L and F_R. F_L is the set of intra-flow edges connecting nodes within the same business entity, i.e., $F_L = \{(v_1, v_2) \mid (v_1, v_2) \in F \wedge SL(v_1) = SL(v_2)\}$. F_R is the set of inter-flow edges interconnecting different business entities, i.e., $F_R = \{(v_1, v_2) \mid (v_1, v_2) \in F \wedge SL(v_1) \neq SL(v_2)\}$.

Activity diagrams for business process must be prescribed under the following three constraints:

C1. There are only one start node denoted by a solid circle and one or more end nodes shown by bull's eye symbols in an activity diagram. The start node has only one post-activity node and the end node has one pre-activity node.



(a) An original activity diagram (b) An activity diagram for business process

Fig. 2. Example: activity diagram for business process

C2. Every node connected to inter-flow edges must be either an activity node or a synchronization node. For example, the activity a_1 in the Buyer swim-lane in Figure 2 (b) is a newly introduced activity node to fulfill this constraint.

C3. No edge is allowed between non-activity nodes. For example, the activity a_2 in the Seller swim-lane in Figure 2 (b) is introduced to satisfy this constraint.

The execution of business process is modeled by the behaviors of its activity diagrams. The behaviors are described in terms of the set of all execution paths from start node to end node.

Definition 2. Let $\boldsymbol{G} = (A, D, S, R, F, A_S, A_F, SL)$ be an activity diagram for business process. An *execution path* in G is the sequence of a start node, followed by one or more nodes, and followed by an end node. $a_i \leq_p a_k$ denotes that an activity a_i occurs before an activity a_k in a path p.

For example, Figure 3 shows a path, $<AS, a_1, a_2, d_{b1}, a_5, S_{j1}, a_{10}, d_{b3}, a_{11}, a_{12}, a_{13}, a_{17}, S_{j2}, a_{18}, a_{16}, AF_2>$ is an execution path in Figure 2.

Generally, a *task* and an *activity* can be treated with no difference in the business process. In this paper, however, we discriminate a task from an activity. To identify a service, we may attempt to map an activity to a service because service orientation treats the activities performed by the business process as services. According to the concept of SOA, however, a service must describe a business workflow. Because an activity is the smallest unit of work that makes sense to a user, it might not describe a complete business flow. Thus we might have to consider the case of a service consisting of several activities. To express the related activities in activity diagrams, we need a logical unit that eventually corresponds to a service.

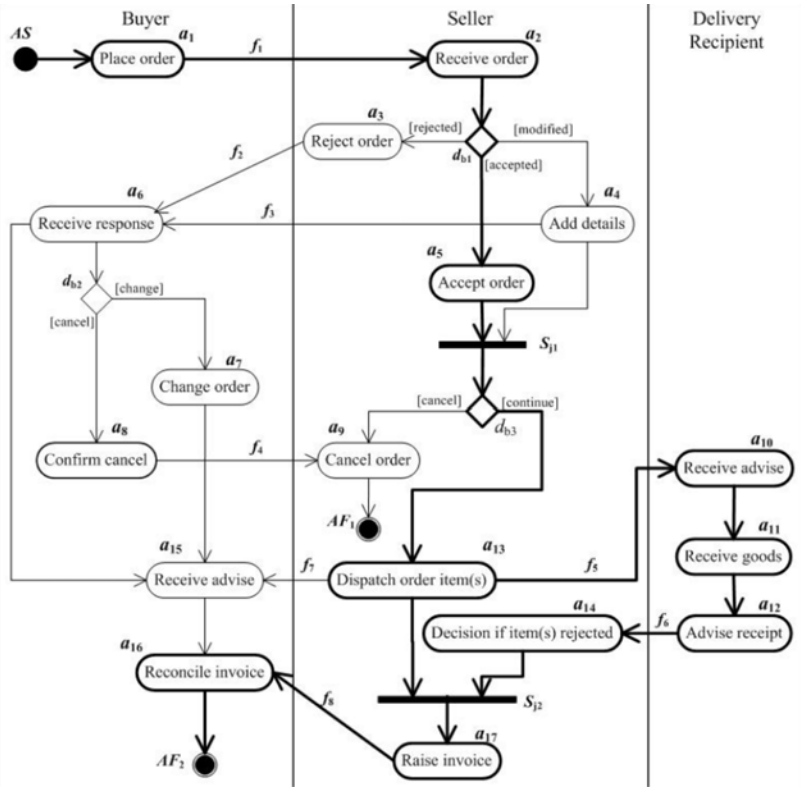


Fig. 3. A path p1 of Fig. 2–(b)

