A Two-tier Data-centric Framework for Flexible Business Process Management

Emily Liu
Research, Hawthorne, United States., rliu@us.ibm.com

Frederick Wu
Research, Hawthorne, United States., fywu@us.ibm.com

Florian Pinel
Research, Hawthorne, United States., pinel@us.ibm.com

Zhe Shan
Accounting, Law, and Computer Information Systems, Manhattan College, Riverdale, NY, United States., zhe.shan@manhattan.edu

A Two-tier Data-centric Framework for Flexible Business Process Management

Rong Liu
IBM T. J. Watson Research Center
rliu@us.ibm.com

Frederick Y. Wu
IBM T. J. Watson Research Center
fywu@us.ibm.com

Florian Pinel
IBM T. J. Watson Research Center
pinel@us.ibm.com

Zhe Shan
School of Business, Manhattan College
zhe.shan@manhattan.edu

ABSTRACT

Business process management provides a means of coordinating interactions between workers and organizations in a structured way. However, the dynamic nature of the modern business environment requires these processes are subject to an increasingly wide range of variations. Therefore, flexible approaches are needed to deal with these variations in order to maintain viable business. In this paper, we propose a two-tier data-centric framework to achieve process flexibility. Our approach is based on Business Entity, a new process modeling paradigm widely recognized in recent years. We design a process design business entity (PD entity) to include business process definitions as a part of its information, and process execution business entities (PE entities) provide the context for defining the behavior of activities in the processes. The business processes, as the PD entity data, can be modified on-the-fly and evolve naturally as the PD entity progresses through its lifecycle. We illustrate this framework with an example from the travel service industry. It shows that this framework is able to improve process flexibility, empower business users with capability of making timely process changes, and reduces the burden of managing process evolution.

Keywords

Business process management, case management, process flexibility, business entity, data-centric, two-tier framework.

INTRODUCTION

Business Process Management (BPM) is a management practice which encompasses all activities of identification, definition, analysis, design, execution, monitoring and measurement, and continuous improvement of business process. In order to retain their competitive advantage in today's dynamic marketplace, it is increasingly necessary for enterprise to alter processes rapidly in response to the volatility of global business environment (Cantara 2008). Therefore, companies need to manage processes that change frequently – often weekly or even daily. Moreover, in some situations they need to deviate from processes and even violate predefined constraints on a case-by-case basis (de Man 2009b). Unfortunately, most BPM technologies are not compatible with frequent ad-hoc changes. In those existing BPM systems (BPMS), processes are modeled at design time by system analysts with process modeling skills. At run time, business users execute these activities rigidly following the pre-designed models (Basu and Kumar 2002; van der Aalst and Jablonski 2000). When business users encounter unanticipated situations they must work around the system and communicate necessary changes to the analysts. As a result, business users perceive traditional BPMS as inflexible and unresponsive to business requirements.

In recent years, process flexibility research has attracted more and more attention in BPM area (Schonenberg Mans Russell Mulyar and van der Aalst 2008). The existing approaches (Pesic Schonenberg Sidorova and P. 2007; Pesic and van der Aalst 2006; Reichert and Dadam 1998; Schonenberg et al. 2008; van der Aalst et al. 2000; van der Aalst and Pesic 2006) accommodate process variation by introducing declarative constraints in process schemas or allowing controlled dynamic alteration of schemas. Complex issues arise from these changes, such as verification of correctness and workflow instance migration (Casati Ceri Pernici and Pozzi 1998). In general, these techniques handle only a limited range of process changes through careful a priori design, and do not satisfy the business need for ad hoc change. Case management (de Man 2009b; Swenson 2010) has been proposed as a way to manage business processes that require constant changes. These processes are

performed by "knowledge workers", or domain experts, who are entrusted with the authorization to plan and carry out the prosecution of the case. At any time during the case prosecution, the worker is enabled to change the plan in any way, including adding activities, deleting activities, or changing their sequence. However, So far, a standardized control language for case management does not yet exist.

To address the challenges of managing highly volatile business processes, in this paper we propose a two-tier data-centric approach based on business entities (Bhattacharya Caswell Kumaran Nigam and Wu 2007; Kumaran Bishop Chao Dhoolia Jain Jaluka Ludwig Moyer and Nigam 2007; Kumaran Liu and Wu 2008; Liu Bhattacharya and Wu 2007; Liu Wu Patnaik and Kumaran 2009; Nigam and Caswell 2003). Business entity is a recently introduced data-centric process modeling paradigm, which has been successfully employed in a variety of customer engagements (Bhattacharya et al. 2007). Each business entity has an information model and a lifecycle model. The information context is updated as the business entity progresses along its lifecycle. All these characteristics make it an appropriate paradigm for flexible process management (de Man 2009a). In this work, we extend the model of business entity by incorporating process schemas into its information model. Thus, the process model of a running instance can be dynamically adjusted as the information content of a business entity evolves throughout its lifecycle.

Our approach can be regarded as a hybrid paradigm that blends the models of both traditional BPM and modern case management. First, processes can be pre-designed at the template level, and then be executed without deviation. But, flexibility can be granted at process instance level. Hence authorized users can change the process schema for specific instances at runtime. Compared to ad-hoc case management, business entities in our approach are well defined, and provide a unified information model and lifecycle structure as a stable context for dynamic process execution. This makes it possible to perform performance monitoring and knowledge consolidation on heterogeneous instances. Therefore, our approach can not only handle runtime changes in case management scenarios, but also stratify flexibility to different levels towards efficient management practice.

The rest of the paper is organized as follows. Section 2 describes several sample scenarios from different industries to reveal a common process pattern that fits with our approach. In Section 3, we review the business entity-centric process paradigm, and present its formal model. Section 4 introduces our two-tier framework, and discusses how to achieve process flexibility in terms of change management. The implementation of our approach is discussed in Section 5. Section 6 compares our approach with related work. Section 7 concludes this paper with a plan of our future work.

MOTIVATING SCENARIOS

Our research is motivated by several interesting examples from different industries. We identify a common pattern and analyze its modeling requirements.

