Service Pattern: An Integrated Business Process Model For Modern Service Industry

Jianwei Yin, Zhiling Luo, Ying Li, and Zhaohui Wu

Abstract—Modern Service Industry (MSI) is becoming a leading and pillar industry in recent years. Its theory construction, however, has not kept up with the industry development. Specifically, the business process designing and reconstructing, the critical part in business transformation and competition of Modern Service Enterprise, are still handled manually. The existing models, e.g. BPMN and EPC, mainly adopt the business activities and events without considering the resource and data generated and consumed in interaction with business collaborators, which is ubiquitous in MSI business process. Hence, the deficiency of resource and data in these models hindered them from popularization in MSI. In this paper, with a systematic analysis of the above issues, we introduce the service pattern of MSI business process from the point view of resource and data with formalized description and some concrete basic patterns. To support the process designing and reconstructing better, we propose a pattern-centered formalization language, called SPDL (Service Pattern Description Language). Furthermore, a service pattern matching approach, in the Constructing-As-Identifying style, is studied and a service process designing tool, called SPDL-Editor, is developed. Additionally, a case of the famous video-on-demand service (Youku) in China is given, though out of this paper, for a better understanding of our theories.

Index Terms—service pattern, modern service industry, business process, process design, process reconstruction

1 Introduction

Business process model, depicting the critical enterprise business with a directed graph, underlies many complex applications such as process designing, reconstructing, reusing, log analysing, frequent process discovery and runtime monitoring [1]–[3]. With the gradual development and revolution in business transactions and its context, the researches on classical business process model, BPM [4] for short, e.g. BPEL (Business Process Execution Language) [5], EPC (Event-driven Process Chains) [6], BPMN (Business Process Modeling Notation) [7] and YAWL (Yet Another Workflow Language) [8], are facing a new challenge. Specifically, Modern Service Industry (MSI), a leading and pillar industry in recent years, cannot adopt classical BPM in its process modeling, because more complex **data and resource exchange** with various collaborators are involved [9]–[11].

The data and resource exchange, denoting the procedure that one participant gives some data or resource to another participant and get some data or resource in return, is common in MSI. For example, *Coco-Cola Inc.* puts its advertisements on the web site of *Facebook Inc.*, namely *http://www.facebook.com*. In this procedure, *Coco-Cola Inc.* gives some money, one kind of resource to *Facebook Inc.* and get the customer traffic, another kind of resource, from *http://www.facebook.com* in return. That is to say, *Coco-Cola Inc*. exchanges resource (money and customer traffic) with *Facebook Inc.* in this procedure.

The importance of the data and resource exchange in business interaction with collaborators is approved by administrators. The resource and data exchange information, as a critical business secret, is quite important for the enterprise and its competitors. For instance, once a competitor knows who provides the Ad. on the *Facebook Inc.* and what the price is, namely cost per presentation (CPP), the competitor can estimate *Facebook Inc.*'s advertising revenue and make a pertinent business plan.

Though the importance of data and resource exchange is approved by the administrators, classical BPMs fail to provide a practical analysing tool. The reason is that these models focus on the process structure and ignore this data and resource exchange among various collaborators. Specifically, in classical BPM, data and resource are both abstracted as the attribute of the entity. The data dependency describes the activity executing order due to the data life cycle. However, the data and resource exchange is hidden behind the activity. Both the process designer and enterprise manager in MSI cannot get more detailed supports from classical BPMs.

In this paper, witnessing the importance in business competition, we introduce the pattern of MSI, called service pattern, supporting the data and resource exchange analysing. Service pattern is the abstraction of the business process from three perspectives: workflow, data and resource. Though workflow pattern [12], [13], process data [14] and resource pattern [15], [16] are studied in different domains, there is still a great gap between these models. We integrate these three perspectives in service pattern, and propose a pattern-centered formalization language, called SPDL (Service Pattern Description Language). The concrete concept definitions and deducing rules, presented by type theory [17] and λ-calculus, are both discussed. To better support business analysing, we introduce a service pattern matching approach in the Constructing-As-Identifying style [18]. We convert the matching into the type identification in Coq[1], a type theory based theory proving tool [19]. These formalization definitions build a solid foundation and

Jianwei Yin, Zhiling Luo, Ying Li and Zhaohui Wu are both with the College of Computer Science, Zhejiang University, Hangzhou, 310027, China.
Ying Li is the corresponding author and his e-mail is cnliying@zju.edu.cn.

1. https://coq.inria.fr

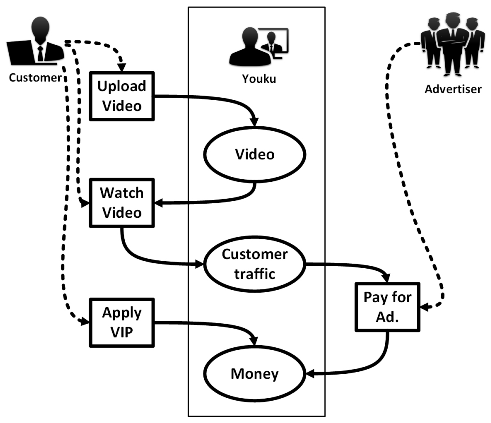


Fig. 1: The resource flow diagram of Youku. From Youku's point view, there are three resources: video, customer traffic and money.

ensure the correctness of business analysing and deducing result for our techniques. Besides, we develop a business process modeling tool, called SPDL-Editor[2], implementing our theories and techniques.

Our major contributions in this paper are summarized as follows:

- an integrated business process model, called service pattern, from three perspectives: workflow, data and resource;
- a pattern-centered formalization language, called Service Pattern Description Language, supporting business process designing and business analysing on data and resource exchange;
- and an approach to match service pattern and service process in the Proving-As-Identifying style.

The rest of this paper is organized as follows. Section 2 introduces the motivation example. Section 3 discusses the problem definition and the overall framework Section 4 briefly introduces the preliminaries including λ-calculus, Type Theory and some notations. Section 5 illustrates SPDL and concrete formalization. Section 6 presents the definition of service pattern. Section 7 studies the problem of service pattern matching (SPM) and introduces our approach. Section 8 briefly introduces the tools supporting SPDL and our service pattern matching approach. We discuss the effect and efficiency of our approach in Section 9. Finally, Section 10 discusses related work and Section 11 concludes this paper.

2 Motivation Case

To study the unfit of classical BPMs in supporting resource and data exchange, we discuss the example of Youku, a typical Modern Service Enterprise in China. The on-line service, youku.com, is the biggest video watching and sharing web site now, which was founded in 2006 and has attracted 150 million customers per day since Apr. 2013. From the perspective of classical BPMs, Youku's core business comprises four basic aspects, which are extracted into following process fragments.

2. http://ccnt.zju.edu.cn/spdl

- **Process 1:** A customer can upload his/her video in youku.com. Then, this video can be stored and exhibited in this web site.
- **Process 2:** A customer can explore the web site and select a video to watch. The video can be played directly if the customer is VIP, otherwise some advertisements must be watched at first.
- **Process 3:** The advertiser will pay to Youku for the presentation of its Ad based on the presentation times.
- **Process 4:** A normal customer can apply for VIP by paying some money to Youku.

Outwardly, we do get the information about concrete activities, their orders and even the control conditions from the point view of BPM. Nevertheless, data and resource exchange in these processes cannot presented intuitively. Specifically, from the perspective of Youku there are three kinds of resources involved in Youku's business described as follows and the resource flow are depicted in Fig. 1.

- **Video** is shot or produced by the customer. Youku gets the copyright of the video and gives some user point to the uploader in process 1.
- **Customer traffic**, the core resource for most Internet companies (e.g. Google and Facebook), means the mount of customers visiting a web site per unit time. Youku attract customer and accumulate customer traffic with free video watching service in process 2.
- **Money** is the essential quantitative resource comprises two parts: paid by advertisers in process 3 and paid by customers in process 4. There are two resource exchanges occurred in this resource. On one hand, Youku provides the customer traffic to advertisers through presenting Ad. and gets money in return. This is the most important component of Youku's business model On the other hand, Youku exchange the customer traffic with the customers themselves in process 4. Once a customer apply to VIP, his/her traffic on this web site cannot be used to exchange with other resource, e.g. the money of advertisers, by Youku.

3 Problem Definition and Solution Framework

Motivated by the scenario mentioned before, we witness that the data and resource exchange is a critical aspect in business process management. However, none of classical BPM covers it very well. Therefore the problem we'd like to address is how to analyze the data and resource exchange in a business process.

Problem: Given a group of processes $t_1, t_2, ..., t_M$, finding out a service element set $\mathcal{E} = Activity|Gateway|Event|Actor|Data|Resource$, a connector set $\mathcal{L}$ and a group of map $\mathcal{F}$ where $t_i = \mathcal{F}_i(\mathcal{E}, \mathcal{L})$, such that any new process t_j, can be easily construceted as $\mathcal{F}_j(\mathcal{E}, \mathcal{L})$.