Definition 3. For $\boldsymbol{G}$ = (A, D, S, R, F, A_S, A_F, SL), an *activity* is the smallest unit of work in the business process and is represented by a node in $A \cup D \cup S$. Especially, every activity node in A is given a weight which is the number of paths the activity state belongs to.

Definition 4. For $\boldsymbol{G}$ = (A, D, S, R, F, A_S, A_F, SL), a *task* is a logical group of related activities in $A \cup D \cup S$, being clustered together in $\boldsymbol{G}$.

A service includes a sequence of business workflow, and services are executed across a network. However, operations internal to the service communicate directly with each other within a single machine, not across a network connection. Thus a service cannot be located across several roles. From these points of view, defining a task as to minimize the amount of interactions between activities has an effect on identifying a service of the right granularity. Thus we establish the task clustering principle as follows: for every activity in $A \cup S \cup D$ in $\boldsymbol{G}$, activities are clustered into a task where they are all connected with intra-flow edges.

A service itself can be coarse-grained or fine-grained. The level of granularity for services tends to be coarser than the level of granularity for objects or components. A service typically exposes a single, discrete business process [11]. Coarse-grained services require less network roundtrips as the execution state is contained in the message. Consequently service granularity refers to how much functionality the service covers. In our view, defining right-grained services is determining a task so as to include an adequate number of activities. There are three possible ways for determining a task:

1. *One-to-One*: One activity could be directly mapped to only one task.
2. *All-to-One*: Activities in a role becomes a cluster in its entirety.
3. *Many-to-Many*: Activities in the same role are decomposed into several clusters, each of which could be mapped to a task.

In *One-to-One* mapping, each activity would only model a single task. For example, we can recognize 17 tasks from 17 activity nodes in Figure 2-(b). However, as previously mentioned, a task may have to comprise a sequence of activities. Moreover, even if an activity can conceptually be an atomic service fully describing a business workflow, too many fine-grained services may lead to heavy traffic between the service providers and consumers. In *All-to-One* mapping, the number of tasks is equivalent to the number of roles. In the example of Figure 2-(b), we can identify 3 tasks from 3 roles. In this case, services are coarse-grained, giving benefits of decreasing network overhead. However, services may not be flexible enough so that reuse may be very hard or not even be possible at all. It is easy to see that the smaller the size of a service is, the higher the chance for its reuse becomes. Hence, it is desirable to decompose tasks properly in order to maximize the reusability and the composability of service components. Therefore, *Many-to-Many* mapping is the best candidate that provides the opportunities to decompose tasks in proper granularity. Because the number of services directly affects the performance and network overheads, we present in the next section the mechanical way of decomposing tasks based on a cost metric that assesses the amount of interactions between tasks.

4 Service Identification

In this section, we present how to identify tasks on $\boldsymbol{G}$ to define services at the right level of abstraction. We formalize the identification problem and define the cost metric.

4.1 Problem Definition

The approach we propose for identifying services uses an activity diagram $\boldsymbol{G}$ as input. Because services are defined from tasks in our method, we mainly focus on the approach of task identification on $\boldsymbol{G}$. We formalize the task identification as the activity allocation problem on the graph under the task clustering principle of the previous section. We consider $\boldsymbol{G}$ as a weighted graph in which each branch and node is given a numerical weight by Definition 5.

