Computer Science and Information Systems 16(2):381–407 https://doi.org/10.2298/CSIS180205001L

Goal-oriented Dependency Analysis for Service Identification ⋆

Jiawei Li[1], Wenge Rong[2], Chuantao Yin[1], and Zhang Xiong[2]

[1] Sino-French Engineer School, Beihang University, Beijing 100191, China
{jiaweili, chuantao.yin}@buaa.edu.cn
[2] School of Computer Science and Engineering, Beihang University, Beijing 100191, China
{w.rong, xiongz}@buaa.edu.cn

Abstract. Highly mature service-oriented architecture systems have great flexibility and reusability, and can align business processes and information technologies with high quality. Service identification plays a key role in this respect. Further, of the different methods employed, the most popular and preferred is process-oriented service identification. However, the absence of dependency analysis in the business process management domain remains a challenge for the quality of future systems. In this paper, we propose a goal-oriented dependency analysis for service identification via business process modeling. In our analysis solution, we apply a dependency tree featuring the relationships among requirements. The dependency relations are analyzed to create business processes via scenarios comprising requirements and process fragments.

Keywords: Service Oriented Computing, Service Identification, Business Process, Dependency

1. Introduction

Aligning business processes and information technology (IT) is an important strategy during a company's development. The results are irreplaceable in the resultant information system architecture [2]. To manage this alignment with different IT implementations, several solutions have been proposed. Service-oriented architectures (SOA) increase versatility and flexibility within a company [54,11,41]. To benefit from SOA, it is essential to define its governance [25]. This becomes a benchmark for justifying whether a given SOA system has achieved its goal. It is nearly impossible to build a perfect SOA system on the first attempt. Therefore, the maturity level and the current state of the system must be analyzed. To this end, several methods have been proposed in the literature. For example, the Combined SOA Maturity Model provides a 7-level maturity process for SOA systems [45]. Similarly, the Independent SOA Maturity Model offers a 5-level process that provides a pathway for SOA systems to become more flexible and mature [42].

A fundamental requirement for SOA governance when pursuing business–IT alignment is fulfilling the need for reusable services in the system [2]. Achieving the proper granularity of a specific service has a significant effect on the reusability of the whole system [17]. As a first and fundamental phase of the management of the SOA's life-cycle,

⋆ Corresponding author: Wenge Rong

service identification helps guarantee the business–IT strategy alignment by communicating business-related issues from an IT perspective [6]. The outcome of this phase influences not only the alignment between strategies [44], but also the development of future systems [35].

Currently there are three main strategies used to identify services within SOA: bottom-up, meet-in-the-middle, and top-down [5]. The top-down strategy is the most popular and most widely used [19]. Of the different kinds of top-down methods, process-driven service identification addresses alignment [16]. Process-oriented solutions for service identification capture functional business requirements. However, non-functional requirements (NFR) are important in business-process modeling, because it provides associated restrictions and constraints [1]. Maintaining the awareness of such dependencies is a challenge, and is helpful for detecting possible conflicts during the early stages [40]. It is difficult to develop a good SOA for complex systems when the complex relations between services are not fully considered [33]. In these studies, it was argued that, when extracting services from business processes, non-functional dependencies should also be assigned importance levels to increase the dependability of identified services.

The degree of dependency between requirements has been proven to have a significant impact on future defects [51]. Moreover, when complexity increases, the number of system errors increases significantly. If we underestimate the importance of dependency, it may result in different bottlenecks and blockages in workflows [47]. Moreover, the idea of services with high adaptability to business changes focuses on managing the dependent relations between business requirements and IT realization [46]. Consequently, it is important to precisely catalogue the dependency-analysis methods.

To solve the dependency-detection problem, many methods have been proposed. User requirement notation (URN) was proposed to provide a more powerful process-modeling language that focuses on dependencies by including goal-dependency management [40]. The authors argued that three perspectives should be guaranteed to achieve this goal: process modeling, goal dependency management, and goal/process traceability. It is thus essential to develop goal-dependency management and goal/process traceability. Whereas URN is powerful with respect to dependencies, it is difficult to implement because of the lack of a suitable design pattern. In the literature, business process management notation (BPMN) is a more user-friendly and popular tool, owing to its graphic presentation [55,43].

Other solutions have been employed to solve this problem by focusing on requirement dependencies [51]. One dependency-detection solution is goal-oriented requirement engineering, which usually applies a model-oriented thinking process [37]. Another approach is i*, which is a pure dependency analysis language proposed for all kinds of possible dependencies [55,15]. In the latest i* model, iStar 2.0, [12], the language was standardized. As a model language, iStar 2.0 proposed relation types without quantitative values to evaluation relations.

Several other model-oriented requirements-engineering methods have been proposed. NFRs are typically more representative of user behavior [36]. However, the logical relationship to business goals is not included. Therefore, Knowledge Acquisition in Automated Specification of Software Systems (KAOS) was proposed to solve this problem [26]. Both NFR [36] and KAOS [27] tended to increase the quantification and traceability of the requirements domain during engineering. Alternatively, GoalBPM was an informal

framework for goal/process traceability [24]. Unfortunately, this solution was dependent on an ambiguous definition of effects.

Cooperating with a model-oriented requirement traceability, Cooperative Requirements Engineering with Scenarios (CREWS) can easily obtain scenarios pertaining to requirements [48]. During the development phase of services, business goals and objectives become performance indicators [39]. Scenarios can be used to trace service performances and goal/process traceability.

Dependency analysis has been successful in requirements management, business process management, and service identification [28]. In this study, we integrate dependency with service identification. First, we model the requirements in the form of scenarios in the requirement-acquisition phase, because the business process is another representation of requirements [7]. Then, the scenario is translated into process fragments [13], which become part of the business process. Each fragment represents a candidate service. Finally, services are grouped per the dependency analysis results. By analyzing the dependency among process fragments, this method identifies services with respect to the successful traceability of business goals, and it processes the dependency relation obtained from the requirement analysis.

Extant dependency analysis methods focus on the graphical representation of dependencies between requirements. One example of dependency analysis is the use of key performance indicators to trace requirements [52]. Another example is iStar 2.0 [12], which uses a dependency net for organized dependency. iStar 2.0 proposes different types of dependencies without quantitative evaluation to identify services. To produce a measurable definition of dependency, we propose a goal-oriented dependency analysis for services identification.

The rest of the paper is organized as follows. In Section 2, we introduce the background to service identification and related methods from a process-oriented perspective. In Section 3, we present details of the proposed method. In Section 4, we evaluate and discuss our method using a case study. Finally, in Section 5, we conclude the paper and present possible future work.

2. Related Work

The alignment of a business–IT strategy is important to an organization's success, considering the fierce market competition and different solutions presented in the literature [18]. As an early attempt to use enterprise architecture, ATIS [3] leveraged the Zachmann framework to measure technology alignment [10]. Recently, with the development of SOA, it was lauded as a feasible method of improving IT governance in the business domain [9].

To implement efficient SOA-based applications, one preliminary task is to obtain proper services [4]. A straightforward idea is to use business entities for service identification by analyzing the relationships among entities [35]. Every element of a business is considered a business entity, and those with strong relations are grouped as services. An example of a business entity is the business process, widely adopted in service identification as a top-down oriented solution, owing to the similarity between business and IT processes [5,19].

Generally, a top-down strategy can have two types of inputs: use cases and business processes [20,22]. Compared to business processes, use cases do not consider tasks that have the same function as units [5]. Alternatively, business process-oriented service identification should design proper metrics for coupling and cohesion [49]. This constitutes the bases for different approaches.

One example of a business process-oriented top-down method was proposed by Kim et al., who created services by analyzing and grouping different business processes or workflows with minimum communication between them [21]. The underlying argument was that a service should represent a group of tasks. Thus, there should be less communication to the outside and more centralization. Similarly, Ma et al. classified business processes by weighting different SOA characteristics, such that customers obtained a group of services with balanced characteristics, according to their needs [31]. Because SOA enhances the flexibility and reusability of services per its design principal [23], to balance the contradictory characteristics, the authors proposed matrix achieved the requirements of an information system.

Another process-driven method, P2S, analyzes the data being sent between tasks [4]. This is suitable for solving complex processes, where interoperation is realized by grouping collaborative tasks. By applying a new definition of business value to determine service definitions, P2S provides a solution to combine data analysis and design metrics. By this definition, business value is a product that is created or treated in one department of an organization and then transferred to another. At this step, P2S obtains several candidate services. Then, it uses pre-defined design metrics to group services together. P2S innovatively combines business values and design metrics to calculate services and improve effectiveness. Moreover, this method has proven to be efficient in decreasing errors.