As illustrated in Fig. 2, our solution framework consists of four layers.

1) The first layer is the theory foundation. We employ λ-calculus and type theory as our formalization basic framework, which are studied in section 4.

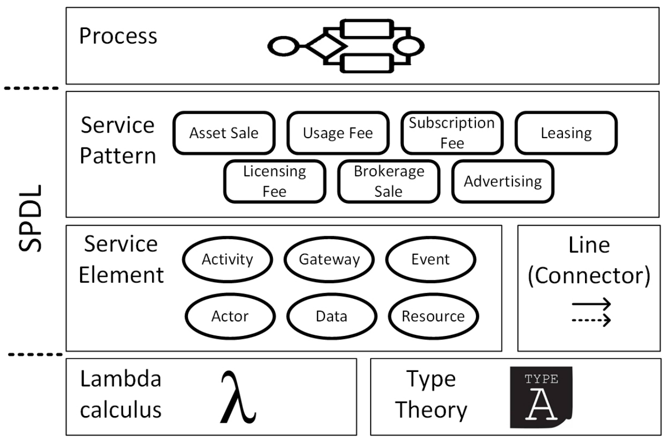


Fig. 2: The overall solution framework.

2) In second layer, we provide the definition of service element $\mathcal{E}$ and connector $\mathcal{L}$.
3) The third layer is the major contribution of our work. We provide the formalization of service pattern, a group of basic service patterns and the composition rule to generate new service pattern.
4) In the top layer, the data and resource exchange in the process t can be studied using the service pattern.

4 Preliminary

Before discussing our theory, we briefly introduce two preliminary theories in this section. We employ the concept about term and some useful logical rules from λ-calculus. For better describing the operations on data and resources in a semantic way , we employ the type construction way from Type Theoery (TT). Also, we summarize the notations used in this paper.

4.1 λ-calculus

λ-calculus is a formal system in mathematical logic and computer science for expressing computation based on function abstraction and application using variable binding and substitution. The complexity of the related theory about λ-calculus prevent us to provide a completed introduction. Therefore we only three basic concepts.

4.1.1 Lambda term

$\lambda x.x^2+1$ is a simple *lambda term* expressing a function using x as the input and outputting x^2+1. In this expression, x is the variable and this term is a *lambda abstraction* on x.

$\nu := \lambda x : nat.x^2 + 1$ is a *typed lambda term*, in which, x is an instance of nat (means nature number). The type of ν is $nat \to nat$.

Though used variously in different domain, lambda term is used to present the operation in our theory.

4.1.2 Beta reduction

A β *reduction* provides a way to replace the variable in lambda term to a concreted case. The simple example is

$$\begin{aligned}\nu \quad 6 &= (\lambda x : nat.x^2 + 1)\, 6\\ &= 6^2 + 1 = 37.\end{aligned}$$

TABLE 1: Logic connectives and type connectives

Logic	Type
$A \wedge B$	Product type ($\times$)
$A \vee B$	Sum type ($+$)
$A \to B$	Function type ($\prod(A, (x)B)$)
$\forall(x : A)B(x)$	Dependent Product type ($\prod(x : A)B(x)$)
$\exists(x : A)B(x)$	Dependent Sum type ($\sum(x : A)B(x)$)
$False$	Empty type ($\perp$)
$True$	Unit type ($\top$)

The β reduction provides a clear way to describe the relation between operation and concreted object. In this example, the lambda term nu is the operation and 6 the concreted object. With the help of TT, we convert this feature to the semantic expression in our SPDL.

4.1.3 Cartesian Product

A *Cartesian Product* is a dependent type.

$$V := \nu_1, \nu_2, ..., \nu_n : \prod x : nat.x^2 + 1 \tag{1}$$

In formula (1), V is defined as the a cartesian production. For any instance $\nu_i : V$, we have $\nu_i : \lambda x : nat.x^2 + 1$.

4.2 Type Theory

Type theory (TT for short) is a logic system used as the foundation of mathematics and computing [17], [20]. It provides computer scientists with a framework to combine logic and computer system design in an elegant and flexible way. Comparing to other formal method theory, TT has following advantages.

- **Semantic Expression.** The semantic information is present in the type of each element. And it makes the expression more similar to human language. For example, the definition of reading book is $read \quad book : Activity$ where $read : Object \to Activity$ and $book : Object$.
- **Curry-Howard isomorphism.** A proposition (e.g. a service process matches a service pattern) can be proved by constructing an instance for this proposition type. If there is an instance can be constructed, the proposition is true, and false versa. This is a called **Constructing-As-Identifying** style [18].

In TT, like most formalized framework, type can be constructed by existing types (such as nat, means nature type) and type connectives (such as $\prod$, $\sum$) recursively. The corresponding of type connectives and normal logic connectives are shown in Table 1. Table 2 shows some notations and Table 3 the logic rules used in the rest of this paper.

5 Service Element in SPDL

The important service elements and business rules in SPDL are formalized in this section by TT. Some notations used in this section can be found in Tab. 2. The paragraph started with **Example** means the example of corresponding definition in Youku. All service elements are defined in the service context E and the business rules are defined in the system context Γ.

TABLE 2: Some extended notations in this paper

Notation	Meaning
$B := C$	B can be written as C
$c : C$	c is an instance of type C
$\nu[\![A]\!]$	A is the explanation of ν and A is the name used in Coq
$b \in_{list} B$	b is an element of list B
$A \subseteq_{list} B$	Any elements in list A are also in list B
$\sigma_B(a)$	The attribute a (the parameter in construction of B) in B
$A :: B$	Connecting A and B to a list
nil	The empty list
$F[\beta]$	Term F contains β as a variable.

TABLE 3: Some logic rules used in this paper

Rule name	Meaning	Expression
Intro Rule	Introduce a variable from context	$\dfrac{\Gamma, a : A \vdash}{\Gamma \vdash a : A}$
Extract Rule	Extract a dependent product type	$\dfrac{\Gamma, a_1 : A \vdash f(a_1) : F}{\Gamma \vdash f : \sum(a : A).F}$
Apply Rule	Apply a function on a variable	$\dfrac{\Gamma, a : A \vdash t : A \to B}{\Gamma \vdash t\, a : B}$
Product Rule	Construct a product type	$\dfrac{\Gamma \vdash a : A \Gamma \vdash b : B}{\Gamma \vdash a \times b : A \times B}$

5.1 Resource

We classify the resources into two types: **money** and **value object**. Because money is easily quantitative and it reflects the profitability of the business process directly. The goods, equipments and fields are value objects which lack a unified quantitative metric. It means that except money, other object which is valuable in business process belongs to value object. We can concentrate on their life cycle in the process and it helps to enrich the semantic information of activities. The formalization of resource in *Backus-Naur-Form* (BNF) is depicted in (2).

$$Resource := Money|(\lambda id_r : nat.ValueObject) \quad (2)$$

The function $\lambda id_r : nat.ValueObject[id_r]$ in typed λ calculus form, constructor of the value object with the parameter id_r , helps to define value object. The nat is short for nature number. Formula (2) means a $resource$ instance can either be an instance of $money$ type or an instance of dependent type, called $ValueObject$.

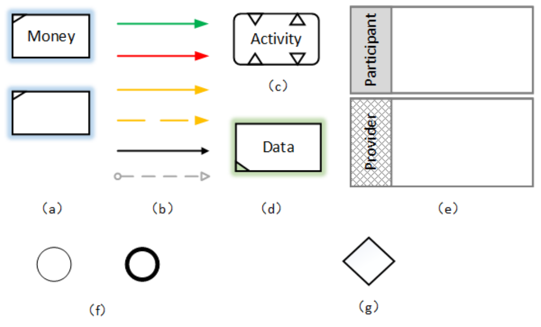


Fig. 3: The notations of SPDL diagram.

Example: *The video, defined by* (3)*, is an important kind of value object (1431 is the global id).*

$$Video := (\lambda id_r : nat.ValueObject) \quad 1431 \quad (3)$$

The global id is a nature number used in many examples follows. The notations in Fig. 3 (a) are **money** and **value object**, from top to bottom, resp. There are four basic operations on the resource: creating, using, exclusively using and consuming.

- **Creating**: generating a resource instance from an activity.
- **Using**: using a resource instance in an activity.
- **Exclusively using**: using a resource instance in an activity, but forbid using from other actor instances at the same time.
- **Consuming**: using up a resource instance in an activity.

The operations (creating, using, exclusively using and consuming) on resource belong to the type of a *Cartesian Product*:

$$\begin{aligned} f_{Op_r} :=& \nu_{cr}[\![Create_r]\!]|\nu_{ur}[\![Use_r]\!]|\nu_{er}[\![ExclusiveUse_r]\!] \\ &|\nu_{or}[\![Consume_r]\!] : \prod r : Resource.Op_r(r). \end{aligned} \quad (4)$$

In (4), ν presents an operation, the r in subscript means resource.