- **Travel Agency**. An agency specializing in international travel offers a range of custom services to clients, including visa services, flight reservation, hotel reservation, tours, etc. The agency offers very flexible end-to-end trip arrangements. A typical scenario starts when a customer signs an agreement with the agency. For an international trip, the first priority is to obtain a valid visa, for which the agency can provide levels of services ranging from light consultation to full service. Once a visa is granted, the trip arrangement continues with flight reservation, hotel booking, etc. as requested by the customer. The customer may change or cancel these reservations at any time. After the trip begins, the customer may ask for more services, such as a local tour, tour guide, ticket booking etc. Trip arrangement is not a predictable process. It is defined during execution. However, process fragments for handling specific requests, for instance, flight reservation and visa applications etc, can be predefined and modeled. In addition, the process flow is driven by the data in the trip plan, and in some cases, there is no explicit sequence of activities. For instance, when a flight is reserved, several activities, e.g. hotel reservation and booking local tours, can be executed in any order. Similarly, changing flight reservations may lead to a series of changes to other reservations. While each trip process instance represents a customer's unique travel experience, specific process fragments (e.g. flight reservation, flight cancellation) are repeated in many process instances. Also a process instance may serve as a template for starting new instances of trips to the same destination. Therefore, it would be useful if process instances could be searched so that agents can learn from customer experiences and make good recommendations to customers.
- **IT Service Delivery.** Service delivery centers (SDCs) provide a range of IT services to outsourcing customers, including server provisioning, server decommissioning, disaster recovery etc. Server provisioning includes installing operating system, installing security patches, and other IT service tasks. To improve service quality and efficiency, IT service tasks have standard information requirements and procedures. When a customer requests IT services, an SDC selects appropriate service process templates and tailors them to the customer's needs. During execution, the customer may request various changes, for example, adding new services or dropping services that are already in progress.

- **Clinical Processes**. Similarly, in medicine, a physician sets up a treatment plan for a patient based on an initial diagnosis following clinical guidelines. The treatment plan defines a process for caregivers to follow. However, as the patient's condition changes, the physician needs to alter the plan on-the-fly.

These examples reveal a common pattern in process design and execution. First, for the purpose of economic efficiency, the work is divided into granular units (e.g. various travel services, IT service tasks, or lab tests), which are standardized and relatively stable. Second, process templates (i.e. IT service process templates or clinical guidelines) are used to accelerate process design, but tailored to customer needs. Third, the actual process is defined while it is executed. Hence, flexibility is the most notable feature. These processes require almost all types of flexibility classified under the taxonomy described in (Schonenberg et al. 2008). They may be underspecified due to the lack of clarity of customer needs -- *flexibility by underspecification*, and some predictable changes can be considered during process design (*flexibility by design*), but most of them are unpredictable, occur during run-time, and mandate process modification on-the-fly (*flexibility by change* or *flexibility by deviation*). Finally, though each process instance represents a unique customer experience, process instances are useful resources for designing new processes and templates, as well as auditing. Thus, process instances need to be properly managed for search and analysis.

To support this pattern, we propose a two-tier, data-centric approach to process modeling and execution. We treat a business process as context data, which can naturally change as the process executes. There is no boundary between design time and run time. Also, process instances can managed in the same way as data in a database. We begin with an introduction to the concept of business entities.

BUSINESS ENTITY – A DATA-CENTRIC PROCESS MODELING PARADIGM

The concept of *business entities* (a.k.a. business artifacts in (Nigam et al. 2003)) has been proposed to model business processes and information in a unified way. Each *business entity type* is characterized by an information model and a lifecycle model. A business process can be captured as a collection of interacting business entity types. During execution, *business entity instances* (for short, *business entities* or *BEs*) are created and progress through their lifecycles. A BE is self-contained in the sense that (1) each state along its journey is defined by its current information content, and (2) progress from one state to the next requires the specification of the activities that need to be performed to accomplish the transition. The advantages of business entity-centric modeling have been described (Bhattacharya et al. 2007; Nigam et al. 2003).

Definition 1 (Business Entity Type): A business entity type e is a tuple $e = <name_e, ds_e, id_e, l_e>$, where $name_e$ is the name of e, ds_e is the data schema of e, id_e is an attribute serving as an identifier of instances of e, and l_e is the lifecycle of e. l_e is typically represented as a finite state machine, $l_e = <S, TR>$, where S is a set of states, TR is a set of state transitions. For any $t \in TR$, $t = <s_{source}, s_{target}, act>$, where $s_{source}, s_{target} \in S$ and act is the activity that causes the state transition.

We illustrate the business entity concept with the travel agency example. The agency's business consists of a collection of travel-related services, each of which can be modeled as a BE. For example, Flight Reservation can be modeled as the BE shown in Figure 1. When a reservation request is submitted, a Flight Reservation BE is created. The agency executes activities to change the entity state and record relevant information. For instance, after an agent reserves a flight, the ticket information is recorded and the BE instance is "Reserved", as shown in Figure 1 (c). The agency may reschedule or cancel the flight reservation. Each state change or data change can be viewed as an event. The business entity can publish these events for processes or other BEs to use. For example, when a flight is rescheduled to another day, this event can trigger the change of a hotel reservation.