Definition 5. Given an activity diagram $\boldsymbol{G}$ = (A, D, S, R, F, A_S, A_F, SL), each activity node a in A has a weight value $freq(a)$ representing how many paths a belongs to. Each arc (a_i, a_j) in F also has a weight value $cost(a_i, a_j)$ which is the interaction cost overhead between nodes a_i and a_j. The interaction cost is usually determined by the system environment.

Our goal is to cluster activities and determine a task to minimize the cost of interactions. Then the task identification can be concretely seen as the *minimal activity distribution* (MAD) problem on a weighted directed graph $\boldsymbol{G}$ that represents business processes. The MAD problem is to allocate activities to tasks such that the total cost of remote interactions is minimized. The followings are assumptions made for the MAD problem.

Assumption 1 For each activity, the cost of executing its own set of operations is constant regardless of the number of operations.
Assumption 2 The remote interactions cost more than the local ones.

Definition 6. For $\boldsymbol{G}$, each activity or task a_i has a weight $freq(a_i)$ as an initial value. Each edge $e = (a_i, a_j) \in F$ has a weight $cost(a_i, a_j)$ as an initial value. Then the interaction cost of a pair of tasks a_i and a_j, $cost(a_i, a_j)$, is calculated iteratively as follows:

$$cost(a_i, a_j) = \min\left(freq(a_i), freq(a_j)\right) * cost(a_i, a_j)$$

Here, we define the total interaction cost $Tcost(G)$ of $\boldsymbol{G}$ is the sum of cost of all tasks.

$$Tcost(G) = \sum_{i,j=1,i\neq j}^{n} \left(cost(a_i, a_j)\right)$$

4.2 Service Identification

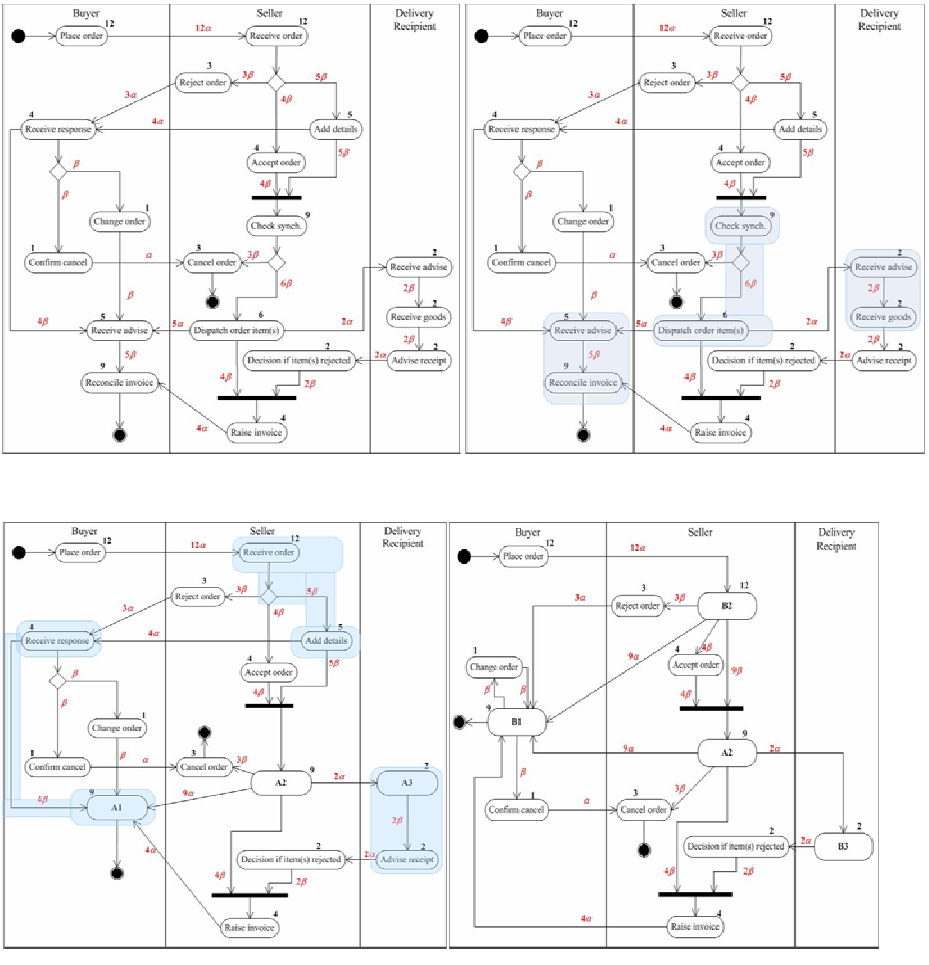
The problem of service identification from an activity diagram of business workflow is formalized as the MAD problem determining tasks in the activity diagram in Section 4.1. In this subsection, we present the way to identify tasks at the right level of abstraction. The identification process consists of two steps: activity clustering and activity allocation. Activities are clustered in such a way that the total cost of communications is minimized. Each cluster is then mapped to a task. The next step is to organize tasks into the initial set of services by allocating activities to the corresponding task and defining operations for each task. Then each task becomes a service that can be described in Web Services Description Language (WSDL).