However, most process-driven methods focus on decomposing business processes. A lack of analysis of their dependencies and goals leaves us to face another challenge with respect to quality analysis [52]. In fact, the reliability of SOA systems depends on the existence of a secure architecture for relation management [14]. However, such an information management system would be difficult to analyze [29]. Thus, it is important to consider the dependency between business processes during service identification.

Identifying dependencies in business processes is recognized as a fundamental challenge in the literature. One possible solution is to use URN [40]. Compared to other popular methods in Table 1, URN has high quality in terms of managing dependencies, including business-process modeling in the goal-management domain. However, its design pattern is incomplete for complex situations. To make it suitable for applications. Three essential parts are necessary [40]. It needs a graphical business-processing modeling language; it needs a goal-oriented method for managing requirements; and it needs a method to relate requirement engineering results to business processes.

Several methods are employed to manage goal-oriented requirements [50], their traceability, and their dependencies. Koliadis et al. proposed the GoalBPM framework [24], which linked BPMN with KAOS [26]. This framework controls goal satisfaction during business-process development. Another goal-oriented requirement traceability method is NFR [36], which goes further in terms of analyzing non-functional requirements and their relations. By classifying goals at different layers, NFR built a goal-oriented system similar to the KAOS model. Instead of focusing on the logical hierarchy among goals, NFR includes non-functional requirements as soft goals in the dependency tree. Instead of us-

Table 1. Different modeling languages for dependency management.

	BPMN	UML	iStar 2.0	NFR	URN
Sequenceflow	√	√	+/-	+/-	√
Roles	√	√	√	×	√
Activities	√	√	×	×	√
Events	√	√	×	×	√
Process Hierarchies	√	√	×	×	√
Goal Modeling	×	×	√	√	√
Goal Model Evaluation	×	×	×	√	√
Goal/Process Traceability	×	×	×	×	√

ing logic relations, as in KAOS, to analyze dependency relations, the NFR dependency tree focuses on the relationship between soft and functional goals. Another efficient goal-oriented method is iStar 2.0 [55,12]. Based on the analysis of the dependency relations among actors, iStar 2.0 forms self-explained modeling languages for tasks in business processes, which include not only the dependency among goals but also dependencies among actors or tasks. At the goal level, iStar 2.0 proposes refinement relationships for goals. See Table 2. For refinement links, iStar 2.0 defines AND-refinement and OR-refinement relationships. However, it does not propose an evaluation method to measure the degree of dependency.

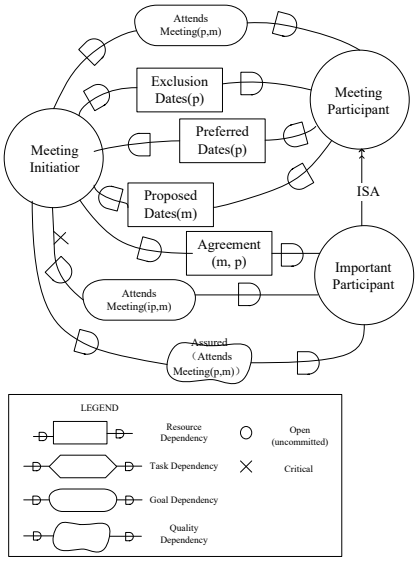


Fig. 1. iStar 2.0 modeling language example

From the literature, business processes have had a close relationship with requirement engineering [7]. Thus, a business process is simply another representation of related and

Table 2. Links between elements in iStar 2.0 model

	Goal	Quality	Task	Resource
Goal	Refinement	Contribution	Refinement	NA
Quality	Qualification	Contribution	Qualification	Qualification
Task	Refinement	Contribution	Refinement	NA
Resource	NA	Contribution	NeededBy	NA

elicited requirements. To model and verify a business process, we must find a suitable requirement engineering method [49,53]. Process fragments [13] are designed to specify an action that is needed to compare business processes with itself in order to manage the overall process. Matching a scenario of requirements to a process fragment helps us understand the logic inside a business process.

3. Dependency-Aware Service Identification

From the above discussion, business process-oriented service identification is promising for SOA-based business–IT alignment, and dependencies among processes should be emphasized simultaneously. There are many dependency detection and analysis tools in the literature, and the methods used to employ their ideas for service identification vary. In this research, we propose a dependency-aware process analysis framework, where we first employ BPMN to model business processes, because BPMN is a powerful extended markup language-oriented machine-friendly language. It enables more choices for gateways and special cases and graphical representations of business processes for ease of understanding [8].

Specifically, we adopted a 3-stage service-identification mechanism for this research. In the requirement-acquisition phase, we recognize requirements as scenarios using the popular CREWS–Scenarios for Acquiring and Validating Requirements [48]. After developing a library of requirements, we develop a KAOS goal-oriented model [26], as a goal-dependency study [32]. We match scenarios using process fragments and. We group service candidates according to the dependency tree, where requirements with dependent relations located in the same root goal have high affinity.

3.1. Requirement Acquisition

The first task for service identification is to define the dependency between different business processes. To analyze the dependency, it is necessary to understand the requirements, because dependencies give rise to conflicts between requirements. To this end, we define requirements as follows:

$$R = \{Id, D, S, Sc\}. \tag{1}$$

In this dependency-analysis method, a requirement, R, is defined by a unique identifier, Id, which serves to guide the relation between requirements and goals when the requirement description changes. For ease of understanding, the description information, D, stored in a unique requirement, should be of a semantic form. It is also possible to trace back to the source of a requirement. The source, S, helps the requirement engineer

review the need for the requirement. The set of scenarios, Sc, included in the definition of a requirement is a representation of traceability management and dependency analysis. CREWS [48] is a model-oriented method employed for scenario construction. In this definition, a unique requirement can have more than one scenario.

After requirements are defined, the next challenge is representing the dependency among requirements. In this research, we employ the dependency tree per the goal-oriented requirement engineering principal by combining the logic relation defined in KAOS [26] and the goal's level distributions of NFR [36]. A branch in the requirement dependency tree is defined as follows:

$$K = \{R, Go, Tt, Sr\}. \tag{2}$$

From the definition of a branch, this equation contains information about the requirements parent and child goals. A branch always points from the leaves to the root. There are two kinds of branches: "AND" branches and "OR" branches. Both branches signify the logical relation between sub-goals and goals. The logical relationship between goals helps identify dependencies between sub goals. The rules are defined in the dependency analysis section. The satisfaction coefficient of a dependency branch is given by the dependency relation between the goal and its sub-goal. To obtain the satisfaction level of a goal, we work from the bottom of the dependency tree. The satisfaction level is classified as "satisfied," "weak," or "unsatisfied." A requirement with all of its scenarios satisfied by the business process will have the state, "satisfied." If only some scenarios are satisfied, the relation is "weak." Otherwise, the requirement is "unsatisfied." This satisfaction relation occurs between the parent goal and a sub-goal, and it can be translated as another form of dependency for the destination goal.

According to the definition of "scenario" in [48], we define a scenario as a sequence of events having one possible pathway through a use case containing some actions.

$$Sce_j = \{ev_0, ..., ev_p\}, \tag{3}$$

where two types of scenarios are further defined. The execution scenario is designed for execution. This kind of scenario has a positive effect on the parent goal. A forbidden scenario is a constraint that should not be executed. Forbidden scenarios have negative effects on the parent goal. When we wish to control for the greatest satisfaction of one goal, it is necessary to combine both positive and negative influences of the sub-goals.

In this research, a scenario is formed by events. In [48], an event could specify the system status before or after an actions resulting in a change. To simplify the comparison between scenarios and business processes, we use only the information of the changing state (i.e., event) but not the details of the action needed in the requirements. Therefore, one event can be defined as a set of data with its new state and the information of the changing source. Each dataset has a data object, a state of data, and a changing source.

$$ev = \{Dt_1, ..., Dt_q\} \tag{4}$$

$$Dt_l = \{Do, st, sc\} \tag{5}$$

Another kind of event is the condition for execution. This event only exists for a condition flux or a condition gateway. A condition event contains one condition description line and a chosen condition. With the chosen condition, we can orient the condition with a

certain condition flux. This kind of event helps us discover complex structures of process fragments.

$$ev = \{Cd_1, ..., Cd_q\} \tag{6}$$

The last kind of event is of temporal significance: state of system. If we need to locate a scenario involving the specific state of a system, it can be found in an event. The state of a system is defined as the need of a company. This kind of event can help to not only define the significant time points for the system, but also the waiting-time for the system. The BPMN modeling business process has several special time events requiring time significance (e.g., interrupted events). Interrupted events make the system wait for a period before executing the predefined action.

$$ev = \{St_1, ..., St_q\} \tag{7}$$

In the analysis of the similarity between scenarios and business processes, process fragments are a part of business processes and can be located as follows.

$$PF = \{Id, T, E, F, A, G, L, \Delta\}. \tag{8}$$

A process fragment is connected to a unique requirement. Therefore, it contains the requirement identification. Inside a process fragment, information exists to rebuild a business process section , including the set of tasks, the set of different kinds of associations, the set of gateways, the set of lanes, and the set of data. Depending on the type of BPMN element, each has its own definition, and they differ according to their identification. Therefore, we follow the identification of each element. There are two types of connecting elements: flow and association.