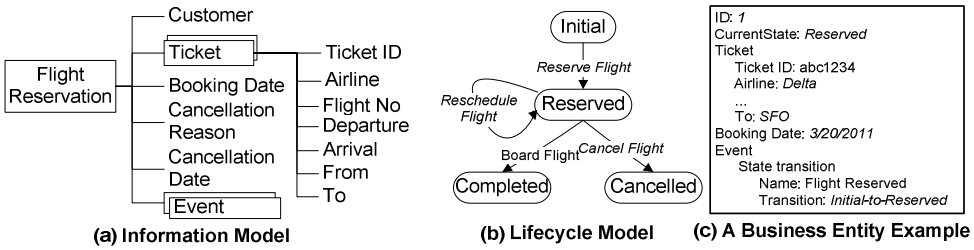

Figure 1: Flight Reservation Business Entity

As shown in Figure 1 (b), activities move a BE along its lifecycle. To provide travel services to customers, the agency executes a number of activities that are standardized and form an activity catalog. Each input to (or an output from) an activity is either a BE in a specific state or a static data source. For example "Customer", a static data source, is an input to

the activity in Figure 2(a). In Figure 2(b), a Visa Application business entity in "Granted" state is required before an international flight can be made. The output of this activity is a Flight Reservation in "Reserved" state. Activities can be initiated by human role players or triggered by events. In Figure 2(c), "Reserve Hotel" is triggered by a state transition event "Flight Reserved" or "Flight Rescheduled" from a Flight Reservation BE. After this activity is executed, although the inputs are still available, it cannot be enabled again unless another "Flight Rescheduled" event arrives.

Definition 2 (Activity): An activity a is a tuple $a = <name_a, I_a, O_a, V_a>$, where $name_a$ is the name of a, I_a is the set of information inputs that are needed in order to perform a, O_a is the set of information outputs produced by completing a, and V_a is the set of triggering events to enable a. $I_a, O_a = \{e(s)\} \cup D$, where $e(s)$ is a business entity e in state s, and D is a set of static data elements. Often a triggering event is a state transition event $\{v_{st}(e, tr)\}$, where e is a business entity and tr is a state transition.

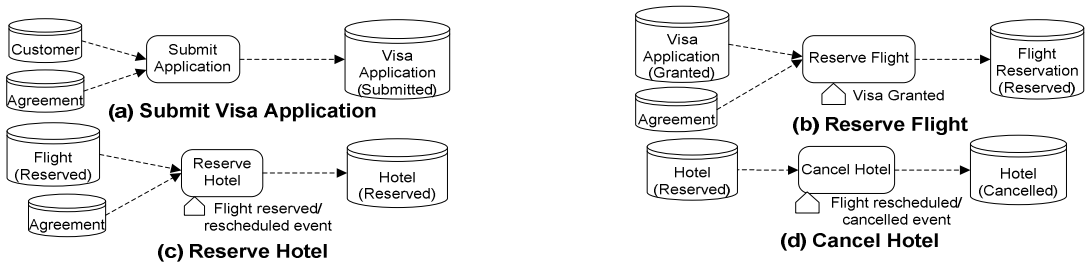

Figure 2: Activity Examples

These BEs capture established practices of providing specific services in the industry. Therefore, they are quite stable, particularly with respect to their information and lifecycle models. Rarely, the activities that lead to state transitions may vary. However, as described before, most changes come from trip processes which are tailored for specific customer needs. Next we describe our framework for handling this type of change.

A TWO-TIER PROCESS MANAGEMENT FRAMEWORK

Extended Entity Model

To enable run time changes at either the process schema or instance level, we extend business entity modeling to incorporate process models as a part of business entity information. As the business entity progresses through its lifecycle, the process model can evolve naturally as the information model is updated. The architecture of our solution framework is shown in Figure 3. This framework contains two tiers of business entities. The first tier is an overarching BE called *process design business entity* (or PD entity). We design the information model of the PD entity to store processes, which can be defined during execution or frequently changed. As the PD entity progresses through its lifecycle, the business processes are launched and executed.

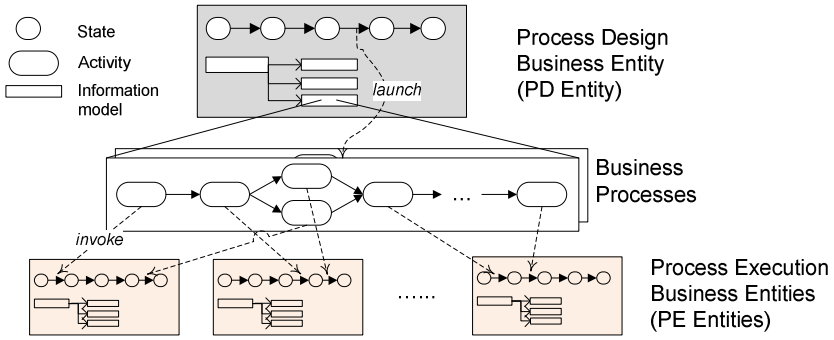

Figure 3: Two-Tier Framework

Definition 3 (Business Process): A business process p is a tuple $p = <name_p, A_p>$, where $name_p$ is the name of p, A_p is the set of activities in p.

Definition 4 (Process Design Business Entity or PD entity): A process design business entity has one or more business processes defined in its information model and the processes are executed in a state of its lifecycle.

A process is composed of activities with inputs and outputs specified in terms of a collection of BEs in the second tier. These BEs, representing the real work of a business process, e.g, Visa Application or Flight Reservation in the travel agency example, are referred to as *process execution business entities* (or *PE entities*). The process creates new PE entities, changes their states, and orchestrates their interactions. The execution order of the process can be inferred from the inputs and outputs of activities. Hence, when defining a process in a PD entity, domain experts or business users focus on definition of activities instead of the execution order. Graphical representations of the process can be automatically generated to help validate the execution order.

To illustrate this framework, we model Trip Arrangement as a PD entity shown in Figure 4. Unlike a regular BE, this PD entity contains a Trip Process definition and references to PE entities. A Trip Arrangement instance is created when a customer signs an Agreement with the agency. Then based on the customer's needs, a Trip Process is designed and added to the business entity. This process contains catalog activities that the agency will execute to prepare the customer's trip, such as visa application and flight reservation. The references to the PE Entities that define activity inputs and outputs are stored in the PD entity. For example, Figure 5(a) shows a Trip Process created at design time. It contains activities for reserving flights, hotels, and two local tours (see Figure 2). The table on the right lists PE entities that are the inputs or outputs of activities in this process. Since this process has not been launched, no Instance ID has been recorded yet.