(1) Clustering activities into tasks

In graph clustering, when an edge e is joined to v_i and v_j, and two vertices v_i and v_j are merged into a new cluster v_k, every edges incident to v_i or v_j except e is incident to the vertex v_k. Activity clustering is similar to the graph clustering and starts by placing each activity in a task by its own, creating $|A|$ tasks in $\boldsymbol{G}$. Tasks are then repeatedly combined in such a way that the total cost of communications is minimized. The following shows how to calculate the weight for a new task:

Rule 1. When adjacent two tasks are merged, one of their incident edges having the largest weight is selected first.
Rule 2. Once an edge is selected, the edge cannot be selected again in each repetition.
Rule 3. When tasks are merged, a new edge is created and its weight is calculated.
Rule 4. When adjacent two tasks are combined into a new task, the weight of the new task is the maximum weights of two tasks.

Based on rules *Rule 1* and *Rule 2*, we determine which tasks are merged. The graph ***G*** is reconstructed by adjusting edges on *Rule 3* and assigning the weight newly to each task on *Rule 4*. The following Figure 4 shows activity clustering steps for Figure 2-(b) using above rules.



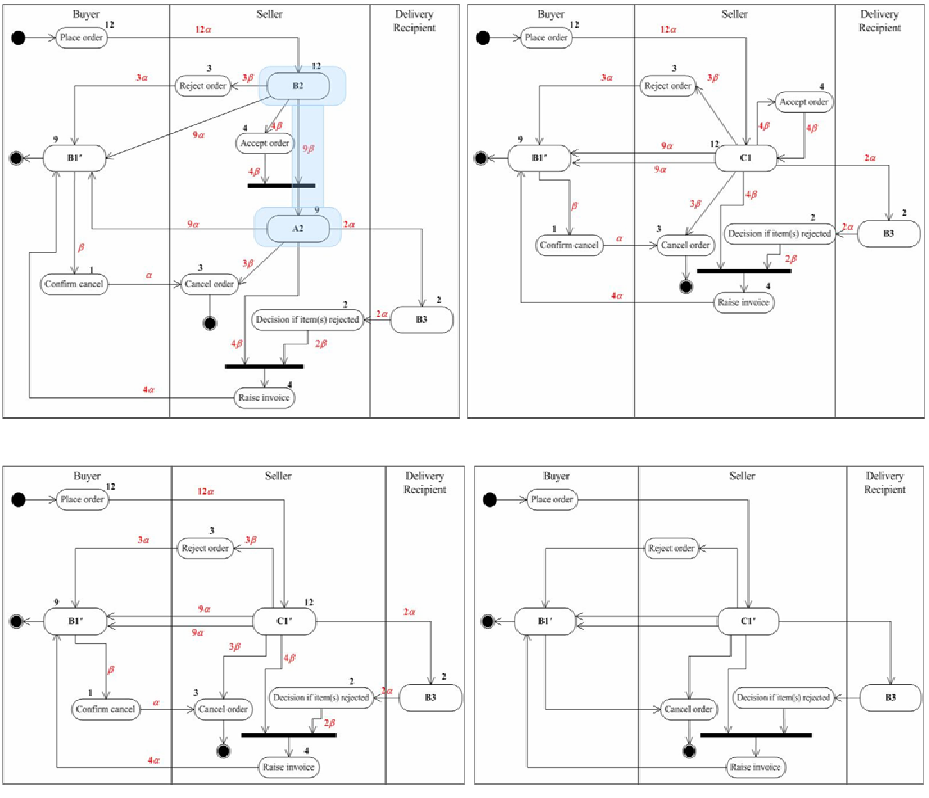


Fig. 4. Clustering steps for Fig. 2-(b)

(2) Organizing tasks into initial set of services

The ideal scenario is one in which the cohesion within a service is maximized and the coupling between services is minimized. Finding a service design that maximizes intra-service cohesion and minimizes inter-service coupling is formalized as the MAD problem in this paper. Now we organize identified tasks into services to derive the service specification. Graph clustering provides a good way for grouping the vertices in a graph according to their connections. We then generate a structure of the tasks. Activities in a task are suggested to be included in a service. This method provides potential designs with different levels of inter-service coupling where activities are organized into services, and smaller services are successively integrated into bigger ones. This leads the initial solution based on the preferred size of services and reasonable number of services desired for the business domain. The initial set can be further refined based on heuristics.

(3) Service description

For identified services, a set of attributes to describe and document the capability of each service is defined. Some of the key attributes include: Who owns it? Who is its

customer? What are the inputs and outputs? This rich description of the capabilities can be passed to development teams who can use the information to help select the appropriate implementation technologies, hosts, and deployment topologies. The service description is easily derived from tasks and activity descriptions provided by the business model. A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: *messages*, which are abstract descriptions of the data being exchanged, and *port types* which are abstract collections of operations [12]. We can gain <*data type*> from activities and their interaction scheme and <*port type*> from operations included in the activity.

