On the Duality of Information-Centric and Activity-Centric Models of Business Processes

Santhosh Kumaran, Rong Liu, and Frederick Y. Wu

IBM T.J. Watson Research Center
19 Skyline Dr. Hawthorne, NY 10532, USA
{sbk,rliu,fywu}@us.ibm.com

Abstract. Most of the work in modeling business processes is activity-centric. Recently, an information-centric approach to business process modeling has emerged, where a business process is modeled as the interacting life cycles of information entities. The benefits of this approach are documented in a number of case studies. The goal of this paper is to formalize the information-centric approach and derive the relationships between the two approaches. We do this by formally defining the notion of a business entity from first principles and using this definition to derive an algorithm that generates an information-centric process model from an activity-centric model. We illustrate the two models using a real-world business process and provide an analysis of the respective strengths and weaknesses of the two modeling approaches.

1 Introduction

The role of information technology (IT) in the enterprise is to support the business operations of the enterprise. Subject Matter Experts (SMEs) document business operations using business process models which prescribe the activities that need to be performed as part of a business operation, the sequencing of these activities, and the input and output data of these activities. While business process models are good for documenting business operations, creating computer programs that support these operations has always been a challenge. Existing approaches to IT-enabling a business process take one of the following two paths:

(1) Business process models are used merely as requirement documents. From these, IT solutions are manually designed and implemented by writing new custom code, or by customizing and integrating legacy applications and packaged software; or
(2) Business process models are automatically converted into workflow definitions which are deployed on workflow engines and augmented with custom code [26].

The first approach leads to a gap between the business process models and IT solutions resulting in poor quality of the IT solutions with respect to their ability to support business processes, poor responsiveness of IT to business process changes, and inefficiency in the overall development processes. The second approach faces a number of difficulties as well, as enumerated below.

Z. Bellahsène and M. Léonard (Eds.): CAiSE 2008, LNCS 5074, pp. 32–47, 2008.

As business processes become complex and large, the workflow approach turns out to be increasingly difficult to implement, as the overall performance of the system degrades substantially and the maintenance of the resulting solution becomes extremely hard. The primary reason for this is that the workflow approach does not lend itself to componentization in a natural way [3].

Advanced features such as backward navigation, event-driven behavior, and conversational client interaction are very difficult to support in workflow models [28]. The cost of supporting these features adds more complexity, which further complicates the first issue. IT solutions often need sophisticated, user-friendly human interfaces which are not readily supported by the workflow approach. Expensive manual tweaking of the code is often needed to integrate function-rich user interfaces with workflow-based backend systems.

In response to this situation, another process modeling paradigm, which models business processes as intersecting life cycles of information entities, has been proposed. Appropriately, this approach is called *information-centric process modeling*. The information entities that are used to describe business processes in this manner have been called various names, including *adaptive documents* (ADoc) [17], *adaptive business objects* (ABO) [21], *business artifacts* [22], and lately *business entities*. In this paper, we will refer to them as business entities.

This new paradigm has been successfully tested through customer engagements. However, several problems remain. First, the concept of business entities is informal with no theoretical underpinnings. Second, in this paradigm, the key is to discover the right information entities that describe the business process. The current practice identifies these entities through intense consulting sessions. Those sessions are time consuming and demand consulting skills that typically are not common. Third, there is a lack of understanding of the relationship between this new paradigm and traditional *activity-centric process modeling* used in workflow management systems [26].

The goal of this paper is to formalize the information-centric approach and present an algorithm for transforming activity-centric process models into information-centric models. We formally define the concept of business entities and use this definition to derive an algorithm for the transformation. We use a real-world example to illustrate this transformation and analyze the respective strengths and weaknesses of the dual representations of a business process.

The remainder of the paper is organized as follows. Section 2 introduces the formal definition of business entities and gives the transformation algorithm for creating information-centric process models from activity-centric process models. A complete example is provided in Section 3 to illustrate this algorithm. Section 4 presents an analysis of the two modeling paradigms. In Section 5, we compare our work with related work. Section 6 concludes with a brief description of future work.

2 Information-Centric Process Modeling

In this section, we give formal definitions of several key notions, including *process scope*, *domination*, *business entity, and activity-centric and information-centric process models*. We start with a brief review of basic concepts of process modeling.

A *business activity* is a logical description of a piece of work that consists of human and/or automated operations and is often realized in conjunction with information processing. A *business process* links business activities by transforming inputs into outputs to achieve a business goal. A set of business processes constitutes a *business function*, provisioning a set of *business services* that defines the external view of the business function. All of the data used by the business function, including the input and output of the business services, form the information domain of the business function. The atomic elements that make up the information domain are called *information entities* (or *entities* for simplicity).