Example: *Watching a video can be expressed as follows. In* (5)*, Video is a kind of resource.*

$$WatchVideo := \nu_{ur} \quad Video \quad (5)$$

5.2 Data

Data is the attribute of the entity in the business process. The formalization of data in BNF is present in (6).

$$Data := \lambda id_d : nat.\mu_d[\![DataConstructor]\!] \quad (6)$$

In (6), μ declares a constructor, the subscript, and d, means constructing a data type. The notation in Fig. 3 (d) is data. Differing from the operation on resource, there are only three operations on data (creating, consuming and using). The data cannot be exclusively used because different actor instances can read the same data at the same time. This is the most important difference between data and resource. These operations are also in the type of *Cartesian Product*.

$$\begin{aligned} f_{Op_d} :=& \nu_{cd}[\![Create_d]\!]|\nu_{ud}[\![Use_d]\!]|\nu_{od}[\![Consume_d]\!] \\ &: \prod d : Data.Op_d(d) \end{aligned} \quad (7)$$

Example: *The identification of customer is an important data. A customer can be either a normal one or a VIP. The global id of VIP is 2432 and Normal 2433.*

$$VIP := \mu_d \quad 2432 \qquad Normal := \quad \mu_d \quad 2433 \quad (8)$$

5.3 Actor

The actor is also called participant or collaborator in some process models. While in SPDL, we use the concept of actor to include the service provider itself. There are two kinds of actor: the **service provider** and the **participant**. The service provider, as a special actor, means the one who owns the process. In the example of Youku, the service provider is Youku itself. The actor is formalized as follows:

$$Actor := Provider|(\lambda id_a : nat.Participant) \tag{9}$$

The actor is present in Fig.3 (e) with **Participant** on the top and **Provider** on the bottom.

Example: *The customer of Youku, the id is 3104, can be defined as follows.*

$$Customer := (\lambda id_a : nat.Participant)3104 \tag{10}$$

5.4 Event

There are two basic events in SPDL, start event and end event. In Fig. 3 (f), the left is the start event and right the end event. To support more events in BPMN, we provide a constructor with an id as a parameter and the formalization of event is present in (11).

$$Event := Start|End|(\lambda id_e : nat.\mu_e[\![EventIn]\!]) \tag{11}$$

5.5 Gateway

We have defined a basic gateway, the exc, namely the exclusive gateway (XOR), and more gateways can be defined by the constructor, $GatewayIn$. The notation of exc is presented in Fig. 3 (g).

$$Gateway := Exc|(\lambda id_g.\mu_g[\![GatewayIn]\!]) \tag{12}$$

5.6 Line

Line is also called connector in some models. There are 6 kinds of line, as Fig. 3 (b) from top to bottom.

- **Creating line**: this line connects from an activity to a resource (or data). It means the connected resource (or data) is generated by the connected activity.
- **Consuming line**: this line connects from an activity to a resource (or data). It means the connected resource (or data) is used up by the connected activity.
- **Using line**: this line connects from an activity to a resource (or data). It means the connected resource (or data) is used by the connected activity.
- **Exclusively using line**: this line connects from an activity to a resource. It means the connected resource is used by the connected activity and at the same time this resource cannot be used by other actors.
- **Sequence line**: this line connects from an activity to another one. It means the latter activity cannot be executed until the former one is finished and these two activities belong to the same actor.
- **Message line**: this line connects from an activity to another one. It means the latter activity cannot be executed until the former one is finished and these two activities do not belong to the same actor.

The first four lines are used to present the data flow and resource flow. The last two lines, present the work flow, are formalized as follows:

$$\begin{aligned} LineP :=& \lambda lp_a : Activity.\mu_{lpa}[\![LinePA]\!] \\ & |\lambda lp_e : Event.\mu_{lpe}[\![LinePE]\!] \\ & |\lambda lp_g : Gateway.\mu_{lpg}[\![LinePG]\!] \\ Line :=& \lambda l_s : LineP.\lambda l_t : LineP.\mu_l[\![LineIn]\!] \end{aligned} \tag{13}$$

In (13), $LineP$ is the port and $Line$ is the combination of two ports (source port and target port).

5.7 Activity

The activity notation is present in Fig. 3 (c). And the formalization of activity is shown in (14):

$$\begin{aligned} Activity :=& \lambda a_p : Actor.\lambda a_r : list(Op_r). \\ & \lambda a_d : list(Op_d). \\ & \mu_a[\![ActivityConstructor]\!] \end{aligned} \tag{14}$$

The first parameter is the actor, the second is the resource operation list (including four kinds) and the third is the data operation list (including three kinds).

Example: *Uploading video is an activity which can be expressed by* (15). *A simple explanation of this formula is that the activity named upload is executed by a customer and creates an instance of video.*

$$Upload := \mu_a(Customer\,(\nu_{cr}Video :: nil)\,nil) \tag{15}$$

5.8 Activity Pattern

Activity Pattern is an λ-abstraction of the activity, as depicted in (16).

$$\begin{aligned} ActivityPattern :=& \lambda ap_p : Actor.\lambda ap_r : list(Op_r). \\ & \lambda ap_d : list(Op_d). \\ & \mu_{ap}[\![ActivityPatternConstructor]\!] \end{aligned} \tag{16}$$

Definition (Activity Pattern rule): An activity a matches an activity Pattern ap, written as $ap \preceq a$, if and only if they have the same actor and the latter's list of resource operations and data operations are sub-list of the former's, resp.

$$\begin{aligned} ap \preceq a :=& (\sigma_{ap}(ap_p)) = (\sigma_a(a_p)) \\ & \times (\sigma_{ap}(ap_r)) \subseteq_{list} (\sigma_a(a_r)) \\ & \times (\sigma_{ap}(ap_d)) \subseteq_{list} (\sigma_a(a_d)) \end{aligned} \tag{17}$$

5.9 Business Rule

The business rules are defined in the type of **dependent type**. Some useful business rules are present in (18):

The $ConnectAAProp$ is the proposition checking whether l connects from a_1 to a_2. The $ConnectAEProp$ and $ConnectEAProp$ check the connection from activity to event and event to activity. The $ActivityRProp$ is the proposition checking whether a involves r as a parameter. In the similar way, $ActivityDProp$ are defined. The $ActorProp$ checks whether p is the actor of a.

All these propositions defined above are the basic business rules. Notation ψ represents a business rule.

In spite of basic business rule, we define a special rule, named **pattern rule** which means the rule constraints on the pattern. The pattern rule can be constructed by basic business rules while it helps to design a new pattern rule. Because of limit of the paper volume, we will not discuss the equation of business rule and pattern rule, and the concrete method of designing a new pattern by pattern rules.

$$\begin{aligned}
\psi_{CAA}[\![ConnectAAProp]\!] &:= (\sigma_l(l_s) = \sigma_{lpa}(a_1))\times \\
&(\sigma_l(l_t) = \sigma_{lpa}(a_2)) : \prod(l : Line) \\
&\prod(a_1, a_2 : Activity)Prop(l, a_1, a_2) \\
\psi_{CAE}[\![ConnectAEProp]\!] &:= (\sigma_l(l_s) = \sigma_{lpa}(a))\times \\
&(\sigma_l(l_t) = \sigma_{lpe}(e)) : \prod(l : Line) \\
&\prod(a : Activity)\prod(e : Event)Prop(l, a, e) \\
\psi_{CEA}[\![ConnectEAProp]\!] &:= (\sigma_l(l_s) = \sigma_{lpe}(e))\times \\
&(\sigma_l(l_t) = \sigma_{lpa}(a)) : \prod(l : Line) \\
&\prod(e : Event)\prod(a : Activity)Prop(l, e, a) \\
\psi_{AR}[\![ActivityRProp]\!] &:= (\sigma_a(a_r) \ni_{list} r) \\
&: \prod(a : Activity)\prod(r : Op_r)Prop(a, r) \\
\psi_{AD}[\![ActivityDProp]\!] &:= (\sigma_a(a_d) \ni_{list} d) \\
&: \prod(a : Activity)\prod(d : Op_d)Prop(a, d) \\
\psi_{A}[\![ActorProp]\!] &:= (\sigma_a(a_p) = p) \\
&: \prod(a : Activity)\prod(p : Actor)Prop(a, p)
\end{aligned} \tag{18}$$

$$\begin{aligned}
\psi^*_{AP}[\![ActivityPatternProp]\!] &:= ap \preceq a : \prod(a : Activity) \\
&\prod(ap : ActivityPattern)Prop(a, ap)
\end{aligned} \tag{19}$$

At last, we note that all these business rules, including pattern rules and basic rules, are both defined as a dependent type. Here we discuss whether a business condition is true or not in a particular context.