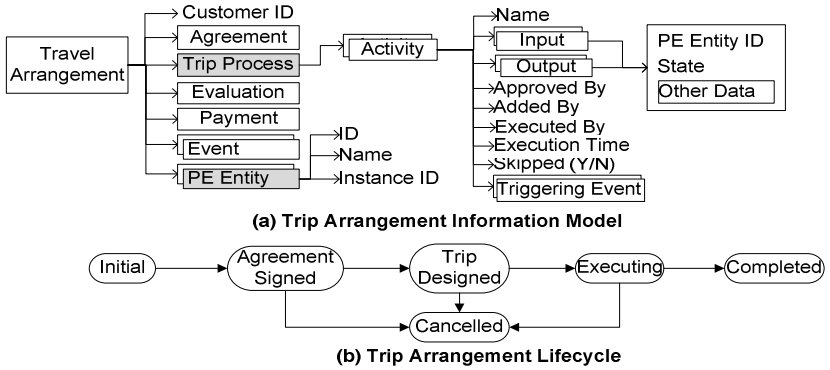

Figure 4: Trip Arrangement Business Entity (Process Design Business Entity)

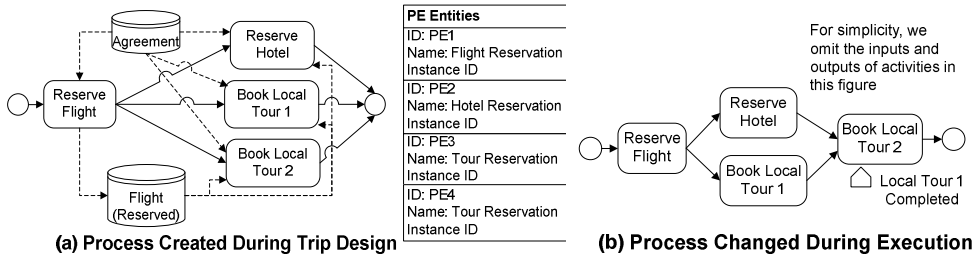

Figure 5: A Trip Process

After a trip process has been designed, process execution can begin. The instance ID of input and output BEs are recorded in the Travel Arrangement PD Entity. During execution, new activities can be added, and existing activities can be skipped or removed. For example, during execution, the agent realizes that many customers cancel Tour 2 after taking Tour 1. She decides to delay booking this tour till after the customer has completed Tour 1, and still shows interest in Tour 2. The changed Trip Process is shown in Figure 5(b). Activity "Book Local Tour 2" has a triggering event "Local Tour 1 Completed".

Figure 3 illustrates the relationship between our approach and traditional activity-centric process modeling and modern case management. As it shows, the business process model corresponds to the Business Process layer alone. The two main distinctions of our approach are: it is data-centric vs. activity-centric and instance-based vs. schema-based. Based on BE-centric modeling, in our approach data provides context for defining activities and processes. In activity-centric process modeling, activities are the focal point around which information is organized. Detailed comparison can be found in (Kumaran et al. 2008; Liu et al. 2007; Liu et al. 2009). Second, an activity-centric model mandates how work is done and

many process instances are spawned from it. In our framework, each process instance corresponds to a PD entity instance, which also contains the process schema. Thus, there is no distinction between a process schema and an instance.

Case management (de Man 2009b; Swenson 2010), in general, can be considered as a variant of Figure 3 with no PE entities in the second tier. A process is specially tailored for a case (i.e. a PD entity) and becomes a part of the case. Activities in the process are defined in the context of the case folder, rather than the more granular and structured context provided by PE entities. In our framework, PE entities prescribe the behavior of activities. This constraint is suitable for a process pattern where activities are commonly standardized.

Therefore, we view our approach as a fusion of traditional workflow technologies and data-centric modeling, and a specific case management technique. In the spectrum of BPM technologies with respect to flexibility and structuredness, our approach can be positioned in the middle between traditional workflow approaches and case management.

Change Management

Our framework achieves great flexibility at process instance level designed for business users or domain experts. A process can be modified during either design or execution time. Also, because each activity is data-driven with explicit inputs and outputs, users can focus on activities without concern for the sequence of activities. A graphical view can be generated or a verification tool can be used to ensure valid execution order. The user interface for defining and modifying a process can be relatively simple. Essentially, its basic function is to let users choose PE entities and design activities. Note that changes to PE entities are not covered in this paper. A reasonable assumption is that PE entities are abstractions of customary behavior of a business domain and are rarely changed. For example, the Flight Reservation in Figure 1 captures the common information and key states that concern a travel agent. Similarly, in the IT service industry, an IT Service Task (e.g. install operating system) has standard lifecycle (e.g. assigned, executed, reviewed etc.) as well as data at each state. Changes to these models are infrequent.

In our framework, since a process is stored as part of the information of a PD entity, we can modify it just as any other data. In general, we can modify, remove, or skip activities that have not been executed, or add new activities.

- *Modifying*: Changing the triggering events, static data inputs/outputs, or read-only BE inputs of an activity. The change causes no alternation to the state transition associated with this activity. If the change will be reused for future process instances, this modified activity should be added to the activity catalog.
- *Add*: Selecting an activity from the activity catalog and adding it to the process.
- *Remove*: Deleting an activity from the process.
- *Skip*: If an activity becomes unnecessary in a process, it is skipped during execution. The attribute value "Skipped (Y/N)" (see Figure 4 (a)) is set to "True". However, this activity is still kept in the process in case this process will be used as a template and other instances originating from this template need this activity.

Modifying a process in our framework is rather simple. Typically, two data items in the PD entity are involved in a change: *PE Entity*, a list of references to PE entities, and *Activity*, which stores activity definition and execution information (see Figure 4(a)).