Fig. 1 shows a simple business process. This process clearly describes business activities and their execution sequence for handling a claim. In addition, activities in this process use a set of information entities, for example, *claim* and *loss event*, as the dotted lines indicate. We call Fig. 1 an activity-centric process model, as defined below.

Definition 1 (Activity-Centric Process Models). An activity-centric process model consists of business activities, connectors as control flows describing the execution sequences of these activities, and optional information entities as inputs or outputs of the activities.

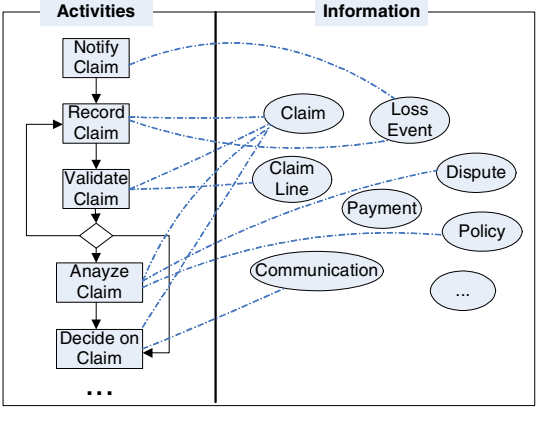

Fig. 1. Activity-Centric Business Process Model – Claim Management

Definition 2 (Process Scopes). A *process scope s* is a group of business processes, together providing a well defined end-to-end function to map input I to output O, i.e. $s: I \rightarrow O, s = \{p_1, p_2, ..., p_n\}, I = \{e_{I1}, e_{I2}, ..., e_{Ix}\}, O = \{e_{O1}, e_{O2}, ..., e_{Oy}\}$, where each p is a process and each e is an information entity.

For example, an end-to-end claim management function involves several business activities such as *record claim* and *validate claim* (see Fig. 1). The input to this process scope may be a loss event and outputs are a set of information entities including a closed claim, outgoing payments, and communication documents with the claimant.

Within a process scope, information entities are created or modified through business activities to produce the desired outputs as defined in the end-to-end function. From an information-centric point of view, information entities influence business

activities in the sense that the execution of an activity is predicated on the availability of the right information entities in the right state [1]. For a set of business activities, there is a corresponding set of information entities that influence the execution of these activities. But there are differences in the degree to which a specific information entity influences the execution of the activities in the set. For example, considering the process shown in Fig. 1, *claim* information entity influences most business activities in the process. On the other hand, the influence of *claim line* is limited to only a subset of activities that are also influenced by *claim*. Therefore, we say *claim* dominates *claim line*, as formally defined below.

Definition 3 (Domination). Information entity e_1 dominates information entity e_2 in a process scope s, denoted as $e_1 \mapsto e_2$, *iff*:

(1) $\forall a \in s$, if $e_2 \in \bullet a$, then $e_1 \in \bullet a$

(2) $\forall a \in s$, if $e_2 \in a \bullet$, then $e_1 \in a \bullet$

(3) $\exists a \in s$, s.t. $e_1 \in \bullet a \cup a \bullet$, but $e_2 \notin \bullet a \cup a \bullet$, where $\bullet a$ ($a \bullet$) denotes the input (output) information entities of activity a.

In other words, e_1 dominates e_2, if (1) for every activity that uses e_2 as an input, e_1 is also used as an input, (2) for every activity that uses e_2 as an output, e_1 is used as an output, and (3) e_1 is used by at least one activity that does not use e_2.

If e_2 is only used as an input in the process scope, this domination is called *referential domination*. With *inclusive domination*, e_2 is used as an output in at least one business activity. A *dominant entity* is one that is not dominated by any other entity in a process scope. Accordingly, a *dominated entity* is an entity dominated by any other entity. The domination relationship is transitive, i.e. if $e_1 \mapsto e_2, e_2 \mapsto e_3$ then $e_1 \mapsto e_3$.

Fig. 2 shows sample domination relationships in an insurance process scope. *Claim* entity referentially dominates *policy* because *policy* is only a necessary input for the processing of *claim* but *policy* itself is not changed within the process scope. *Claim* inclusively dominates *claim line*, *dispute*, and *communication* entities. Some entities, for example, *loss event* and *payment*, are not involved in any domination relationship in this figure. Fig. 2 is referred to as an *entity domination graph,* which describes the domination relationships between entities within a process scope.