5 Evaluation

Comparison of approaches: A variety of heterogeneous approaches have been proposed as shown in Table 2. Approaches especially vary in terms of service hierarchies and analysis objectives. Thus, methods are proposed to identify services by utilizing the information systems in place in a bottom-up approach or follow a

Table 1. Comparison of approaches regarding service identification

Approach (Year) **Criteria**	**Jain et al. (2004)**	**T. Erl (2006)**	**M. Bell (2008)**	**Proposed approach (2009)**
Background & starting point	Bottom-up. Analysis of current IS and their functionality	Meet-in-the-middle. Parallel analysis of business processes and IS	Bottom-up. Analysis of current IS	Top-down. Business processes are decomposed
Service classification scheme	Implicit distinction of elemental and composed services	11 service types are proposed (partially orthogonal)	6 categories for conceptual services	3 service hierarchies: service, task, activity
Covering of SOA design phases	Focus on service development, service identification is a minor stage	All SOA phases	Service categorization and specification	Service identification and specification
Characteristics	7 phases, short documentation	Exhaustive documentation, case studies, guidelines	Decision tree	Formal criteria, WSDL description
Application of process models for service identification	Functional areas only to enable code reengineering	Initially decomposed into service candidates	–	Process model divided into service candidates

procedure of analyzing business requirements in a top-down approach. Other approaches integrate both perspectives into a hybrid strategy, referred to as 'meet-in-the-middle' approach. Compared to those approaches, the procedure presented in this paper introduces a strong business perspective into the derivation of service candidates. This is done by integrating business partners as important participants when deriving services from business process models.

Analysis of the clustering algorithm: We first analyze the time complexity of clustering algorithm. Suppose n activities are obtained from the business model, then there are n^2 pairs of activities or cluster candidates. Select the pair with greatest cost from n^2 pair of activities, with time complexity of $\boldsymbol{O}(n^2)$. The time complexity processing all possible activity pairs is $\boldsymbol{O}(n^2)$ and suppose the average time complexity to compute the interaction cost of a activity pair is $\boldsymbol{O}(m)$, then the time complexity for the whole process is approximately $\boldsymbol{O}(\mathrm{m}*n^2)$. To compute cost between two activities, suppose the number of activities in every tasks is T/n and suppose every activity contains L operations in average and then the time complexity to compute *cost*() between tasks is $2*(T/n)*L$, i.e. $\boldsymbol{O}(L*T/n)$. Therefore, the number n of activities has great influence on the clustering algorithm.