A sequence flow is the basic connecting element in the BPMN language, and it contains information about the source and the target references. A message flow is a special flow that includes additional information about a message sent in the same direction as the flow itself. As with the definition of the flow, it uses data association. The difference between a basic flow and a simple association is whether or not the two connected elements belong to the same participant. If an association is simple, it connects an internal task with one outside the current participant. To trace the data information of a task, we collect information about the data association. A data association has two additional important variables compared to a basic association: $ioSpecification$ and $DataSet$. If data association is linked to the input data, the $ioSpecification$ is "input," and the $DataSet$ is an inputSet. If the data association is linked to the output data, the $ioSpecification$ is "output," and the $DataSet$ is an outputSet.

For ease of management in data information, the process fragment uses Δ as a set of data. Input Data is a data object linked to a data association with $ioSpecification$="input." Output Data is a data object linked to a data association with $ioSpecification$="output." An event shows changes in the state of data or information before and after a task. We can now compare the difference to understand a business fragment. The matching process is described in the next section.

3.2. Scenario Matching

The objective of this step is to locate a process fragment linking the scenario of a business process requirement. A business process, BP, has a similar definition as process fragment, pf:

$$BP = \{T, E, F, A, G, L, \Delta\}. \tag{9}$$

A business process should have at least one start event and one end event. Normally, a business process belongs to a process fragment. However, a process fragment is not always a business process. To manage the dependency of each business process, we define a relation-matching matrix, which maps the business process to the satisfied process fragment in a requirement. The satisfaction process-fragment management matrix linked to the giving business process saves information pertaining to connected requirements. This matrix is defined as:

$$M_i := [Id, pf_1, ..., pf_m]. \tag{10}$$

This matrix is a $1 \times (m + 1)$ matrix and belongs to a specific business process, where the result corresponds to the scenarios of one requirement. If the business process satisfies the scenario of this requirement, the corresponding pf_j equals 1. Otherwise, it is 0. The size of this matrix depends on the number of scenarios processed by the corresponding requirement. The entire management matrix forms the set, $M[n]$. After searching for the corresponding requirements and scenario sets for each M_i, we have a set of requirements linked to the business process, $R[n]$. For each chosen R_i, we have a set of process fragments, $PF_i[m]$, linked to them. For each R_i, we check each scenario, Sce_j. If the sequence of the process is found to match the sequence of events in the scenario, $pf_{ij+1} \in M_i$ is set to 1. Otherwise, it is set to 0.

Comparing a scenario and business process begins with the first event in Sce_j. According to the definition of PF, we can define a $\tau\{F, A, L, \Delta\}$. The elements belonging to F are sequence flows, condition flows, or default flows. Condition flows and default flows are considered special events. For these, we recognize the condition as information inside the data. In other words, when we meet a conditioned gateway, we should match the condition with the existing data content in an event. Otherwise, a task can only have one in and one out. Thus, neither the flow pointing to the task nor the flow leaving the task influences the comparison of scenarios and business processes. If a scenario has found a matching sequence of tasks, the flows in the business process will be succeeded by the process fragment. From the definitions of the relation between output data and a task and its input, we know that all data are linked to a certain task via data association or another association. Consequently, most comparisons consider the difference between associations and lanes.

Events normally occur either before or after a task. In this research, we assume that our process fragment involves a task before an event. However, tasks after the last event are not considered. First, we determine whether the belonging lane of a task is the same as the changing source of the event. A task's belonging lane should be the same as the changing source of the data post event. When an event has more than one changing source, this is possible only when the event occurs after a gateway. When an event has only one changing source for all data, a task can have only one input and one output. Data actions

generally have four states: create, read, update, and delete [4]. We group all actions (e.g., rewrite, fill up, send, and copy) in the update state, which represents operations performed on the data. Therefore, given a task, τ, and several flows, F, associations, A, swim lanes or collapsed pools, L, and some portions of data, D, linked to the tasks, we can determine the matching method for a satisfied scenario, as shown below.

To match an event with on in a business process, if a start event has a message mission, the same message should have at least $data \in ev$ and $data.state =' R'$, with the message being a part of the data. If an end event has a message mission, the same message should have at least data $data \in ev$ and $data.state =' U'$, with the message being a part of the data.

It is more difficult to match an event with a task than to match an event with another event in a business process. The matching rules are proposed depending on the state of the data. For a data event, if the data state is "C," (i.e., data is created during this task), we have

$$data_l.dataObject \in \tau.output. \tag{11}$$

When this piece of data is a message connected to a message association, then this message association is directed outwards. Most importantly, an object that is created during a task should not be found at any time before this task. For a data event with a data state, "U,"

$$data_l.dataObject \in \tau.input \ \& \ data_l.dataObject \in \tau.output. \tag{12}$$

When this piece of input data is a message connected to a message association, then this message association is directed inwards. When this piece of output data is a message connected to a message association, then this message association is directed outwards. Because updating is a complex operation on a piece of data, the detailed definition of the same update action should be defined by the company itself. For a data event with a data state given as "R," we get

$$data_l.dataObject \in \tau.input. \tag{13}$$

For a data event with a data state given as "D," the data situation should be given as an "R" state. However, in this case, we should be sure that this object will no longer be used.

When an event is found to match the data states of two tasks, the task in front of the testing event will be examined if it is in the assumed lane. If so, the task will be a part of the process fragment. Because we consider the business process for a scenario, we will have the result of satisfaction. For a scenario where we find a process fragment that fulfills all scenario events, this scenario is satisfied. Otherwise, it is unsatisfied. To build a process fragment, we ignore the tasks or gateways between two matched tasks and use a simple flow for connection. If the matched tasks have a parallel, inclusive, or exclusive relation between them, the gateway relation should be inherited by the process fragment. After building the matching-process fragment, we obtain several matching matrices for the relation between the business process and requirements.

3.3. Service Grouping

In this phase, we already have a business process linked to requirements with a matching matrix. Because we used the scenario comparison, the location of the requirement should

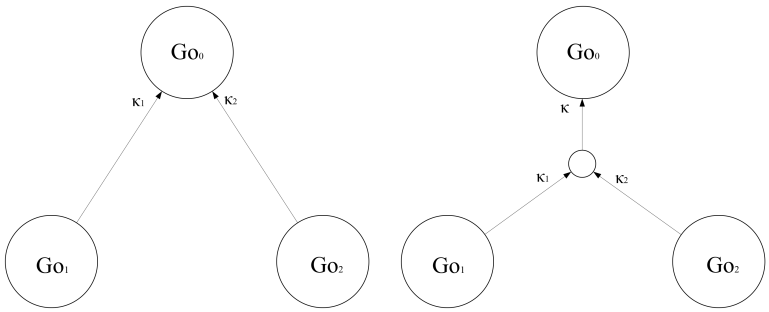
group several tasks together, or they may be located inside one task. A matching scenario forms a candidate service. For candidate services that satisfy the same requirement, we propose that they be grouped together. If a task is identified as being used by several requirements, we recommend grouping services per the minimum connection rule with respect to how loosely coupled they are.

If two process fragments are situated next to one another, we should go through the requirement dependency-relation tree to minimize the dependent relation between two services. The dependent requirement will only be analyzed for one generation, which indicates the leaf generation for the requirement. The resulting service dependency relation is defined as follows:

$$Rs = \{M_0, ..., M_x\}. \tag{14}$$

This is a set of matching matrices for requirements with a satisfaction level of at least "weak". The dependency relation is traced back to the dependency tree by the matching matrix. On each occasion, when a service changes, we trace back to the related requirements for verification, and, according to the goal-oriented model, we obtain a list of possibly impacted services. In the case where there are changes to a specific requirement, services linked to the requirement can be modified rapidly.

To analyze the dependency, we need a goal-dependency coefficient that has a direct relation with the dependency tree. Apart from the branch that points to a goal null, each branch of the dependency tree has a coefficient for the identification of the contribution of a sub-goal, and each coefficient should be between 0 and 1.



(a) Or relation (b) And relation

Fig. 2. Goal relation.