From a domination graph, a data model for dominant entities can be derived using containment data modeling [27]. Each dominant entity can be treated as a

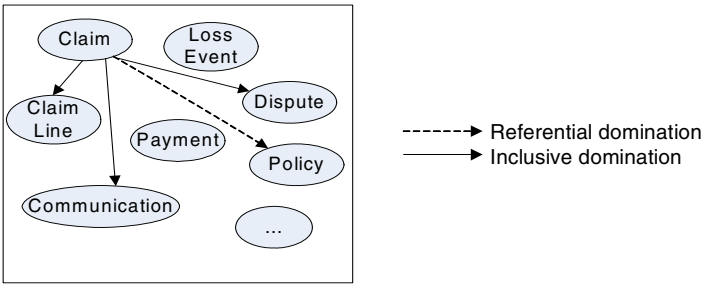

Fig. 2. Examples of Domination

"container", each inclusively dominated entity is a "contained member", and each referentially dominated entity becomes a reference member. A complete example will be provided shortly to illustrate how to derive data models for entities from domination relationships.

Definition 4 (Business Entities). A business entity is a dominant information entity with an associated data model and an associated behavior model in the context of a process scope. The data model describes the data dependencies between the dominant entity and the dominated entities as the dominant entity logically containing the dominated entities. The behavior of the business entity is modeled as a state machine where state transitions are caused by activities acting on the dominant entity.

Fig. 3 shows three business entities in the claim management process scope. Each business entity has a behavior model shown as a state machine. Business entities can be thought of as an abstraction that componentizes the information domain of a business such that the behavior models associated with these components fully capture the business process functionality. Moreover, business entities provide the information context for business activities and processes. Typically, within a process scope, the provided business functions require customer inputs. The customer inputs may initiate an instance of a business entity. The outputs of the processes may be represented as the final state of the business entity and perhaps other business entities created during the processes. Therefore, the business processes are also the process of business entities walking through their lifecycles, from their initial states to their final states.

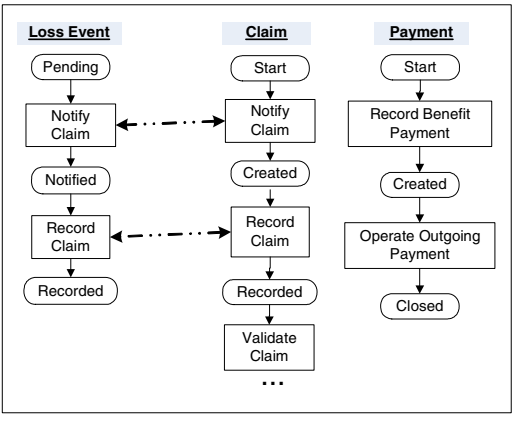

Fig. 3. Business Entities in Claim Management Process Scope

Definition 5 (Information-Centric Business Process Models). An information-centric business process model of a process scope is a set of connected business entities. Two business entities are connected if their behavior models share at least one business activity.

An information-centric process model of a process scope may contain multiple business entities. The lifecycles of these entities are linked through an instance creation pattern or a synchronization pattern [15]. In the creation pattern, an existing business

entity creates a new instance of another business entity as part of a business activity. For example, in Fig. 3, when *notify claim* activity is performed on *loss event*, it creates a new instance of the *claim* entity (i.e., the state of *claim* is changed from *start* to *created*). In the synchronization pattern, two existing business entities exchange information as part of performing a business activity. The example in Fig. 3 shows *loss event* and *claim* exchanging information as part of the *record claim* activity.

Using the concept of domination, we can transform an activity-centric process model into an information-centric process model through the algorithm shown in Fig. 4. The algorithm contains four main steps: (1) Discovering business entities and constructing an entity domination graph purely based on the definitions of domination and business entities; (2) Finding input (*I*) and output (*O*) business entities of each business activity; (3) Creating an output state for each output business entity of an activity; and (4) Connecting an activity to its output business entity states and connecting each output state to the next activity touched by the business entity to construct a state machine preserving the sequence between activities as in the original activity-centric business process model. Note that an activity-centric process model may include control nodes, such as an OR-SPLIT node for alternative decisions, in addition to the activities. However, we disregard such nodes since control flow semantics is implicit in the state machine definition.

This transformation algorithm is very generic. It can be applied to any activity-centric process model that meets two conditions: (1) each activity has input and output information entities, and (2) the process model is connected such that each activity is in at least one path from the start node to the end node. There are no constraints on the format and content of an information entity. However, with respect to the complete and correct definition of an information-centric model of a business process, the information entities should exhibit the following properties.