Definition 5.1 (Truth Condition) *In a service context E and the system context Γ, a business rule Ψ is true if there is an instance i in the type of Ψ can be constructed.*

Truth condition can be proved by the **Curry-Howard isomorphism** [18] and we omit it here. An interesting fact is that, the instance of basic business rules can be constructed by the service elements directly and the instance of complex one can be constructed by the existing instances of basic ones. This is the called **Constructing-As-Identifying** style.

5.10 Youku

In this section, we provide the complete modeling details of Youku by SPDL. The SPDL diagram of Youku is presented in Fig. 4.

- In the first process, the customer uploads a video to provider, namely Youku. A corresponding video instance is created for Youku.
- In the second process, the customer browses the web site and watches an Ad. if he/she is the normal customer than he/she watches the video. This process uses the video and the data of customer identification.
- In the last process, the customer applys VIP and pays some money to Youku.

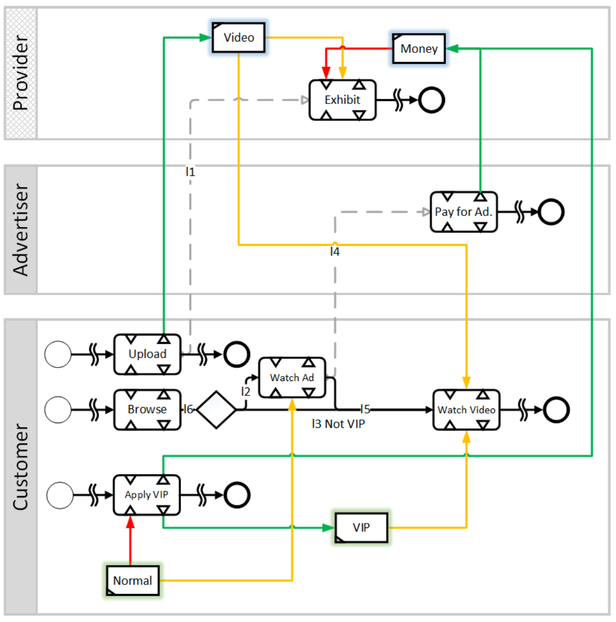


Fig. 4: Youku SPDL diagram: including three swimlanes (namely three actors) and seven activities, two kinds of resources and data.

The types, including different resources, data and actors, in the service context E of Youku are defined at first.

$$\begin{aligned}
Video &:= (\lambda id_r : nat.ValueObject[id_r])1431 \\
VIP &:= \mu_d \quad 2432 \\
Normal &:= \mu_d \quad 2453 \\
Customer &:= (\lambda id_a : nat.Participant[id_a])3104 \\
Advertiser &:= (\lambda id_a : nat.Participant[id_a])3105
\end{aligned} \tag{20}$$

Then the activities (Upload, Exhibit, Browse, WatchAd, WatchVideo, ApplyVIP, PayForAd) are defined.

$$\begin{aligned}
Upload &:= \mu_a(Customer\,(\nu_{cr}Video :: nil)\,nil) \\
Exhibit &:= \mu_a(Provider\,(\nu_{or}Money :: \nu_{ur}Video \\
&\quad :: nil)\,nil) \\
Browse &:= \mu_a(Customer\,nil\,nil) \\
WatchAd &:= \mu_a(Customer\,nil\,(\nu_{ud}Normal :: nil)) \\
WatchVideo &:= \mu_a(Customer\,(\nu_{ur}Video :: nil)\,(\nu_{ud} \\
&\quad VIP :: nil)) \\
ApplyVIP &:= \mu_a(Customer\,(\nu_{cr}Money :: nil) \\
&\quad (\nu_{od}Normal :: \nu_{cd}VIP :: nil)) \\
PayForAd &:= \mu_a(Advertiser\,(\nu_{cr}Money :: nil)\,nil)
\end{aligned} \tag{21}$$

Then there are some lines.

$$\begin{aligned} l1 &:= \mu_l((\mu_{lpa} Upload)\,(\mu_{lpa} Exhibit)) \\ l2 &:= \mu_l((\mu_{lpg} Exc)\,(\mu_{lpa} WatchAd)) \\ l3 &:= \mu_l((\mu_{lpg} Exc)\,(\mu_{lpa} WatchVideo)) \\ l4 &:= \mu_l((\mu_{lpa} WatchAd)\,(\mu_{lpa} PayForAd)) \\ l5 &:= \mu_l((\mu_{lpa} WatchAd)\,(\mu_{lpa} WatchVideo)) \\ l6 &:= \mu_l((\mu_{lpa} Browse)\,(\mu_{lpg} Exc)) \end{aligned} \tag{22}$$

6 Service Pattern

This section introduces the formalization of service pattern and a specific example.

6.1 Definition

The service pattern can be formalized as the combination of the business rules about the activities, resources, data and actors. We specify service pattern Ψ by the following BNF rule.

$$\Psi := \psi | \sum(x : X).\Psi(x) | \prod(x : X).\Psi(x) | \Psi \times \Psi | \Psi + \Psi \tag{23}$$

There are two possible methods, present in Fig. 6, to construct the service pattern.

- **Mining service pattern.** Mining service pattern from service processes is the reasonable and reliable measure. This method, however, relies on the accumulation of service processes (especially in SPDL). Limited to this condition, we leave the study on this method in a near future.
- **Extracting service pattern.** Extracting service pattern from literatures (especially literatures from management science and economics) is a feasible method.

We have extracted 7 basic service patterns, **Asset sale**, **Usage fee**, **Subscription fee**, **Leasing**, **Licensing fee**, **Brokerage fee** and **Advertising** from literature [21] and studied 30 (e.g. Alibaba Inc. [3]) companies [22] in China.

- **Asset sale** is the classical service pattern of traditional manufacturing industry service. There are two important steps: producing something valuable and selling it to earn benefit.
- **Usage fee** is generated by the use of a service. A telecom operator may charge customers for the number of minutes spent on the phone.
- **Subscription fee** is generated by selling continues access to a service. World of Warcraft Online, a Web-based computer game, allows users to play its online game in exchange for a monthly subscription fee.
- **Leasing** created by temporarily granting someone the exclusive right to use a particular asset for a fixed period in return for a fee. Zipcar.com, which allows customers to rent cars by the hour in North American cities, provides a good illustration.
- **Licensing fee** is generated by giving customers permission to use protected intellectual property in exchange for licensing fees? Licensing is common in the media industry, where content owners retain copyright while selling usage licenses to third parties.

3. http://www.alibaba.com

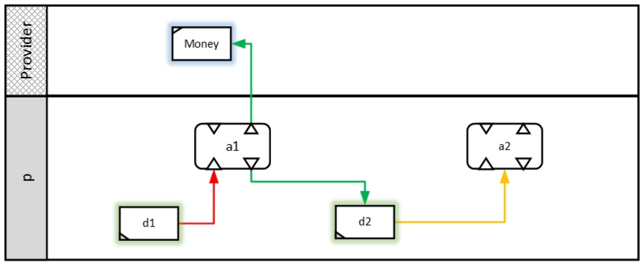


Fig. 5: Subscription fee SPDL diagram: including two swimlanes, two activities, one resource and two data

- **Brokerage fee** derives from intermediation services performed on behalf of two or more parties. APP Store (Apply Inc.) provides a platform for APPs and makes benefit by brokerage.
- **Advertising** results from fees for advertising a particular product, service, or brand. In recent years software and services have started relying more heavily on advertising revenues.

Restricted to the length of this paper, we just discuss **subscription fee** in following parts in detail. An Introduction and comparison of these seven service patterns can be found in Tab. 4. These service patterns are quite basic and the more complex patterns (e.g. the combination of two basic patterns) can also be supported in our model.

6.2 subscription fee

The SPDL diagram (Fig. 5) of subscription fee presents the two actors ($Provider$ and a participant: p), two activities (a_1 and a_2), $Money$ and two data (d_1 and d_2). This pattern consists of two steps:

- Subscribe: p consumes the data d_1, creates the data d_2 and pays some $Money$ in activity a_1.
- Use: p uses the data d_2 in other activity a_2.