Next, we use an example to illustrate how to change a Trip Process. A customer plans to travel to New York and is interested in three local tours. An agent helps the customer get a valid visa and makes reservations for flights, hotels, and the local tours. Each of these services is captured as a PE entity. The PD entity, Trip Arrangement (see Figure 4(a)), has references to these PE entities. The agency designs a trip process which involves six PE entities (PE1-6) as shown in the white part of Table 1. At design time, these PE entities are not initialized and therefore the "Instance ID" column has no value. The activities of the trip process (A1-6) are shown in the white portion of Table 2. Each activity is defined with PE entities as inputs and outputs and triggering events (we omit static data inputs and outputs for simplicity). For example, after the visa application (PE1) is granted, flights (PE2) are reserved. It is very straightforward for business users to define activities in a table. These activities are saved in the PD entity information and can be translated into a graphical representation, for example, shown in Figure 6(a). Note that "Reserve Hotel" (A3) and "Book Local Tour" (A4-6) can be executed in any order because these activities require the same PD entity (Flight Reservation) as a read-only input. Next, we describe two changes.

Change #1: During the execution of the trip process, the agent finds that no reservation is needed for Local Tour 3. The tour can be given whenever requested. Hence, this activity is skipped, i.e. "Skipped (Y/N)" is set to "Y" in Table 2. Later, due to a change in the customer's itinerary (an event), the reserved flights must be rescheduled to other dates. Consequently, the reserved hotel has to be cancelled, and a new hotel reservation is needed. Also, the customer cannot make the first local tour

and hence the tour reservation needs to be cancelled. To make this change, a series of activities (A7-10) are added to the trip process (see Table 2) and will be executed. Similarly, these added activities can be converted into a graphical representation as shown in Figure 6(b). Since a new hotel reservation is needed, a new PE entity (PE7) will be created by Activity A9. As shown in Table 1, reference to this new PE entity is added to the PD entity. At this point, activities A1-5 have been executed and PE entities (PE1-5) have been created. The Instance IDs of these entities are recorded in the PD entity. The grey portion of Table 1 shows the data has been added between design time and Change #1.

Change #2: After completing the first local tour, the customer is not happy with the tour and decides to take a self-guided tour with a rental car. To accommodate this change, the agent reserves a rental car (A11) (see black area of Table 2). The graphical view of this portion is shown in Figure 6(c). A new PE entity, Car Rental, is added to Table 1.

ID	Name	Instance ID
PE1	Visa Application	V1
PE2	Flight Reservation	F1
PE3	Hotel Reservation	H1
PE4	Tour Reservation	T1
PE5	Tour Reservation	T2
PE6	Tour Reservation	
PE7	Hotel Reservation	H2
PE8	Car Rental	R1

Table 1: PE Entity information added in different phases: (a) white area – design time, (b) grey area – immediately after Change #1, and (c) black area – immediately after Change #2

Activity ID	Activity	Inputs		Triggering Events	Outputs		Skipped (Y/N)
		PE ID	State		PE ID	State	
A1	Submit Application	PE1	Initial	v_1	PE1	Submitted	
A2	Reserve Flight	PE1	Granted	v_2	PE2	Reserved	
A3	Reserve Hotel	PE2	Reserved	v_3	PE3	Reserved	
A4	Book Local Tour 1	PE2	Reserved	v_3	PE4	Reserved	
A5	Book Local Tour 2	PE2	Reserved	v_3	PE5	Reserved	
A6	Book Local Tour 3	PE2	Reserved	v_3	PE6	Reserved	Y
A7	Reschedule Flight	PE2	Reserved	v_4	PE2	Reserved	
A8	Cancel Hotel	PE3	Reserved	v_5	PE3	Cancelled	
A9	Reserve Hotel	PE2	Reserved	v_5	PE7	Reserved	
A10	Cancel Tour 1	PE4	Reserved	v_5	PE4	Cancelled	
A11	Reserve Rental Car	PE7	Reserved	v_6	PE8	Reserved	

Note:
v_1: Customer Request 1
v_2: PE1 Granted, i.e. v_{st}(PE1, submitted-to-granted),
v_3: PE2 Reserved, i.e. v_{st}(PE2, Initial-to-Reserved),
v_4: Customer Request 2
v_5: PE2 Rescheduled, i.e. v_{st}(PE2, Reserved-to-Reserved),
v_6: Customer Request 3

Table 2: Activities added in different phases: (a) white area – design time, (b) grey area – immediately after Change #1, and (c) black area – immediately after Change #2

Note that all of these changes are made to a process instance without affecting other instances. However, it is also possible to apply these changes to a process schema. First, this process instance can be saved as a template for creating new instances. Second, the changes can even be applied to other running instances. For example, if Local Tour 3 is no longer available,

technically, an SQL statement can remove the "Book Local Tour 3" activity that has not been executed from all running process instances. With traditional workflow technologies, modifying a process model at run time involves complications with existing instances. Some instances can be migrated, while others cannot. As a result, different versions of the process model are needed. This burden of managing process evolution is exacerbated if a process is full of ad hoc changes.

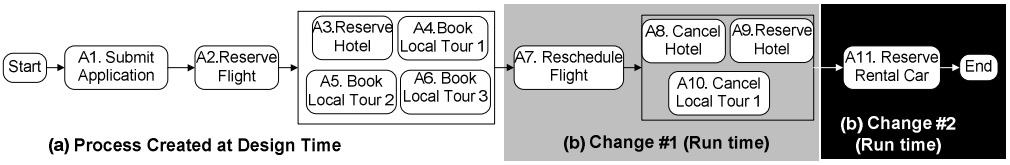

Figure 6: A Trip Process with Changes

Correctness Verification

In our framework, a process is said to be valid if each activity in this process can be enabled. For an activity to be enabled, its inputs must be available and its triggering events must occur. Since its inputs are specified in terms of PE entities, enabling it requires that each input PE entity has reached the specified state. The reachability of this state can be decided based on the entity lifecycle. For triggering events, we focus on state transition events because their availability can be forecast. However, the arrival of other types of events, for example, external events or data change events, is unforeseeable. The sketch of our verification algorithm is shown in Figure 7. This algorithm takes a process definition and the associated PE entity types as inputs, calculates the relationships between any pair of activities in the process (i.e., connected, exclusive, or parallel), tests the arrival of triggering events, and lists incorrectly defined activities and triggering events as outputs. Now we apply it to the process defined in Table 2.