For a goal-dependency relation type, "OR," as shown in Fig. 2(a), we define the coefficient of the dependency as a measurable degree to enable us to understand how well the goal can be satisfied by satisfying the sub-goal. This kind of dependency between parent goal and sub-goal is the level of satisfaction that is contributed by the child goal to the satisfaction of the parent goal. The satisfaction dependency is created by the use of the model for requirement management.

$$\kappa_1, \kappa_2 \in [0, 1] \; and \; \kappa_1 + \kappa_2 \leq 1. \tag{15}$$

The dependency relation between the two sub-goals, $G_D(Go_1, Go_2) = 0$. Because we ignore the other types of dependency, two different sub-goals with the "OR" relation will not influence one another. Then, the dependency relation between Go and Go_1 or Go_2 should be the same value as the coefficient of dependency:

$$G_D(Go_0, Go_1) = \kappa_1,\ G_D(Go_0, Go_2) = \kappa_2. \tag{16}$$

For a goal-dependency relation type, "AND," as shown in Fig. 2(b), the coefficient of dependency should be as follows:

$$\kappa \in [0, 1]\ and\ \kappa_1 \times \kappa_2 = \kappa. \tag{17}$$

In the "AND" relation, two sub-goals have a higher dependency on each another than with the "OR" relation. When we analyze their relations, it is easy to tell if one goal of this type of relation causes a conflict with the parent goal. Their combined effect should also be negative to the parent goal. Therefore, for an "AND" relation, two sub-goals should be at least weakly satisfied for a satisfied goal, Go. Moreover, the dependency relation between the two sub-goals, $G_D(Go_1, Go_2) = 1$, meaning the two sub-goals are not independent of each other and that they should cooperate for the parent goal.

To calculate the goal-dependency relation of a given goal, Go_x, with another goal, Go_y, when we already have a known $\overline{G_D(A, B)}$, we have:

$$G_D(Go_x, Go_y) = G_D(Go_x, A) \times \overline{G_D(A, B)} \times G_D(B, Go_y). \tag{18}$$

Given the definition of the dependency equation between goals, we should find the co-parent for these two goals in the lowest position to obtain their dependency coefficient. Using the special coefficient calculation equation, we predefine if $G_D(Go, Go) = 1$. In other words, one requirement dependent entirely depends on itself, because it shares the same resources with itself.

Using requirement-dependency equations, we can thus conclude a dependency calculation equation, as follows, for two identified services:

$$Y = \frac{\sum_{0 \le i \le u}^{0 \le j \le v} G_D(Go_i, Go_j)}{u \times v}. \tag{19}$$

In this equation, the dependency between services is calculated by the sum of each of their requirements. u and v represent the number of requirements belonging to the two services that are compared.

4. Case Study

To evaluate the service identification method, we performed a case study of the booking process to validate its capability. The booking process contains a basic hotel booking and an entertainment service that is an alternative for customers. Each reservation should be paid for a confirmation of booking. The reservation process is shown in Fig. 3.

To deal with the reservation requirement, the employee of the sales department will show the customer a detailed table of prices. If the customer is not satisfied with the prices and decides against reserving a room or a ticket, the process will end. If they continue to

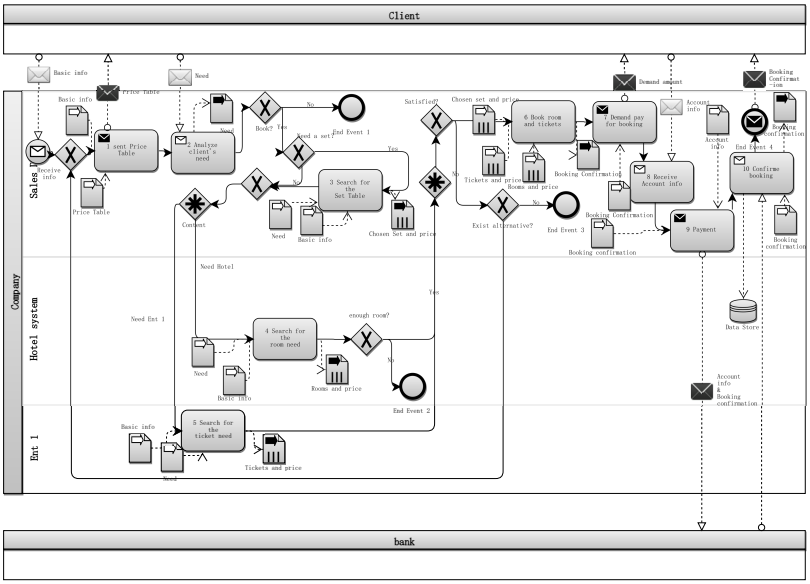


Fig. 3. Booking Process.

the next step, the customer can select from booking only for rooms, only for tickets, or for both. After the booking process, the customer will be either satisfied or unsatisfied with the search result. If they need to look up an alternative, it will be easy to restart from the beginning. When the booking process is completed, the customers are required to pay a reservation fare. Afterwards, a booking confirmation will be sent to the customers.

To calculate the dependency between requirements so that we can reuse the results for obtaining services, we developed a Java-based tool. The first tab of the application is designed for the information of the business process shown in Fig.4.

4.1. Requirement Acquisition

This booking process is linked to several requirements. We have a list of main requirements. The goal-oriented model is built upon the KAOS model proposed in [26]. This goal-oriented model is built on a tree model with a goal-level definition from an NFR requirement management tree and a logic relation definition from a KAOS dependency tree. First, the requirements for this booking process can be derived as follows:

1) R1: Customers want to book hotels or entertainment tickets. 2) R2: Customers want to view the price table. 3) R3: Customers want to receive booking confirmation feedback at the end of the booking. 4) R4: The marketing department wants to promote a different package of tickets to customers. 5) R5: The hotel wants to avoid over-booking. 6) R6: The financial department wants to collect a reservation fee before the booking process ends. 7) R7: Customers want to return to review the price table if not satisfied. 8) R8: The financial department wants to charge booking fee to confirm the booking.

In the tool developed for dependency calculation, we can use requirement management windows to insert a new requirement into the tool, as shown in Fig. 5. In this win-

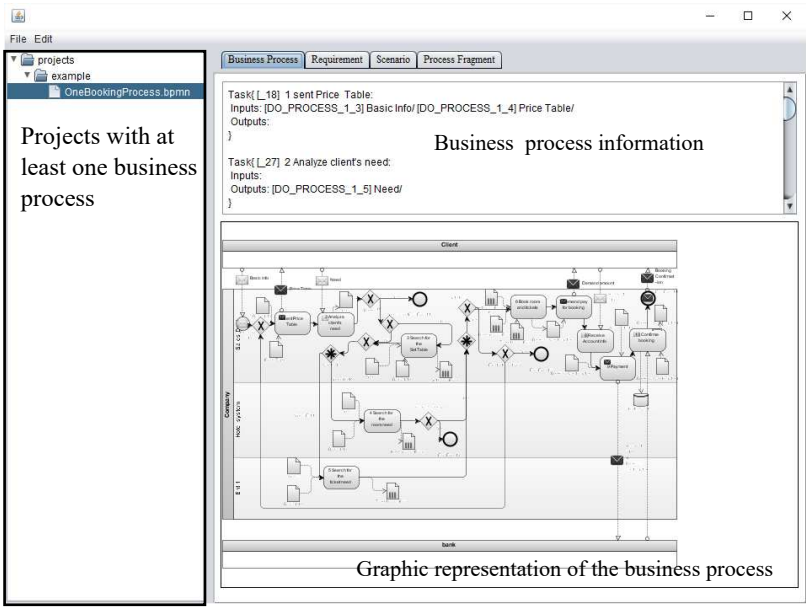


Fig. 4. Dependency Aware Requirement Analysis Tool.

dow, if we create a requirement without pointing it to a parent goal that is not null, we create the goal in terms of strategy levels. If there are choices with respect to the parent goal, a new goal can be chosen from among them. When a goal is connected to its parent goal with a logic AND, it can choose from a list of possible sub-goals of this parent goal with logic AND. Because all sub-goals with logic AND are not connected directly to their parent goal, these choices will influence the dependency analysis process.

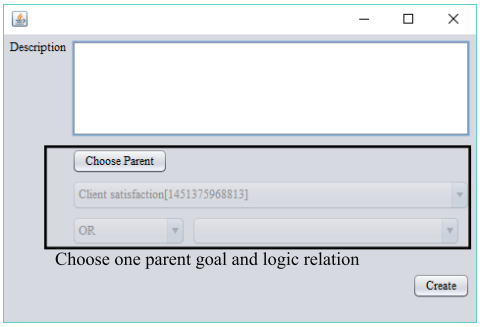


Fig. 5. Requirement-management window used to develop new requirements.