These two steps can be represent by a group of business rules, which compose the definition of this service pattern in **business rule form**.

$$\begin{aligned} &\Psi^r_{SF}[\![SubscriptionFee]\!] : \\ &\quad \sum(a_1, a_2 : Activity) \sum(d_1, d_2 : Data) \\ &\quad \sum(p : Participant) \psi_A(a_1\, p) \times \psi_A(a_2\, p) \\ &\quad \times \psi_{AD}(a_1\, \nu_{od}(d_1)) \times \psi_{AR}(a_1\, \nu_{cr}(Money)) \\ &\quad \times \psi_{AD}(a_1\, \nu_{cd}(d_2)) \times \psi_{AD}(a_2\, \nu_{ud}(d_2)). \end{aligned} \tag{24}$$

Here comes the **pattern form** use the pattern rule (ψ_{AP} for example).

$$\begin{aligned} &\Psi^*_{SF}[\![SubscriptionFee]\!] : \\ &\quad \sum(a_1, a_2 : Activity) \sum(d_1, d_2 : Data) \\ &\quad \sum(P : Participant) \psi_{AP}(\mu_{ap}(p\, nil \\ &(\nu_{cr}(Money) :: nil)\,(\nu_{od}(d_1) :: nil)\,(\nu_{cd}(d_2)))\, a_1) \\ &\quad \times \psi_{AP}(\mu_{ap}(p\, nil\, nil\, (\nu_{ud}(d_2) :: nil)\, a_2) \end{aligned} \tag{25}$$

The two forms are equivalent and the proof is omitted here since the length limit of this paper.

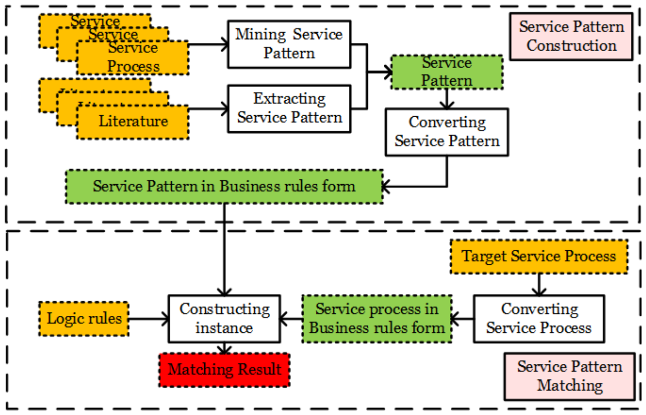

Service
Service
Service Process
Mining Service Pattern
Service Pattern Construction
Service Pattern
Literature
Extracting Service Pattern
Converting Service Pattern
Service Pattern in Business rules form
Target Service Process
Logic rules
Constructing instance
Service process in Business rules form
Converting Service Process
Matching Result
Service Pattern Matching

Fig. 6: Service Pattern Matching Framework: including a service pattern construction phase and a service pattern matching phase

7 Service Pattern Matching

A basic application about the service pattern is the **service pattern matching (SPM)**. The framework of our solution is present in Fig. 6. Detailed explanation can be found in this section.

7.1 Problem Description

Definition 7.1 (Service Pattern Matching) *Considering a service process t and a service pattern Ψ, service pattern matching is to find out whether $t \propto_p \Psi$.*

In this definition, $t \propto_p \Psi$ is satisfied if two conditions hold on the same map $\mathcal{M} : V(t) \to V(\Psi)$ in which $V(t)$ means the activity node set of process t and $E(t)$ is the sequence line set:

1) **Semantic Consistency Condition**: Existing a map $\mathcal{M}$, for any activity node $v \in V(t)$ that v and $\mathcal{M}(v)$ must express the same function.
2) **Ontology Consistency Condition**: Existing a map $\mathcal{M}$, for any sequence line $(v_1, v_2) \in E(t)$ that $(\mathcal{M}(v_1), \mathcal{M}(v_2)) \in E(\Psi)$.

The ontology consistency condition is also called the directed subgraph isomorphism condition, if we treat the process t as the two tuples $(V(t), E(t))$. In this way, we can use the subgraph isomorphism approaches to partially solve SPM. However, the semantic consistency condition is much harder to satisfy, because this condition is not quantitative. In the rest of this section, we propose an approach to solve SPM by converting the activity semantic by its resource operation list and data operation list.

7.2 Typed SPM

As the semantic consistency condition described, the similarity measurement between two activities are very important. A possible measurement is the activity pattern rule defined by (17). The typed SPM is defined as follows:

Definition 7.2 (Typed SPM) $t \propto_p \Psi$ *if existing a map $\mathcal{M}$, for any activity A in t that $A \preceq \mathcal{M}(A)$ and for any line L in t that $\sigma_l(l_s) \preceq \mathcal{M}(\sigma_l(l_s))$ and $\sigma_l(l_t) \preceq \mathcal{M}(\sigma_l(l_t))$.*

We can find that typed SPM is a specification of SPM. And the essential of typed SPM is a composite business rule. So the truth condition of typed SPM is as follows.

Theorem 7.1 (Typed SPM Theorem) *Given a target service process t and a service pattern Ψ, in context E and Γ, $t \propto_p \Psi$ is true if an instance i of Ψ can be constructed.*

The truth condition is sufficient for Typed SPM. The proof is discussed in section 9.1.

7.3 Constructing as Identifying

Constructing as Identifying comes from the **formulae-as-types interpretation** in **Curry-Howard isomorphism**. It provides a framework to construct an instance of a dependent type (including dependent sum type and dependent product type in Table 1) by a group of logic rules in Table 3. Based on this style, our solution framework consists of three steps:

- **Step 1:** Converting the target service pattern into a dependent type. It may in the business rule form (e.g. (24)) or the pattern form (e.g. (25)).
- **Step 2:** Extracting the service elements from ecosystem E into a group of instances of business rules.
- **Step 3:** Trying to construct an instance of dependent type by using logic rules on the existing instances. If succeed, the service process matches such service pattern. If failed, the service process doesn't match this service pattern.

As Fig. 6 presents, the service patterns defined in Section 6 are already in business rule form. New service pattern can be extracted or mined from existing service processes. Before matching, new service pattern must be converted into business rule form.

Example: *This example is identifying whether Youku matches subscription fee pattern. Subscription fee pattern is defined in business rule form in* (24). *So we just need to complete step 2 and 3. In step 2, we take the 'applyVIP' as an example and the extracting details is as* (26).

$$\frac{E \vdash Customer : Actor \quad \text{(Definition (20))} \qquad E \vdash ApplyVIP : Activity \quad \text{(Activities definition (21))}}{\Gamma\, E \vdash \sigma_{ApplyVIP}(a_p) = Customer : \psi_A(ApplyVIP\, Customer)} \ (\text{Intro } \psi_A) \tag{26}$$

In the similar way, we can extract other propositions.

$$\begin{aligned}\Gamma\, E \vdash \sigma_{WatchVideo}(a_p) &= Customer \\ &: \psi_A(WatchVideo\, Customer)\end{aligned} \tag{27}$$

$$\frac{E \vdash Normal : Data \quad \text{(Eco-system definition (20))} \qquad E \vdash ApplyVIP : Activity \quad \text{(Activities definition (21))}}{\Gamma\, E \vdash (\sigma_{ApplyVIP}(a_i d) \ni_{list} \nu_{od}(Normal)) : \psi_{ID}(ApplyVIP\, \nu_{od}(Normal))} \ (\text{Intro } \psi_{ID}) \tag{28}$$

similarly, we can get following equations.

$$\begin{aligned}\Gamma\, E \vdash (\sigma_{ApplyVIP}(a_o r) \ni_{list} \nu_{cr}(Money))\\ : \psi_{OR}(ApplyVIP\, \nu_{cr}(Money))\\ \Gamma\, E \vdash (\sigma_{ApplyVIP}(a_o d) \ni_{list} \nu_{cd}(VIP))\\ : \psi_{OD}(ApplyVIP\, \nu_{cd}(VIP))\\ \Gamma\, E \vdash (\sigma_{WatchVideo}(a_i d) \ni_{list} \nu_{ud}(VIP))\\ : \psi_{ID}(WatchVideo\, \nu_{ud}(VIP))\end{aligned} \tag{29}$$

In step 3, we extract $ApplyVIP$, $WatchVideo$, $Customer$ from (26), (27), (28) *and* (29)*. As illustrated in Tab. 3, the extract rule finds out the dependent product type from the λ term. In detail, since we get the type of $\sigma_{ApplyVIP}(a_o r) \ni_{list} \nu_{cr}(Money)$, $\sigma_{ApplyVIP}(a_o d) \ni_{list} \nu_{cd}(VIP)$, and $\sigma_{WatchVideo}(a_i d) \ni_{list} \nu_{ud}(VIP)$, we can extract the process as a dependent product type.*

$$\begin{aligned}\Gamma\, E \vdash (\sigma_{ApplyVIP}(a_p) = Customer)\\ \times(\sigma_{WatchVideo}(a_p) = Customer)\\ \times(\sigma_{ApplyVIP}(a_i d) \ni_{list} \nu_{od}(Normal))\\ \times(\sigma_{ApplyVIP}(a_o r) \ni_{list} \nu_{cr}(Money))\\ \times(\sigma_{ApplyVIP}(a_o d) \ni_{list} \nu_{cd}(VIP))\\ \times(\sigma_{WatchVideo}(a_i d) \ni_{list} \nu_{ud}(VIP)) : \Psi^{r}_{SF}\end{aligned} \tag{30}$$

In (30)*, the item $(\sigma_{ApplyVIP}(a_p)......\nu_{ud}(VIP))$ is the instance of Ψ^{r}_{SF}. Therefore we can determine that Youku matches the service pattern Ψ^{r}_{SF}.*

8 Tools

To support our theories, we have built some tools, including an eclipse based visual process modeling software, named **SPDL-Editor** which implements SPDL. We also provide a SPDL lib for *Coq* and a package for *Microsoft Visio*.