Notations

$e(s)$: Business entity e in state s.

$v_{st}(e, tr)$: A state transition event associated with state transition tr of e

$r(x, y)$: Relationship between activities x and y. (i) if $r(x,y)=(x \rightarrow y)$, there is a path from x to y, (ii) if $r(x,y)=(x \backslash y)$, x and y are exclusive, and (iii) if $r(x,y)=\emptyset$, x and y can be executed in parallel

Algorithm 1: Process Verification (Sketch)

Inputs: a business process definition and associated PE entity models

Outputs: ACT_{err}: Incorrectly defined activities,

V_{err}: Incorrect triggering events of activities

R: Relationships of activities

(1). Determine the relationships of the activities in a process

(1.1). For any pair of activities x and y in the process, let $r(x, y) = \emptyset$.

(1.2). For any pair of activities act_1 and act_2, and for any entity e such that $e(s_1)$ is an output of act_1 and $e(s_2)$ is an input of act_2,

a) If s_2 can be reached from s_1, let $r'(act_1, act_2) = (act_1 \rightarrow act_2)$

b) If s_1 and s_2 are exclusive (i.e. there is no path containing both of them), let $r'(act_1, act_2) = (act_1 \backslash act_2)$. If act_1 and act_2 correspond to state transitions tr_1 and tr_2 respectively, and tr_1 and tr_2 have the same source state, let $Decision(act_1, act_2)$ = true, i.e. act_1 and act_2 are connected from a decision node.

If $r(act_1, act_2)$ and $r'(act_1, act_2)$ are in conflict, i.e. one is $(act_1 \backslash act_2)$ but the other is $(act_1 \rightarrow act_2)$, add act_1 and act_2 to ACT_{err}. Otherwise, let $r(act_1, act_2)=r'(act_1, act_2)$ if $r(act_1, act_2)=\emptyset$.

(2). Check the correctness of each triggering events. For each triggering event $v_{st}(e, tr)$ of any activity act, let x be the activity corresponds to tr.

(2.1). If x is in the process, x must be executed before act. Therefore,

a) If $r(x, act) = \emptyset$, set $r(x, act) = (x \rightarrow act)$. This event is valid.

b) If $r(x, act) = (x \rightarrow act)$ but $r(act, x) = \emptyset$, this event is valid.

c) If $r(x, act) = (x \rightarrow act)$, and $r(act, x) = (act \rightarrow x)$, then x and act form a loop. If x is closer to the entrance of the loop than act, this event is valid.

d) Otherwise, (i) act and x are either exclusive or (ii) act is supposed to be executed before x but this cannot happen (a deadlock situation). Add $v_{st}(e, tr)$ to V_{err}.

(2.2). If x is not in the process, check whether x is indeed executed outside of the process or x should be brought into the process

Figure 7: Process Verification Algorithm

Following Step (1), we get the following relationships: (A1→A2), (A2→A3), (A2→A4), (A2→A5), (A2→A6), (A2→A7), (A9→A11). In Step (2), since v_1, v_4 and v_6 (Customer Request) are external events, we cannot forecast their arrival and are excluded. Event v_{st}(PE1, "Submitted-to-Granted") is associated with activity "Grant Visa", which is not executed by the agent. Instead, when this activity is completed, a message is sent to the agent to start the state transition. This event can only be monitored. v_{st}(PE2, "Initial-to-Reserved") and v_{st}(PE2, "Reserved-to-Reserved") are caused by activities in the process. The first event does not bring any new relationship. The second event brings new relationships (A7→A8), (A7→A9), (A7→A10). The process is valid without any error.

With the relationships between activities are identified. It is easy to construct a graphical representation of the process. The algorithm is shown in Figure 8. Continue the example with this algorithm, we get a graphical representation which is very similar to Figure 6, except that A7 (Reschedule Flight) can be executed in parallel with A3-6, which have no dependency with A7. However, A7 is in fact executed after A3-6. This temporal constraint is taken into account in Figure 6, which shows A7 is connected from A3-6.

Algorithm 2: Generating Graphical Representation of a Process (Sketch)

Inputs: *R* - Relationships of activities, resulting from Algorithm 1.

Outputs: *G* - The process graph of this process

(1). (*Connection*) For any pair of activities *x* and *y* with $r(x, y) = (x \rightarrow y)$, connect *x* to *y*.
(2). (*Decision*) For any pair of activities *x* and *y* with *Decision*(*x*, *y*) = true, connect them from a decision node and reconnect all incoming connectors of *x* and *y* to this decision node. Collapse two decision nodes into one if they are directly connected.
(3). (*Start and End*) For any activity *act* corresponding to a state transition *tr*, (i) if the source of *tr* is an initial state, and *act* is not connected from any other activity, connect it from a "Start" node. (ii) If the target of *tr* is a final state, and *act* is not connected to any other activity, connect it to an "End" node.
(4). (*Remove redundant paths*). For a connector between two activities, if there is already path with length more than 1 between them, remove this connector.

Figure 8: Algorithm for Generating Graphical Representations of Processes

IMPLEMENTATION

Figure 9 describes the implementation of our approach and combines two solutions for the two kinds of business entities. PE entities are implemented using Data4BPM (Nandi Konig Moser Hull Klicnik Claussen Koppmann and Vergo 2010), a business entity-centric BPM runtime for Websphere Process Server (IBM 2008). Each activity in a process is converted into a WSBPEL (OASIS WSBPEL TC 2007) process snippet enhanced with data extensions (known as BPEL4Data) to specify data access operations on certain activities. In turn, the data access operations manipulate BE data using the BEDL (Business Entity Definition Language) runtime (Nandi et al. 2010).