After creating requirements, the dependency tree is automatically built. The dependency in this study is equally distributed. In other words, we consider that all requirements

can fully satisfy their parent goals if satisfied. Then, each sub-goal is equally important according to its logic relation. We can therefore obtain a requirement table using all the information inside, as shown in Fig. 6. In the case of modifying the information of one requirement, we can simply select a row of this table and change the information in the form below it. The dependency tree of this case study is shown in Fig. 7.

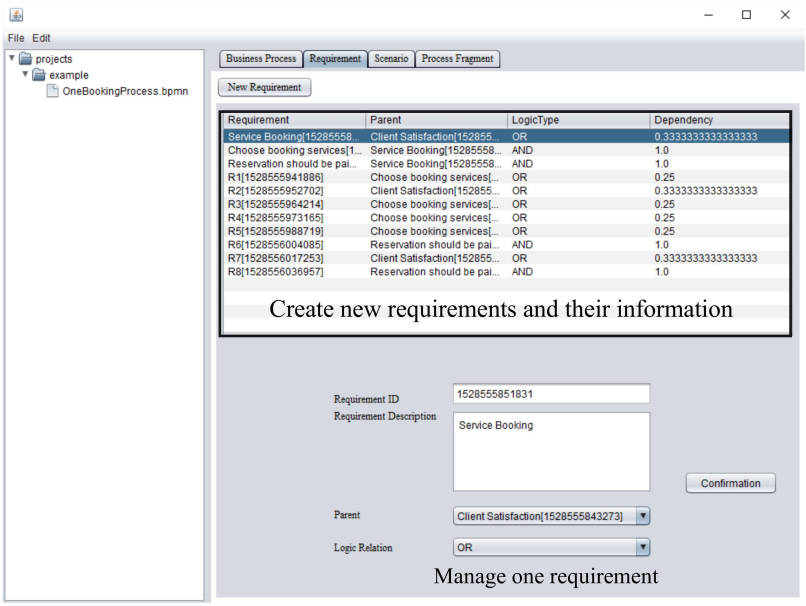


Fig. 6. Requirement-management window for information and editing information.

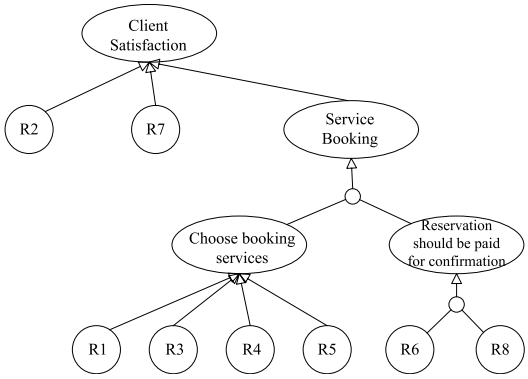


Fig. 7. Goal-oriented model.

Taking requirement R1 as an example, we can derive a requirement and scenarios by using the method proposed in [48], as shown in Fig. 8. We first study requirement R1 and obtain a use case with two possible actions taken by clients. Before these two actions, we can create a “need document. After reading the needs of clients and choosing rooms or tickets for the client, the price shows up, and the process produces a “booking confirmation document.

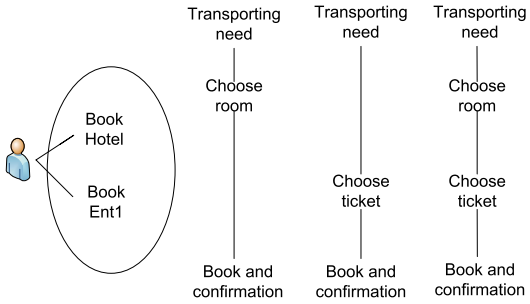


Fig. 8. Scenarios and use case for requirement R1.

Scenario	Events
Scenario 1	Need(C) - Need(R) & RoomChosen(C) - RoomChosen(R) & Confirmation(C)
Scenario 2	Need(C) - Need(R) & Ent1Chosen(C) - Ent1Chosen(R) & Confirmation(C)
Scenario 3	Need(C) - Need(R) & RoomChosen(C) - Need(R) & Ent1Chosen(C) - RoomChosen(R) & Ent1Chosen(R) & Confirmation(C)

Table 3. Scenarios of requirement R1.

According to Table 3, we can use the tab scenario to model the scenario of requirement R1 and the other requirements using data events. Otherwise, we can have the list of other requirements modeled using scenarios shown in Table 4. When we apply all the information related to the scenarios, we obtained in the tool the table shown in Fig. 9

4.2. Scenario Matching

After obtaining scenarios, we can match the process fragment. In the tab, “process fragment, if we apply the “refresh button, we can obtain a simplified version of the process fragment according to certain scenarios. With the help of automatic calculation, we can remodel each process fragment. To continue the analysis, the detailed results of process fragments are shown in Fig. 1 to 8 in the process fragments section in the appendix. Therefore, we can have a group of tasks as a candidate service $\{\tau_2, \tau_4, \tau_5, \tau_6\}$ for requirement R1. Similarly, we have $\{StartEvent, \tau_1, EndEvent_1\}$ for requirement R2, $\{EndEvent_6\}$ for requirement R3, $\{\tau_2, \tau_3, \tau_6\}$ for requirement R4, $\{\tau_4, EndEvent_2\}$ for requirement R5, $\{\tau_7, \tau_8, \tau_9, \tau_{10}, EndEvent_4\}$ for requirement R6,

Requirements	Events
R2	BasicInfo(R) - BasicInfo(R) & PriceTable(U)
R3	Confirmation(U)
R4	Need(R) - BasicInfo(R) & Need(R) & SetChosen(C) - SetChosen(R) & Confirmation(C)
R5	Need(R) & RoomChosen(C) - ”Room: No?”
R6	Confirmation(R) - AcountInfo(R) - AccountInfo(U) - Confirmation(U)
R7	”Exist alternative: No? Yes”
R8	Confirmation(C)

Table 4. Scenarios of requirements R2 to R8.

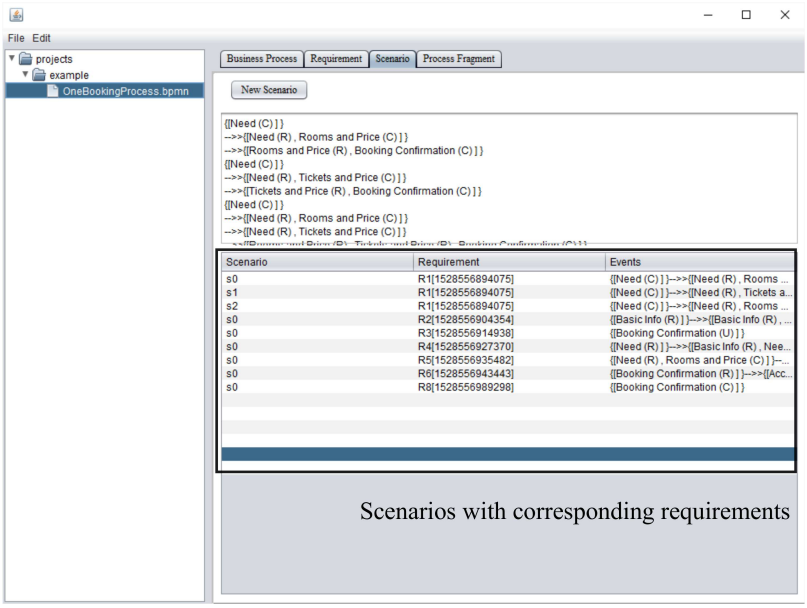


Fig. 9. Information about all scenarios.

$\{StartEvent, EndEvent_3\}$ for requirement R7, and $\{\tau_{10}, EndEvent_4\}$ for requirement R8.

In order to analyze the relation between two services, we calculated the dependency relation between them using the modeling tool shown in Fig. 10. The result is calculated automatically per the definition of dependency. According to the dependency tree, we can classify three main services: {StartEvent, τ_1}, {τ_2, $EndEvent_1$, τ_3, $EndEvent_2$, τ_4, τ_5, $EndEvent_3$, τ_6}, and {τ_7, τ_8, τ_9, τ_{10}, $EndEvent_4$}. With this proposition of candidate services, we can calculate the dependency between any two, and the results are shown in Table 5. According to the table, three services are relatively independent of each other.