8.1 SPDL-Editor

Fig. 7 is the snapshot of SPDL-Editor. It now supports following functions.

- Creating, Editing and Saving the service process with resource.
- Matching a service process automatically with existing 7 service patterns (in section 6).

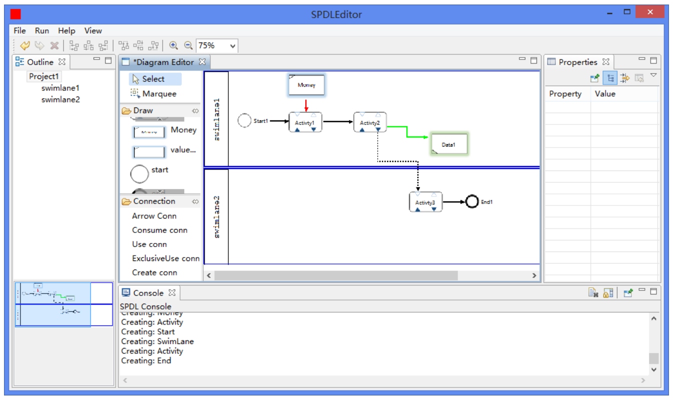
More technology details can be found in the technical report [23].

8.2 Other tools

A SPDL lib for Coq is specially mentioned here. Coq is a proving assistant tool in the logic framework of TT. We have developed a tool (SPDL4Coq) to convert a SPDL model into a Coq script. With the help of SPDL lib and the Coq script, SPM can be solved in Coq automatically.

9 Discussion

In this section, we discuss the effectiveness and efficiency of our approach in solving SPM.



```
Creating: Money
Creating: Activity
Creating: Start
Creating: SwimLane
Creating: Activity
Creating: End
```

Fig. 7: Snapshot of SPDL-Editor: including the editor area, thumbnail, console and outline.

9.1 Effectiveness

Let's consider the typed SPM theorem mentioned above.

Theorem 9.1 (Typed SPM Theorem) *Given a target service process t and a service pattern Ψ, in context E and Γ, $t \propto_p \Psi$ is true if an instance i of Ψ can be constructed.*

Proof 1 *To prove typed SPM theorem, we need to prove the existence of a map $\mathcal{M}$ on two consistency conditions. We use $A_1(i), A_2(i), ..., A_{n_i}(i)$ to represent the activities in instance i and $A_1(t), A_2(t), ..., A_{n_t}(t)$ to represent the activities in process t. The service pattern rule ensures that for any activity $A_j(i)$ in instance i, there is an activity $A_k(t)$ in process t that $A_j(i) \preceq A_k(t)$. In this way, we can construct a map $\mathcal{M}' = \{(A_k(t), A_j(i)) | \forall k \in [1, n_i]\}$. Therefore, map $\mathcal{M}'$ satisfies semantic consistency condition. If $\mathcal{M}'$ satisfies ontology consistency condition, then it is the wanted map. Considering a random line l in the instance i, $\sigma_l(l_s) \preceq \mathcal{M}(\sigma_l(l_s))$ and $\sigma_l(l_t) \preceq \mathcal{M}(\sigma_l(l_t))$, because of ψ_{CAA}. Therefore, $\mathcal{M}'$ satisfies ontology consistency condition.**QED.***

Typed SPM Theorem ensures that typed SPM, a possible specification of SPM, can be solved by constructing an instance of the service pattern.

Another possible specification is ignoring the semantic consistency condition and degrading the original problem into a subgraph isomorphism problem. pThe research on subgraph isomorphism has a long history and the detailed comparison is discussed in section 10.

9.2 Efficiency

The efficiency is the shortcoming of most theorem-proving methods. Our approach has a high time complexity in finding a possible abstraction for each activity in service pattern. Considering a service pattern with n activities and a service process with m activities. The number of possible matching is as large as m^n. To increase the efficiency, the following improvements can be made.

- **Appending subtypes for activities (Subtyped SPM).** Assuming that n service pattern activities are classified as k subtypes, each subtype has n_i service pattern activities and m_i process activities in which $i \in [1, k]$, $\sum_i n_i = n$ and $\sum_i m_i = m$.

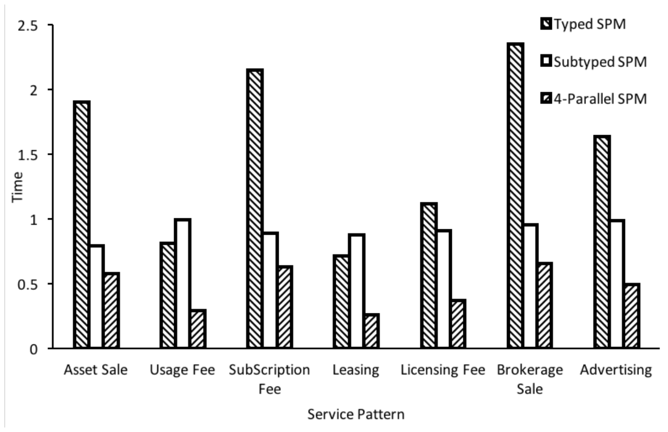


Fig. 8: The evaluation on different patterns with different approaches.

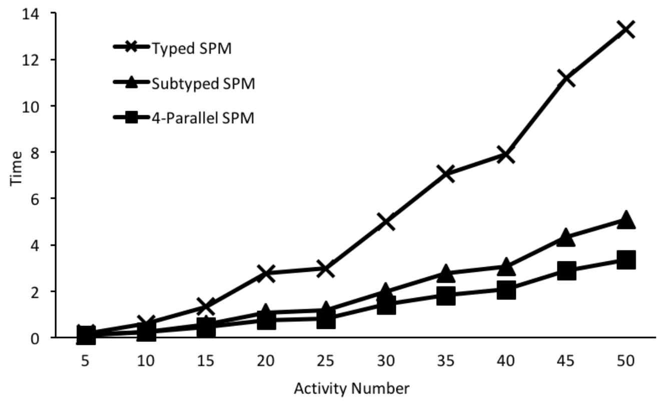


Fig. 9: The evaluation on different activity number with different approaches.

The number of possible matching is $\prod_i^k m_i^{n_i}$ which is much smaller than m^n. The classification basis is the activity behavior, which can be specified by the data and resource exchange. For example, we can put *WatchAd* and *WatchVideo*, two activities in *Youku* (Fig. 4), in the same group because they both have other data and resource exchange except using a kind of data. *WatchAd* uses the data *Normal*, and *WatchVideo* uses the data *VIP*.

- **Parallelizing matching (Parallel SPM).** The parallelizing framework (e.g. MapReduce [24]) can improve the efficiency by matching many pattern activities at the same time. In the Maping procedure, each computing node takes responsibility of a service pattern and in reducing procedure, the matching results from each computing node are summarized. Considering a k-parallelize framework, the matches number is $\lceil m/k \rceil^n$ which is also much smaller than m^n. For example, considering the matching of *Youku* with seven known patterns, described in section 6, we can put the matching processing, e.g. section 6.3, in seven computing nodes. In each node, our Typed SPM can be used to identify the relation between the service pattern and *Youku*. At last, in the reducing procedure, we summarize the result from each computing node. In this way, we can speed our method and improve the efficiency in mass activity environment.

9.3 Experiment

To quantitatively evaluate the efficiency of Typed SPM, Subtyped SPM and Parallel SPM, we conduct two experiments. In these experiments, when using Subtyped SPM, we abstract 5 activities as a subtype. And when using Parallel SPM, we deploy the Typed SPM on four independent threads, namely 4-parallel. We record the executing time for each approach.

In the first experiment, we test seven service pattens on same process with Typed SPM, Subtyped SPM and Parallel SPM. The report is illustrated in Fig. 8, in which 4-parallel always performs best for each service pattern. In most situations, except Usage Fee, Leasing and Licensing Fee, Subtyped SPM uses less time comparing with Typed SPM. This is because both Usage Fee, Leasing and Licensing are patterns with only one activity, as Tab. 4. It means that Subtyped SPM is a more appropriate choice when the pattern has more activities.

In the second experiment, we test these approaches on processes with different activity number. As depicted in Fig. 9, the executing time of Typed SPM increases fastest comparing with Subtyped SPM and Parallel SPM. It concludes the Subtyped SPM and Parallel SPM have better scalablity.

10 Related Works

For better understanding the originality and contribution of our works, the classical business process models and workflow patterns and subgraph isomorphism techniques are discussed in this section. These approaches and models, althought achieve a great success of their field, performs poorly in analzying the data and resource exchange in business process.