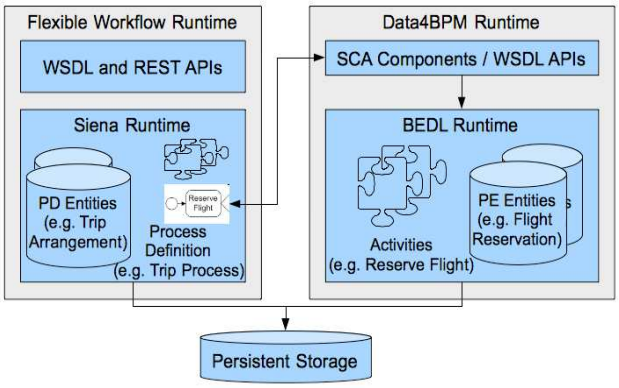

Figure 9: Implementation Architecture

PD Entities require a different approach to accommodate on-the-fly changes to their embedded processes. This feature is provided by an exploratory research project named Siena (Cohn Dhoolia Heath Pinel and Vergo 2008). The core of this implementation is a container that facilitates data access, access control, lifecycle management and microflow execution (a lightweight BPEL engine). This runtime can communicate with other components through protocols such as Web Services, and a user interface can be plugged into the framework. The processes are executed using several Siena microflows in charge

of creating process instances as well as creating, claiming and completing activity instances. For each activity instance creation, a microflow communicates with Data4BPM using a Web Service to initiate a BPEL snippet.

In Figure 9, we also illustrate the flight reservation activity in the case of a trip process. As an agent creates a trip, a new Trip Arrangement entity is created by the Siena Runtime. When this entity changes its state to "Executing", the Siena microflow engine reads the Trip Process stored in the entity and creates an instance of the first activity of this Process, "Reserve Flight", by making a Web Service call to Data4BPM. Data4BPM instantiates a "Reserve Flight" BPEL snippet, which consists of at least two tasks: a human task letting a user enter the flight information, and a BPEL4Data task to create "Flight Reservation" (a PE entity) and persist the data using the BEDL runtime. State transition events may be broadcast to the Siena Runtime by subscription. After the BPEL snippet is complete, the Siena microflow engine is notified and it records the task execution information in the PD entity. Then, the engine reads the Trip Process again to find the next activity, calls Data4BPEL to check the availability of input entities of this activity, and determines whether this activity can be enabled. If this activity is enabled, the engine moves on to it.

RELATED WORK

Process flexibility has been an area of great interest, particularly in the area of business process management (Baez Casati and Marchese 2009; Casati et al. 1998; Ceri Daniel Matera and Raffio 2009; Pesic et al. 2007; Pesic et al. 2006; Reichert et al. 1998; Schonenberg et al. 2008; van der Aalst Basten Verbeek Verkoulen and Voorhoeve 1999; van der Aalst et al. 2000; van der Aalst et al. 2006; Weber Reichert and Rinderle-Ma 2008). A comprehensive review of literature on process changes can be found in (van der Aalst et al. 1999; van der Aalst et al. 2000). A survey of the contemporary approaches to process flexibility is given in (Schonenberg et al. 2008). The approach to flexibility in (Ceri et al. 2009) is similar to ours in that each process instance has its own process definition. This work considers activities in a process to have a universal lifecycle, different from our approach where activities are defined with a collection of PE entities. A universal lifecycle management is given in (Baez et al. 2009). In this approach, the implementation of a state transition is separated from its definition to support light coupling. This idea coincides with the concept of PE entities where state transitions are implemented by dynamic processes. Declarative languages, for example, ConDec (Pesic et al. 2007; Pesic et al. 2006) and DecSerFlow (van der Aalst et al. 2006), are proposed to avoid over-specification in process schemas and thus provide process flexibility. An application of declarative languages in clinical practices to address process flexibility issues is given in (Mulyar Pesic van der Aalst and Peleg 2008). However, it may be challenging to build a declarative model for a complicated business scenario. ADEPT system (Reichert et al. 1998) provides a set of change operations to support dynamic deviation from process schemas with correctness properties considered. However, managing process schema evolution is a complicated issue, as discussed in (Casati et al. 1998).

Our work adds a new approach to Case Management (de Man 2009b; Swenson 2010). Case management has been proposed as a way to manage business processes that are performed by "knowledge workers." When a knowledge worker opens a new case, one of the first activities the worker performs is creating a plan of activities for the case. At any time during the case prosecution, the worker is enabled to change the plan. Typically the information that identifies the case and its attributes remain unchanged. Just like our approach, case management tools may offer the possibility of defining standard activities that are common to many case instances to expedite the manual task of creating the activity plan.

The concept of storing processes in the information of business entities was first proposed in (Kumaran et al. 2007), a solution framework designed for service delivery management. We extend this framework by a two-tier framework to support process flexibility. In this research, we also extended the business entity-centric approach (Bhattacharya et al. 2007; Liu et al. 2009; Nigam et al. 2003) by introducing dynamic process behavior. Notably, recent business entity research is also exploring declarative approaches to support process flexibility (Cohn and Hull 2009).

CONCLUSION

In this paper, we presented a two-tier data-centric framework that achieves process flexibility by extending business entity process modeling paradigm. The main features of the framework are summarized as follows. First, the framework differentiates process changes by their frequency and provides an elegant solution for frequent ad hoc changes. Volatile business processes are captured in PD entity information models for dynamic change. Second, the framework blurs the boundary between process definitions and instances. Each process instance has its own process definitions as well as information created during runtime. Changes to dynamic behavior can be applied to either schema or instance level. Third, this work can significantly reduce the burden of managing process variations. Definitions of dynamic processes, as data of information models, can be managed by database techniques. Finally, it is the business entity approach to process modeling,

i.e. the unification of information and activity modeling, which makes it possible for business users to modify process instances dynamically.

In future work, we plan to extend this framework to support the scenario where flexibility at PE entity tier is also needed, and to enrich the framework with features to support business users in designing and modifying processes.