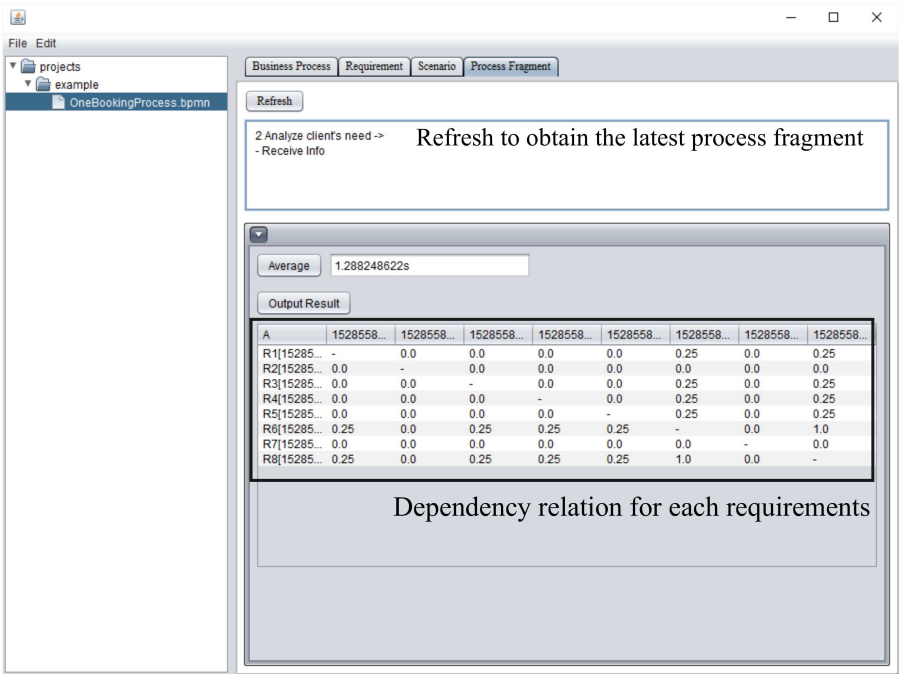


Fig. 10. Dependency relation for services.

Candidate services	Requirements	C1	C2	C3
C1	R_1, R_4, R_5	-	0	0.17
C2	R_2, R_7	0	-	0
C3	R_3, R_6, R_8	0.17	0	-

Table 5. Dependency between candidate services.

4.3. Discussion

To evaluate the proposed method, we compared it to other popular methods, as shown in Table 6. The [35] method was used in an attempt to cluster business entities, but the definition of business entities can change from person to person. Once the business process changes, the standard of business entities will need to be redefined. The [31] method was applied to a workflow. The result can change according to the weight matrix. However, we should not amend the weight matrix. This matrix will be used for different business needs and an amendment will cause a big impact. [21] provided another process-driven method that works by analyzing the Petri net. However, it depends on a rule respecting a minimum communication between services. When a new communication is established, it may influence the belonging location of a task. The structure of services may also change. [4] is the only method that can be tested for our case study. The result shows that R6 is separated. We determined that the reason was the absence of a connection between tasks 7 and 8 and tasks 9 and 10. Therefore, the business process should have enough details to enable the calculation.

The goal-oriented dependency analysis in the service-identification method has not only an advantage in the service-identification phase, but also with respect to the continuity of the life cycle of services. If the business process changes for a short period, all the other methods must redo the calculation. Because the services identified from our method are related to specific requirements, it is not expected that there would a significant change to the structure of services. However, the other method does not guarantee this.

Table 6. Comparison between service-identification methods.

Method	Inputs	Method	Evaluation	Apply to this case?	For future governance in SOA?
Goal-Oriented dependency analysis	Business process in BPMN	top-down	case study and evaluations	Yes 3 services	Yes. No need to redo the calculation. Services are traceable linking to requirements and they change only when requirements change.
[4]	Business process in BPMN	top-down	case study and evaluations	Yes 5 services	No. Should redo the calculation if business process changes.
[21]	Petri Net	top-down	case study and evaluation	No	No. Should redo the calculation if Petri net changes.
[31]	Workflow	top-down	case study	No	No. Should redo the calculation.
[35]	Business Entities	top-down	case study and evaluation	No	No. Should redo the calculation.

From the dependency relation table shown in Fig. 10, we can also get dependency calculation results for any two requirements. Depending on the number of requirements

accumulated in the first step, we will obtain a table of a different size. In this table, we can also verify the time required to obtain the matching process fragment. We then obtain the result shown in Table 7. The times taken to obtain all data events are nearly equal, and the number of events is not expected to change while obtaining results. The searching method used in this tool causes the only dependency of time based on the size of the business process.

Table 7. Testing the matching efficiency.

Requirement/Scenario	First data event	Last data event	Entire scenario	Average
R1sce1	0.023131306 s	0.035481503 s	0.022244722 s	0.020 s
R1sce2	0.038197306 s	0.02403289 s	0.020332213 s	0.022 s
R1sce3	0.020299843 s	0.02483421 s	0.024278812 s	0.022 s
R2	0.020369713 s	0.026578955 s	0.023192885 s	0.025 s
R3	0.020369713 s	0.023752625 s	0.027197117 s	0.022 s
R4	0.020315238 s	0.021352217 s	0.024698419 s	0.021 s
R5	0.020663793 s	0.021249191 s	0.020074053 s	0.021 s
R6	0.020581293 s	0.024200654 s	0.020899847 s	0.021 s
R8	0.020717083 s	0.020313659 s	0.024139469 s	0.020 s

5. Conclusion and Future Work

We proposed a goal-oriented dependency-analysis method for service-identification by finding the business requirements in a business process, realizing the dependency relation of requirements for business processes and services in SOA, and proposing a definition of dependency among services. As shown in the case study, this method can provide another proposed standard for classifying tasks to services before applying design metrics to identify services. We considered the definition of cohesion; the loosely coupling of SOA is closely linked to the requirement. A service that integrates fewer numbers of possible requirements is more specific. A service will be more independent if it is not required cooperate with another service serving the same requirement. Therefore, in this study, we developed a tool to manage requirements and calculate the process fragments. Moreover, with the predefinition of dependency equations, we can easily obtain the dependency among services.

Because there are still a variety of gateways and events in business processes, process fragments face more complex business processes with which they should be matched. Additionally, we plan to study the case where a business process does not fully satisfy a requirement. For example, if the requirement is difficult to fulfill because of limited capability, the important part of the scenario will be satisfied and the rest will be ignored. In this case, it should still be possible to recognize the requirement in a business process.

For future work, service identification will be extended to services with web service definition language so that the identified services can be discovered. Then, by pairing business process with IT processes [20], it will possible to develop a service-identification phase that is more traceable both from requirements and technical perspectives. Furthermore, when applying SOA to the design of Web services, the low frequency with which

services are reused is a challenging task [54]. To decrease the difficulty of finding services, several solutions have been proposed, for which a complete knowledge warehouse appears to be a promising solution [38]. However, it difficult to relocate a particular service from a large service center [30]. Thus, it would be a large amount of management work for the service center. In this paper, we proposed to organize different services based on the requirements management method, because it helps improve the efficiency of managed changes [34]. Therefore, there is a need for further research to develop linkages between requirements and the knowledge-based services center.

Acknowledgments. This work was partially supported by the National Natural Science Foundation of China (No. 61472021).

References

1. Aburub, F., Odeh, M., Beeson, I.: Modelling non-functional requirements of business processes. Information & Software Technology 49(11-12), 1162–1171 (2007)
2. Aversano, L., Grasso, C., Tortorella, M.: A literature review of business/it alignment strategies. In: Proceedings of 14th International Conference on Enterprise Information Systems. pp. 471–488 (2012)
3. Avila, O., Goepp, V., Kiefer, F.: ATIS: A method for the complete alignment of technical information systems. International Journal of Computer Integrated Manufacturing 24(11), 993–1009 (2011)
4. Bianchini, D., Cappiello, C., Antonellis, V.D., Pernici, B.: Service identification in interorganizational process design. IEEE Transactions on Services Computing 7(2), 265–278 (2014)
5. Bianchini, D., Pagliarecci, F., Spalazzi, L.: From service identification to service selection: An interleaved perspective. In: Proceedings of Formal Modeling: Actors, Open Systems, Biological Systems - Essays Dedicated to Carolyn Talcott on the Occasion of Her 70th Birthday. pp. 223–240 (2011)
6. Börner, R., Goeken, M.: Identification of business services literature review and lessons learned. In: Proceedings of 15th Americas Conference on Information Systems (2009)
7. Castano, S., Antonellis, V.D., Melchiori, M.: A methodology and tool environment for process analysis and reengineering. Data & Knowledge Engineering 31(3), 253–278 (1999)
8. Chinosi, M., Trombetta, A.: BPMN: an introduction to the standard. Computer Standards & Interfaces 34(1), 124–134 (2012)
9. Choi, J., Nazareth, D.L., Jain, H.K.: The impact of SOA implementation on it-business alignment: A system dynamics approach. ACM Transactions on Management Information Systems 4(1), 3 (2013)
10. Dahman, K., Charoy, F., Godart, C.: Alignment and change propagation between business processes and service-oriented architectures. In: Proceedings of 2013 IEEE International Conference on Services Computing. pp. 168–175 (2013)
11. Dai, W.W., Vyatkin, V., Christensen, J.H., Dubinin, V.N.: Bridging service-oriented architecture and IEC 61499 for flexibility and interoperability. IEEE Transactions on Industrial Informatics 11(3), 771–781 (2015)
12. Dalpiaz, F., Franch, X., Horkoff, J.: istar 2.0 language guide. CoRR abs/1605.07767 (2016)
13. Daniel, F., Casati, F., D'Andrea, V., Mulo, E., Zdun, U., Dustdar, S., Strauch, S., Schumm, D., Leymann, F., Sebahi, S., Marchi, F.D., Hacid, M.: Business compliance governance in service-oriented architectures. In: Proceedings of 23rd IEEE International Conference on Advanced Information Networking and Applications. pp. 113–120 (2009)
14. Delac, G., Silic, M., Srbljic, S.: A reliability improvement method for soa-based applications. IEEE Transactions on Dependable and Secure Computing 12(2), 136–149 (2015)