10.1 Business Process Models

As a basic and practical standard, Business Process Model and Notations, BPMN [7], laid a solid foundation for most researches on this domain. BPMN provides the basic concepts, including activity, gateway and event, which both play important roles in various business process models. The limitation of BPMN, however, comes from its poor abstraction in data and resource.

To present the data relation between different activities, BPMN involves data dependency which helps in process designing. Bhattacharya, et al. [25] stepped further and formulated a model called artifact-centric BPM. Artifact-centric BPM is also called data centric BPM [26] [27] in previous researches. Estanol, et al. [28] used UML to represent each element in artifact-centric model. Richard, et al. [14] provided a brief survey on artifact-centric model. The key idea of artifact-centric BPM is encapsulating the data in business process as an special object called artifact. In the executing procedure, both the data creating and destroying are happened in artifact. The most important improvement is that it helps to access the state of data at any time. In our work, we refer artifact centric BPM to present the life cycle of the data. However, artifact cannot directly be used to support resource aware service process designing, because it does not distinguish the resources with attributes.

There are many research results in business process model including: Event-driven Process Chains [6] organises the process as the event chain. Reijgers et al. [29] discussed the declarative model of business process and proposes a research agenda for the development of modeling approach. Wong et al. [30] provided the process semantics for BPMN. Russell et al. [16] extended BPEL with the human task considered. Rodriguez et al. [31] extended BPMN with data quality considered. Maamar et al. [32] introduced a framework to resolve the resource confliction happened in process. Tzivikou et al. [33] proposed a language for modeling business terms, which is named SeDL-C. In our previous works, [34] summarized the works on Artifact-Centric Models and [35] studied the characteristic of Modern Service Industry.

Comparing with these BPMs, SPDL holds a better formalization on data and resource which supporting the analyzing of data and resource exchange in process.

10.2 Workflow Pattern

YAWL [8], [36], proposed by Aalst, is a famous workflow modeling language based on Petri Net. Besides, Aalst studied the workflow pattern [12] and lays a solid foundation for the workflow pattern analysing. In literature [15], [16], Aalst proposed the resource workflow pattern. However, the resource is only limited to the human labor resource. The money and other value objects are not considered in this framework.

10.3 Subgraph Isomorphism

As we mentioned in previous section, subgraph isomorphism can be used to solve the SPM, if we ignore the semantic consistency condition [37]. subgraph is also called subgraph mining, subgraph matching or subgraph discovery in some literatures. FSG(Frequent Subgraphs) [38], AGM(Apriori-based Graph Mining) [39] and gSpan [40] are common used subgraph isomorphism algorithms. Uncertain subgraph mining [41] extends the classical subgraph isomorphism to uncertain graph. All these algorithms can efficiently solve the SPM with ontology consistency condition only.

11 Conclusion and Future Work

Due to the poor performace of both classical BPMs, workflow patterns in analyzing the data and resource exchange in Business Process, which is widely approved by the managers in practice, we introduce a new pattern-centered business process modeling language, named SPDL. Taking advantages of the SPDL formalization, we study the formalized definition of service pattern, in the form of business rules. To support applying service pattern in enterprise business process analysing, the service pattern matching problem (SPM) is studied. And we propose a service pattern matching approach in the Constructing-As-Identifying style. Besides, we discuss the effectiveness of our approach and propose two ways to improve the efficiency. Furthermore, a service pattern modeling and matching tool, called SPDL-Editor, is developed. As future work, it is interesting to consider the combination of our approach and subgraph isomorphism. It may take the advantage of the efficiency of subgraph isomorphism and cover the sematic consistency condition at the same time.

Acknowledgments

This work was supported by Zhejiang Provincial Natural Science Foundation of China under grant No.LY15F02007.

References

[1] J. Jeston and J. Nelis, *Business process management*. Routledge, 2014.

[2] Y. Li, B. Cao, L. Xu, J. Yin, S. Deng, Y. Yin, and Z. Wu, "An efficient recommendation method for improving business process modeling," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 1, pp. 502–513, 2014.

[3] M. Le, B. Gabrys, and D. Nauck, "A hybrid model for business process event and outcome prediction," *Expert Systems*, 2014.

[4] W. M. Van Der Aalst, A. H. Ter Hofstede, and M. Weske, "Business process management: A survey," in *Business process management*, ser. Business process management. Springer, 2003, pp. 1–12.

[5] C. Barreto, V. Bullard, T. Erl, J. Evdemon, D. Jordan, K. Kand, D. Konig, S. Moser, R. Stout, and R. Ten-Hove, "Web services business process execution language version 2.0 primer," *OASIS Web Services Business Process Execution Language (WSBPEL) TC, OASIS Open*, 2007, bPEL.

[6] A.-W. Scheer, O. Thomas, and O. Adam, "Process modeling using event-driven process chains," *Process-Aware Information Systems*, pp. 119–146, 2005.

[7] S. A. White, "Introduction to bpmn," *IBM Cooperation*, vol. 2, no. 0, p. 0, 2004.

[8] W. Van Der Aalst and K. M. Van Hee, *Workflow management: models, methods, and systems*. MIT press, 2004.

[9] Z. H. Wu, X. B. Wu, and M. M. Yao, *Business model innovation of modern service company a value network perspective*, 1st ed. Beijing: Science Press, 2013.

[10] D. Zelin, H. Shuhua, and Z. Wenjing, "Evaluation index system of modern service industry and its empirical analysis [j]," *Technology Economics*, vol. 10, p. 009, 2012.

[11] Z. Wu, "Modern service industry in china: Crossover, convergence, and complex services," Zhejiang University, Tech. Rep., 2014.

[12] W. M. Van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and parallel databases*, vol. 14, no. 1, pp. 5–51, 2003.

[13] S. A. White, "Process modeling notations and workflow patterns," *Workflow Handbook*, vol. 2004, pp. 265–294, 2004, bPMN.

[14] R. Hull, "Artifact-centric business process models: Brief survey of research results and challenges," in *On the Move to Meaningful Internet Systems: OTM 2008*, ser. On the Move to Meaningful Internet Systems: OTM 2008. Springer, 2008, pp. 1152–1163.

[15] N. Russell, W. M. van der Aalst, A. H. Ter Hofstede, and D. Edmond, "Workflow resource patterns: Identification, representation and tool support," in *Advanced Information Systems Engineering*, ser. Advanced Information Systems Engineering. Springer, 2005, pp. 216–232.

[16] N. Russell and W. M. van der Aalst, "Evaluation of the bpel4people and ws-humantask extensions to ws-bpel 2.0 using the workflow resource patterns," *Bpm center report, Department of Technology Management, Eindhoven University of Technology GPO Box*, vol. 513, 2007.

[17] P. Martin-Lof, "An intuitionistic theory of types," *Twenty-five years of constructive type theory*, vol. 36, pp. 127–172, 1998.