REFERENCES

1. Baez, M., Casati, F., and Marchese, M. "Universal Resource Lifecycle Management," 25th IEEE International Conference on Data Engineering (ICDE '09), IEEE Computer Society, 2009, pp. 1741-1748.
2. Basu, A., and Kumar, A. "Research Commentary: Workflow Management Issues in e-Business," *Information Systems Research* (13:1), March 2002 2002, pp 1-14.
3. Bhattacharya, K., Caswell, N.S., Kumaran, S., Nigam, A., and Wu, F.Y. "Artifact-centered operational modeling: Lessons from customer engagements," *IBM Systems Journal* (46:4) 2007, pp 703-721.
4. Cantara, M. "Market Trends: Impact of Business Process Management on Consulting and Development & Integration Services, Worldwide, 2008-2010 " Report No. 163729, Gartner
5. Casati, F., Ceri, S., Pernici, B., and Pozzi, G. "Workflow evolution," *Data & Knowledge Engineering* (24:3) 1998, pp 211-238.
6. Ceri, S., Daniel, F., Matera, M., and Raffio, A. "Providing Flexible Process Support to Project-Centered Learning," *IEEE Transaction on Knowledge and Data Engineering* (21:6) 2009, pp 894-909.
7. Cohn, D., Dhoolia, P., Heath, F., Pinel, F., and Vergo, J. "Siena: From PowerPoint to Web App in 5 Minutes," 6th International Conference on Service-Oriented Computing (ICSOC '08), Springer Berlin / Heidelberg, 2008, pp. 722-723.
8. Cohn, D., and Hull, R. "Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes," *IEEE Data Engineering Bulletin* (32) 2009, pp 3--9.
9. de Man, H. "Case Management: A Review of Modeling Approaches," avaiable at http://www.businessprocesstrends.com/publicationfiles/01%2D09%2DART%2D%20Case%20Management%2D1%2DDeMan%2E%20doc%2D%2Dfinal%2Epdf, BPTrends.
10. de Man, H. "Case Management: Cordys Approach," avaiable at http://www.bptrends.com/publicationfiles/02-09-ART-BPTrends - Case Management-DeMan -final.doc.pdf, BPTrends.
11. IBM "WebSphere Process Server v6.2," 2008.
12. Kumaran, S., Bishop, P., Chao, T., Dhoolia, P., Jain, P., Jaluka, R., Ludwig, H., Moyer, A., and Nigam, A. "Using a model-driven transformational approach and service-oriented architecture for service delivery management," *IBM Systems Journal* (46:3) 2007, pp 513-529.
13. Kumaran, S., Liu, R., and Wu, F. "On the Duality of Information-Centric and Activity-Centric Models of Business Processes," 20th International Conference on Advanced Information Systems Engineering (CAiSE '08), Springer Berlin / Heidelberg, 2008, pp. 32-47.
14. Liu, R., Bhattacharya, K., and Wu, F.Y. "Modeling Business Contexture and Behavior Using Business Artifacts," 19th International Conference on Advanced Information Systems Engineering (CAiSE '07), Trondheim, Norway, 2007, pp. 324-339.
15. Liu, R., Wu, F., Patnaik, Y., and Kumaran, S. "Business Entities: An SOA Approach to Progressive Core Banking Renovation," 6th IEEE International Conference on Services Computing (SCC '09), IEEE Computer Society, Bangalore, India 2009, pp. 466-473.
16. Mulyar, N., Pesic, M., van der Aalst, W.M.P., and Peleg, M. "Declarative and procedural approaches for modelling clinical guidelines: addressing flexibility issues," 5th international conference on Business process management (BPM '08), Springer-Verlag, Brisbane, Australia, 2008, pp. 335-346.
17. Nandi, P., Konig, D., Moser, S., Hull, R., Klicnik, V., Claussen, S., Koppmann, M., and Vergo, J. "Data4BPM, Part1: Introducing Business Entities and the Business Entity Definition Language (BEDL)," available at http://www.ibm.com/developerworks/websphere/library/ techarticles/1004_nandi/1004_nandi.html, IBM developerWorks.
18. Nigam, A., and Caswell, N.S. "Business artifacts: An approach to operational specification," *IBM Systems Journal* (42:3) 2003, pp 428-445.
19. OASIS WSBPEL TC "Web Services Business Process Execution Language Version 2.0," 2007, pp. Available at http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.
20. Pesic, M., Schonenberg, M.H., Sidorova, N., and P., v.d.A.W.M. "Constraint-based workflow models: change made easy," 15th International Conference on Cooperative Information Systems (CoopIS '07), Springer-Verlag, Vilamoura, Portugal, 2007, pp. 77-94.

21. Pesic, M., and van der Aalst, W.M.P. "A Declarative Approach for Flexible Business Processes Management," BPM 2006 International Workshops, Vienna, Austria, 2006, pp. 169-180.
22. Reichert, M., and Dadam, P. "Adept_flex - Supporting Dynamic Changes of Workflows Without Losing Control," *Journal of Intelligent Information Systems* (10:2) 1998, pp 93-129.
23. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., and van der Aalst, W.M.P. "Process Flexibility: A Survey of Contemporary Approaches," 4th International Workshop CIAO! and 4th International Workshop EOMAS, Montpellier, France, 2008, pp. 16-30.
24. Swenson, K. *Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done* Meghan-Kiffer Press, Tampa, Florida, 2010.
25. van der Aalst, W., M. P., Basten, T., Verbeek, H.M.W., Verkoulen, P.A.C., and Voorhoeve, M. "Adaptive Workflow-On the Interplay between Flexibility and Support," 1st International Conference on Enterprise Information Systems (ICEIS '99), 1999, pp. 63-70.
26. van der Aalst, W.M.P., and Jablonski, S. "Dealing with Workflow Change: Identification of Issues and Solutions," *International Journal of Computer Systems Science & Engineering* (15:5) 2000, pp 267-276.
27. van der Aalst, W.M.P., and Pesic, M. "DecSerFlow: Towards a Truly Declarative Service Flow Language," The Role of Business Processes in Service Oriented Architectures, 2006.
28. Weber, B., Reichert, M., and Rinderle-Ma, S. "Change patterns and change support features - Enhancing flexibility in process-aware information systems," *Data & Knowledge Engineering* (66:3) 2008, pp 438-466.