15. Gonçalves, E., Castro, J., Araújo, J., Heineck, T.: A systematic literature review of istar extensions. Journal of Systems and Software 137, 1–33 (2018)
16. Gu, Q., Lago, P.: Service identification methods: A systematic literature review. In: Proceedings of 3rd European Conference on ServiceWave. pp. 37–50 (2010)
17. Haesen, R., Snoeck, M., Lemahieu, W., Poelmans, S.: On the definition of service granularity and its architectural impact. In: Proceedings of 2008 Advanced Information Systems Engineering, 20th International Conference. pp. 375–389 (2008)
18. Henderson, J.C., Venkatraman, N.: Strategic alignment: Leveraging information technology for transforming organizations. IBM Systems Journal 32(1), 4–16 (1993)
19. Huergo, R.S., Pires, P.F., Delicato, F.C., Costa, B., Cavalcante, E., Batista, T.: A systematic survey of service identification methods. Service Oriented Computing and Applications 8(3), 199–219 (2014)
20. Inaganti, S., Gopala, Behara, K.: Service identification: BPM and SOA handshake. BPTrends (2007)
21. Kim, Y., Doh, K.: Formal identification of right-grained services for service-oriented modeling. In: Proceedings of 10th International Conference on Web Information Systems Engineering. pp. 261–273 (2009)
22. Kim, Y., Doh, K.: Use-case driven service modelling with xml-based tailoring for SOA. International Journal of Web and Grid Services 9(1), 35–53 (2013)
23. Kohlborn, T., Korthaus, A., Chan, T., Rosemann, M.: Identification and analysis of business and software services - A consolidated approach. IEEE Transactions on Services Computing 2(1), 50–64 (2009)
24. Koliadis, G., Ghose, A.: Relating business process models to goal-oriented requirements models in KAOS. In: Proceedings of 2006 Pacific Rim Knowledge Acquisition Workshop on Advances in Knowledge Acquisition and Management. pp. 25–39 (2006)
25. Koumaditis, K., Themistocleous, M.: A detailed framework for SOA governance. International Journal of Systems and Service-Oriented Engineering 5(3), 52–74 (2015)
26. van Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. In: Proceedings of 5th IEEE International Symposium on Requirements Engineering. pp. 249–262 (2001)
27. Letier, E., van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. In: Proceedings of the 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering. pp. 53–62 (2004)
28. Li, J., Rong, W., Yin, C., Xiong, Z.: Dependency aware business process analysis for service identification. In: Proceedings of 9th Asia-Pacific Services Computing Conference. pp. 137–152 (2015)
29. Li, Q., Wang, Z., Cao, Z., Du, R., Luo, H.: Process and data fragmentation-oriented enterprise network integration with collaboration modelling and collaboration agents. Enterprise Information Systems 9(5-6), 468–498 (2015)
30. Llinas, G.A.G., Nagi, R.: Network and qos-based selection of complementary services. IEEE Transactions on Services Computing 8(1), 79–91 (2015)
31. Ma, Q., Zhou, N., Zhu, Y., Wang, H.: Evaluating service identification with design metrics on business process decomposition. In: Proceedings of 2009 IEEE International Conference on Services Computing. pp. 160–167 (2009)
32. Maiden, N.A.M., Lockerbie, J., Randall, D., Jones, S., Bush, D.: Using satisfaction arguments to enhance i* modelling of an air traffic management system. In: Proceedings of 15th IEEE International Requirements Engineering Conference. pp. 49–52 (2007)
33. Mayer, S., Wilde, E., Michahelles, F.: A connective fabric for bridging internet of things silos. In: Proceedings of 5th International Conference on the Internet of Things. pp. 148–154 (2015)
34. Mellegård, N., Staron, M.: Improving efficiency of change impact assessment using graphical requirement specifications: An experiment. In: Proceedings of 11th International Conference on Product-Focused Software Process Improvement. pp. 336–350 (2010)

35. Merabet, M., Benslimane, S.M.: A multi-objective hybrid particle swarm optimization-based service identification. In: Proceedings of 1st International Conference on Advanced Aspects of Software Engineering. pp. 52–62 (2014)
36. Mylopoulos, J., Chung, L., Nixon, B.A.: Representing and using nonfunctional requirements: A process-oriented approach. IEEE Transactions on Software Engineering 18(6), 483–497 (1992)
37. Nuseibeh, B., Easterbrook, S.M.: Requirements engineering: a roadmap. In: Proceedings of 22nd International Conference on on Software Engineering. pp. 35–46 (2000)
38. Papazoglou, M.P., van den Heuvel, W., Mascolo, J.E.: A reference architecture and knowledge-based structures for smart manufacturing networks. IEEE Software 32(3), 61–69 (2015)
39. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. IEEE Computer 40(11), 38–45 (2007)
40. Pourshahid, A., Amyot, D., Peyton, L., Ghanavati, S., Chen, P., Weiss, M., Forster, A.J.: Business process management with the user requirements notation. Electronic Commerce Research 9(4), 269–316 (2009)
41. Raman, A., Bharadwaj, S.S., Mukherjee, J.: Developing soa-enabled service agility capabilities: case studies in services industry. International Journal of Business Information Systems 27(1), 21–44 (2018)
42. Rathfelder, C., Groenda, H.: isoamm: An independent SOA maturity model. In: Proceedings of 8th International Conference Distributed Applications and Interoperable Systems. pp. 1–15 (2008)
43. Salles, G.M.B., Fantinato, M., Barros, V.A., de Albuquerque, J.P.: Evaluation of the strali-bpm approach: strategic alignment with BPM using agreements in different levels. International Journal of Business Information Systems 27(4), 433–465 (2018)
44. Schelp, J., Aier, S.: SOA and EA - sustainable contributions for increasing corporate agility. In: Proceedings of 42nd Hawaii International International Conference on Systems Science. pp. 1–8 (2009)
45. Söderström, E., Meier, F.: Combined SOA maturity model (CSOAMM): towards a guide for SOA adoption. In: Proceedings of the 3th International Conference on Interoperability for Enterprise Software and Applications. pp. 389–400 (2007)
46. Stephan, B., Bauer, T., Reichert, M.: Bridging the gap between business process models and service composition specifications. In: Service Life Cycle Tools and Technologies: Methods, Trends and Advances, pp. 124–153 (2011)
47. Strode, D.E.: A dependency taxonomy for agile software development projects. Information Systems Frontiers 18(1), 23–46 (2016)
48. Sutcliffe, A.G., Maiden, N.A.M., Minocha, S., Manuel, D.: Supporting scenario-based requirements engineering. IEEE Transactions on Software Engineering 24(12), 1072–1088 (1998)
49. Vanderfeesten, I.T.P., Reijers, H.A., van der Aalst, W.M.P.: Evaluating workflow process designs using cohesion and coupling metrics. Computers in Industry 59(5), 420–437 (2008)
50. Vilela, J., Castro, J., Martins, L.E.G., Gorschek, T., Silva, C.T.L.L.: Specifying safety requirements with GORE languages. In: Proceedings of the 31st Brazilian Symposium on Software Engineering. pp. 154–163 (2017)
51. Wang, J., Wang, Q.: Analyzing and predicting software integration bugs using network analysis on requirements dependency network. Requirements Engineering 21(2), 161–184 (2016)
52. Wetzstein, B., Leitner, P., Rosenberg, F., Dustdar, S., Leymann, F.: Identifying influential factors of business process performance using dependency analysis. Enterprise Information Systems 5(1), 79–98 (2011)
53. Xu, L.D., Viriyasitavat, W., Ruchikachorn, P., Martin, A.: Using propositional logic for requirements verification of service workflow. IEEE Transactions on Industrial Informatics 8(3), 639–646 (2012)
54. Yao, J., Tan, W., Nepal, S., Chen, S., Zhang, J., Roure, D.D., Goble, C.A.: Reputationnet: Reputation-based service recommendation for e-science. IEEE Transactions on Services Computing 8(3), 439–452 (2015)

55. Yu, E.S.K.: Towards modeling and reasoning support for early-phase requirements engineering. In: Proceedings of 3rd IEEE International Symposium on Requirements Engineering. pp. 226–235 (1997)

Appendix A. Process fragment

The result of process fragments are shown below from Fig. 1 to 8.