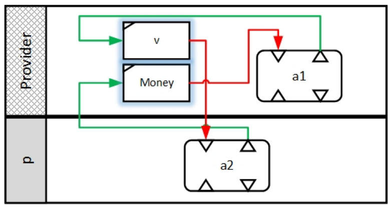
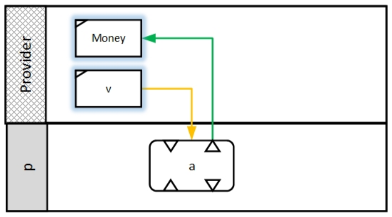
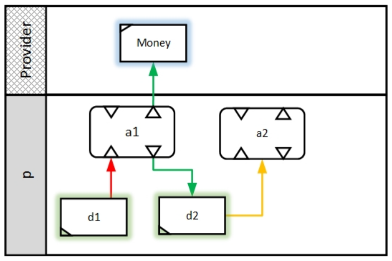
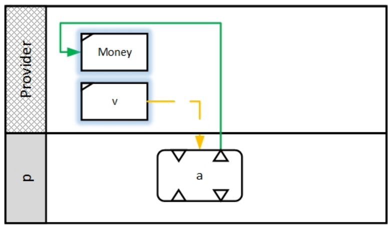
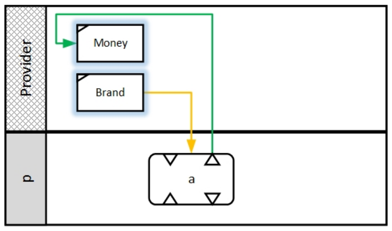
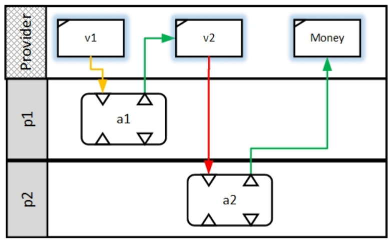
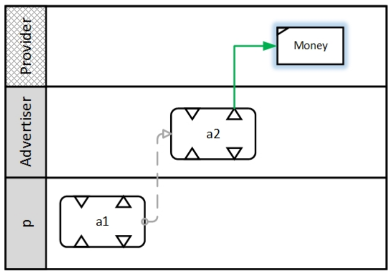
Name	SPDL diagram	Formalization
Asset Sale		$\Psi^r_{AS}[\![AssetSale]\!]$: $\sum(a_1, a_2 : Activity) \sum(v : Resource)$ $\sum(p : Participant)$ $\psi_A(a_1\,Provider) \times \psi_A(a_2\,p)$ $\times\psi_{AR}(a_2\,\nu_{or}(v)) \times \psi_{AR}(a_1\,\nu_{cr}(Money))$ $\times\psi_{AR}(a_1\,\nu_{or}(Money)) \times \psi_{AR}(a_1\,\nu_{cr}(v))$
Usage Fee		$\Psi^r_{UF}[\![UsageFee]\!]$: $\sum(a : Activity) \sum(v : Resource)$ $\sum(p : Participant)\psi_A(a\,p)$ $\times\psi_{AR}(a\,\nu_{ur}(v)) \times \psi_{AR}(a\,\nu_{cr}(Money))$
Subscription Fee		$\Psi^r_{SF}[\![SubscriptionFee]\!]$: $\sum(a_1, a_2 : Activity) \sum(d_1, d_2 : Data)$ $\sum(p : Participant)$ $\psi_A(a_1\,Provider) \times \psi_A(a_2\,Provider)$ $\times\psi_{AD}(a_1\,\nu_{od}(d_1)) \times \psi_{AD}(a_1\,\nu_{cd}(d_2))$ $\times\psi_{AR}(a_1\,\nu_{cr}(Money)) \times \psi_{AD}(a_2\,\nu_{ud}(d_2))$
Leasing		$\Psi^r_{L}[\![Leasing]\!]$: $\sum(a : Activity) \sum(v : Resource)$ $\sum(p : Participant)$ $\psi_A(a\,p) \times \psi_{AR}(a\,\nu_{er}(v))$ $\times\psi_{AR}(a\,\nu_{cr}(Money))$
Licensing Fee		$\Psi^r_{LF}[\![LicensingFee]\!]$: $\sum(a : Activity) \sum(p : Participant)$ $\psi_A(a\,p) \times \psi_{AR}(a\,\nu_{ur}(Brand))$ $\times\psi_{AR}(a\,\nu_{cr}(Money))$
Brokerage Sale		$\Psi^r_{BF}[\![BrokerageFee]\!]$: $\sum(a_1, a_2 : Activity) \sum(v_1, v_2 : Resource)$ $\sum(p_1, p_2 : Participant)$ $\psi_A(a_1\,p_1) \times \psi_A(a_2\,p_2)$ $\times\psi_{AR}(a_2\,\nu_{or}(v_2)) \times \psi_{AR}(a_2\,\nu cr(Money))$ $\times\psi_{AR}(a_1\,\nu ur(v_1)) \times \psi_{AR}(a_1\,\nu cr(v_2))$
Advertising		$\Psi^r_{UF}[\![Advertising]\!]$: $\sum(a_1, a_2 : Activity) \sum(p : Participant)$ $\sum(i : Line)$ $\psi_A(a_1\,p) \times \psi_A(a_2, Advertiser)$ $\times\psi_{CAA}(l\,a_1\,a_2) \times \psi_{AR}(a_2\,\nu_{cr}(Money))$

TABLE 4: The SPDL diagrams and formalizations of seven service patterns

[18] ——, "Constructive mathematics and computer programming," in *Proc. of a discussion meeting of the Royal Society of London on Mathematical logic and programming languages*, ser. Proc. of a discussion meeting of the Royal Society of London on Mathematical logic and programming languages. Prentice-Hall, Inc., 1985, pp. 167–184.

[19] B. Barras, S. Boutin, C. Cornes, J. Courant, J.-C. Filliatre, E. Gimenez, H. Herbelin, G. Huet, C. Munoz, C. Murthy *et al.*, "The coq proof assistant reference manual: Version 6.1," 1997.

[20] D. J. Howe, "Constructive type theory," *Logic, Algebra, and Computation: International Summer School*, vol. 79, p. 265, 2012.

[21] A. Osterwalder, Y. Pigneur, M. A.-Y. Oliveira, and J. J. P. Ferreira, "Business model generation: A handbook for visionaries, game changers and challengers," *African Journal of Business Management*, vol. 5, no. 7, 2011.

[22] Y. Huang and X. Wu, *Cases of Business Model Innovation in Service Sector*. Zhejiang University Press, 2010.

[23] L. Zhiling, "Spdl technological manual," 2013.

[24] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[25] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su, "Towards formal analysis of artifact-centric business process models," pp. 288 – 304, 2007 2007.

[26] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu, "Automatic verification of data-centric business processes," in *Proceedings of the 12th International Conference on Database Theory*, ser. Proceedings of the 12th International Conference on Database Theory. ACM, 2009, pp. 252–267.

[27] K. Bhattacharya, R. Hull, and J. Su, "A data-centric design methodology for business processes," *Handbook of Research on Business Process Modeling*, pp. 503–531, 2009.

[28] D. Calvanese, M. Montali, M. Estanol, and E. Teniente, "Verifiable uml artifact-centric business process models (extended version)," pp. 292–303, 2014.

[29] H. A. Reijers, T. Slaats, and C. Stahl, "Declarative modeling–an academic dream or the future for bpm?" in *Business Process Management*, ser. Business Process Management. Springer, 2013, pp. 307–322.

[30] P. Y. Wong and J. Gibbons, "A process semantics for bpmn." Springer, 2008, pp. 355–374.

[31] A. Rodríguez, A. Caro, C. Cappiello, and I. Caballero, "A bpmn extension for including data quality requirements in business process modeling," in *Business Process Model and Notation*, ser. Business Process Model and Notation. Springer, 2012, pp. 116–125.

[32] Z. Maamar, N. Faci, S. K. Mostefaoui, and E. Kajan, "Network-based conflict resolution in business processes," in *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*, ser. e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on. IEEE, 2013, pp. 132–137.

[33] C. Tziviskou, M. Palmonari, M. Comerio, and F. De Paoli, "Sedl-c: A language for modeling business terms in service descriptions," pp. 547–554, 2013 June 28 2013-July 3 2013 2013.

[34] B. Fan, Y. Li, S. Liu, and Y. Zhang, "Run jta in jtang: Modeling in artifact-centric model and running in activity-centric environment," in *Asia Pacific Business Process Management*, ser. Asia Pacific Business Process Management. Springer, 2015, pp. 83–97.

[35] Y. Li, Z. Luo, J. Yin, L. Xu, Y. Yin, and Z. Wu, "Enterprise pattern: integrating the business process into a unified enterprise model of modern service company," *Enterprise Information Systems*, no. ahead-of-print, pp. 1–21, 2015.

[36] W. M. P. van der Aalst, "Yawl: yet another workflow language," *Information Systems*, vol. 30, no. 4, pp. 245 – 275, 2005.

[37] R. Dijkman, M. Dumas, and L. García-Bañuelos, "Graph matching algorithms for business process model similarity search," in *Business Process Management*, ser. Business Process Management. Springer, 2009, pp. 48–63.

[38] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," pp. 313–320, 2001 2001.

[39] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data," in *Principles of Data Mining and Knowledge Discovery*, ser. Principles of Data Mining and Knowledge Discovery. Springer, 2000, pp. 13–23.

[40] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, ser. Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on. IEEE, 2002, pp. 721–724.

[41] A. Moustafa and M. Zhang, "Learning efficient compositions for qos-aware service provisioning," in *ICWS*, ser. ICWS. IEEE, 2014, pp. 185–192.

Jianwei Yin is currently a professor in the College of Computer Science, Zhejiang University, China. He received his Ph.D. in Computer Science from Zhejiang University in 2001. He is the visiting scholar of Georgia Institute of Technology, US, in 2008. His research interests include distributed network middleware, software architecture and information integration.

Zhiling Luo is the Ph.D. candidate in College of Computer Science, Zhejiang University, China. He received his B.S. in Computer Science from Zhejiang University in 2012. His research interests include service computing and data mining.

Ying Li received the B.S., M.S. and Ph.D. degrees in computer science from Zhejiang University, China,in 1994, 1997 and 2000, respectively. He is currently an associate professor with the College of Computer Science, Zhejiang University. He is currently leading some research projects supported by National Natural Science Foundation of China and National High-tech RD Program of China (863 Program). His research interests include service computing, business process management and compiler.

Zhaohui Wu received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 1988 and 1993, respectively. He is currently a Professor with the College of Computer Science, Zhejiang University. His research interests include distributed artificial intelligence, semantic grid, and pervasive computing. Dr. Wu is a Standing Council Member of the China Computer Federation.