Case study: The results of this work are currently being validated at a simplified version of the actual procedure that deals with requests for student grants in a LMS (Learning Management System). To have control over granularity is one of the major concerns in their migration to SOA. The validation of this work is that the presented identification process is adopted by the LMS. This means that the impact of each service under development is verified with respect to each type of granularity, *One-to-One*, *All-to-One*, and *Many-to-Many*. For each type of the granularity, we evaluate the total interaction cost between activities. Although the simplified use on the LMS, *Many-to-Many* mapping is appropriate in the given context.

6 Conclusion and Future Work

Although the concept of SOA has been intensively debated in recent years, a unified methodical approach for identifying services has not yet been reached. Service granularity, the scope of functionality that is exposed by a service, is a crucial issue in designing SOA. Every SOA needs to have well designed services in order to gain the predicted benefits, such as flexible business processes and low development costs. However, while many literature sources suggest architects to choose the right level of granularity of services, none of these studies goes into detail about how to do this.

In this paper, we attempted to discuss service granularity. Although the importance of coarse-grained services is stated, the enterprise architecture nowadays has to deal with a broad spectrum of possible service granularity levels. From this perspective, we defined a formal method of right-grained service identification based on the business model, using the graph clustering technique. To effectively extract service information from the business model, we proposed an activity clustering approach based on cost metrics. This approach takes activities as the smallest units in service identification, takes the cost metric as a measure of connectivity between activities,

and clusters activities with high interaction cost into a task through hierarchical clustering algorithm, so as to reduce the coupling of remote tasks and to increase local task cohesion.

This approach still needs further improvement. Such as, it does not take into account the concept of service category which is based on the nature of the logic they encapsulate and the manner in which they are typically utilized within SOA. Moreover, certain thresholds in the identification process shall be given by experienced engineers to have expected effect which may be in the selection of center points during the process of clustering is not ideal. As part of further research, we will propose concrete metrics and rules to determine which granularity levels are appropriate in a particular context. Now we are developing a tool for service design based on business modeling.

References

1. Adam, S., Doerr, J.: How to better align BPM & SOA. LNCS, vol. 5074, pp. 49–55. Springer, Heidelberg (2008)
2. Jain, H., Zhao, H., Chinta, N.: A Spanning Tree Based Approach to Identifying Web Services. International Journal of Web Services Research 1(1), 1–20 (2004)
3. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice-Hall, New York (2005)
4. Arsanjani, A.: Service-Oriented Modeling and Architecture (SOMA), IBM white paper (2005)
5. Fensel, D., Bussler, D., Ding, Y., Omelayenko, B.: The Web Service Modeling Framework WSMF. In: Electronic Commerce Research and Applications, vol. 1(2), pp. 113–137. Elsevier B.V., Amsterdam (2002)
6. Herzum, P., Sims, O.: Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise. John Wiley & Sons Inc., Chichester (2000)
7. Vitharana, P., Zahedi, F., Jain, H.: Component-based Software Development: Design, Retrieval, and Assembly. Communications of the ACM 46(11), 97–102 (2003)
8. Bell, M.: Service-Oriented Modeling: Service Analysis, Design, and Architecture. John Wiley & Sons Inc., Chichester (2008)
9. Sehmi, A., Schwegler, B.: Service Oriented Modeling for Connected Systems (Part 1). Microsoft Architect Journal (7) (2006)
10. Zhijun, Y.: Consistency Analysis of Interorganizational Processes Based on Activity Diagrams. In: Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE 2005), pp. 187–190. IEEE Computer society, Los Alamitos (2005)
11. Hanson, J.: Coarse-grained Interfaces Enable Service Composition in SOA. JavaOne article (2003)
12. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1, W2C Note (2001)