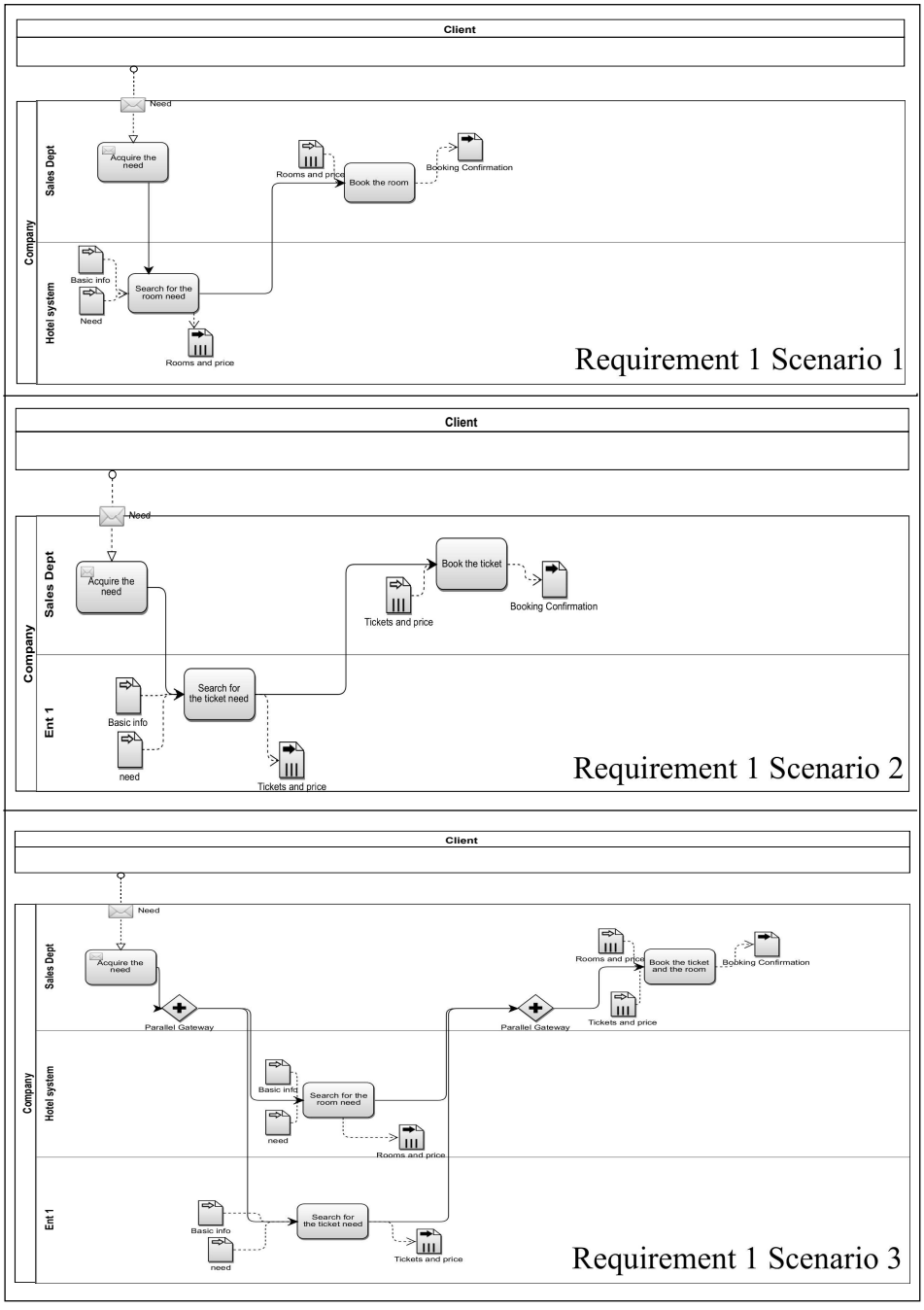


Fig. 1. Requirement R1.

Fig. 2. Requirement R2.

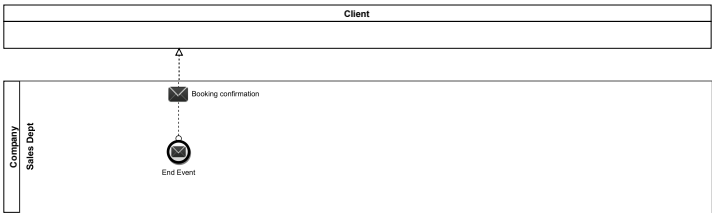


Fig. 3. Requirement R3.

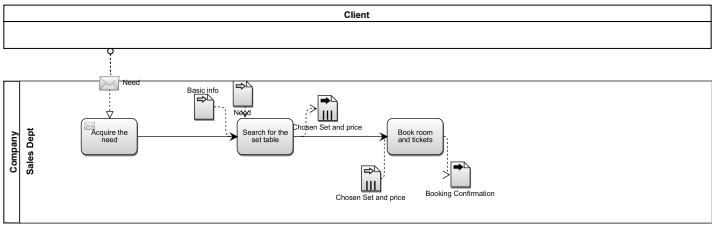


Fig. 4. Requirement R4.

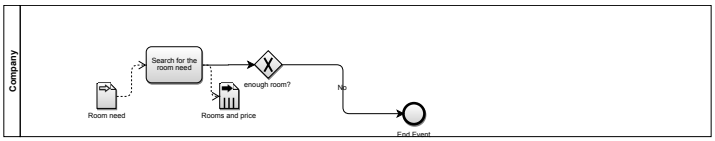


Fig. 5. Requirement R5.

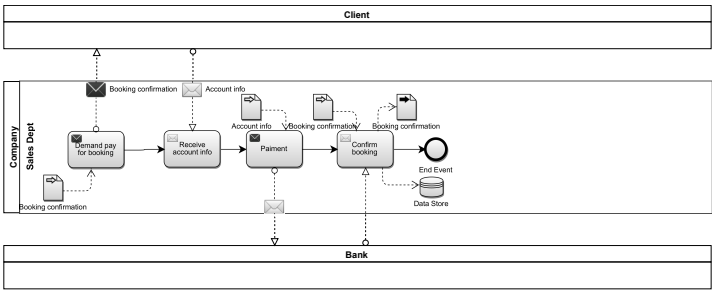


Fig. 6. Requirement R6.

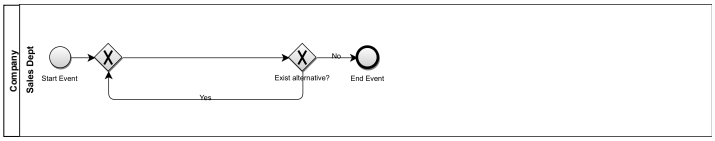


Fig. 7. Requirement R7.

Fig. 8. Requirement R8.

Jiawei Li received her MSc degree from Sino-French Engineering School at Beihang Universit in 2016. Her research interest covers service oriented computing, software engineering and information systems.

Wenge Rong received the B.Sc. degree from the Nanjing University of Science and Technology, China, in 1996, the M.Sc. degree from Queen Mary College, U.K., in 2003, and the Ph.D. degree from the University of Reading, U.K., in 2010. He is currently an Associate Professor with Beihang University, China. He has many years of working experience as a Senior Software Engineer in numerous research projects and commercial software products. His current research interests include machine learning, natural language processing, and information management.

Chuantao Yin received his PhD degree on computer science in 2010 from Ecole Centrale de Lyon. He works as associate professor in Sino-French Engineering School at Beihang University in China. His research activities are focused on human learning, smart learning, smart city, etc.

Zhang Xiong is currently a Professor with the School of Computer Science of Engineering, Beihang University, and the Director of the Advanced Computer Application Research Engineering Center, National Educational Ministry of China. He has published over 200 referred papers in international journals and conference proceedings. His research interests and publications span from smart cities, knowledge management, and information systems. He received the National Science and Technology Progress Award.

Received: February 5, 2018; Accepted: January 15, 2019.