- *Self-describing*: An information entity is self-describing if it contains metadata that describes its data content. The metadata of a business entity is composed from the metadata of its constituent information entities.
- *Non-overlapping*: Information entities partition data in the information domain of a process scope into disjoint sets.
- *Decoupled*: An entity is decoupled from others in the sense that it can evolve independently of others through business activities. Each information entity is used distinguishably by some business activities. The granularity of information entities requires that any pair of entities should not always be used by the same business activities. If so, these two entities should be merged into one. In other words, the granularity of information entities is determined by business activities.

If the information entities in a process scope possess these three properties, they are *normalized*, formally defined as follows.

Definition 6 (Normalized Information Entities). Let A be the activity set, $A=\{a_1,a_2,...,a_n\}$. Let D denote the information domain of a set of processes, $D=\{e_1,e_2,...,e_m\}$. e_i is a normalized information entity for $1 \le i \le m$, if:

(1) $e_i=(S,V)$, where V is a set of values over attribute set S (*self-describing*)

(2) For any pair of $e_i=(S_i,V_i), e_j=(S_j,V_j)$, $S_i \cap S_j = \varnothing$ (*non-overlapping*)

(3) $e_i \neq e_j$ and $\exists a \in A, s.t.\ e_i \in \bullet a \cup a\bullet, e_j \notin \bullet a \cup a \bullet$ (*decoupled*)

where $i \neq j, 1 \leq i, j \leq m$, $\bullet a$ ($a\bullet$) denotes the input (output) of activity a, and $\bullet a \subset D$, $a\bullet \subset D$

If the original set of information entities does not possess these properties, the resulting business entities may have degraded modularity as their behavior models could be connected at many shared activities. Therefore, it may be necessary to normalize the information entities by clearly defining their data schema and removing data overlap and coupling between them (By Definition 6). In addition, incomplete input and output information entities in the original activity-centric model may result in a poor-quality information-centric process model, characterized by a large number of business entities with interwoven behavior models.

```
INPUT    E     : the information entity set of process scope s, E = {e1, e2, ..., en}
         A     : the activity set in process scope s, A = {a1, a2, ..., am}
         C     : control nodes (AND - SPLIT, AND - JOIN, OR - SPLIT, OR - JOIN, START, END) in S
         Ii (Oi) : the set of activities using ei as input (output)
         ri    : connector associated with e from node ns to nt, ri = (ns, nt, e), R = {r1, r2, ...}
OUTPUT   T     : the business entity set in scope s
         Di    : the set of dominated entities of ei
         Si    : the set of states of business entity ei
BEGIN
for every ei ∈ E                                    / * compute entity domination graph * /
      for every ej ∈ E and j ≠ i
          if ei ↦ ej then save ej to Di
       end for
end for
for every ei ∈ E and Di ≠ ∅                        / * find dominant information entities * /
     if ei ∉ Dj for any j ≠ i then save ei to T
end for
If T  ≠ ∅ then
    for every ei ∈ T
       remove activity a from Ik for any a ∈ Ik and any ek ∈ Di   / * remove dominated items * /
       remove activity a from Ok for any a ∈ Ok and any ek ∈ Di
       remove connector r = (nx, ny, ek) from R for any ek ∈ Di
       for any pair of r = (nx, ny, ei) and r' = (ny, nz, ei) where ny ∈ C / * remove control nodes * /
           remove r and r' from R, create r" = (nx, nz, ei), add r" to R
       end for
       for any connector r = (nx, ny, ei) where nx, ny ∉ C
          create a state sij and save sij to Si       / * create states and update connectors * /
          create connectors r' = (nx, sij, ei) and r" = (sij, ny, ei)
          remove r, and add r' and r" to R
       end for
    end for
END
```

Fig. 4. Transformation Algorithm

3 Example – An Insurance Process Model

In this section, we use a real example to illustrate the transformation of business processes based on the notion of business entities. As an experiment, we examine the process models in IBM Insurance Application Architecture (IAA) [13]. IAA is a comprehensive set of best practice process models for the insurance industry. In general, these process

models are used for analytical purposes, and thus are called analysis models. In practice, system analysts manually customize these analysis models case-by-case in order to implement them. This customization involves the redesign of business activities and data models. We propose to use the transformation algorithm to automatically generate information-centric process models which may then be used to implement service-oriented business process management solutions. Fig. 5 gives an example process for administering property damage claims. In this process, a claim information entity is created when a loss event is reported. This claim is validated and analyzed which could lead to one of three outcomes: rejection of the claim, acceptance of the claim, or postponement of a decision pending additional information. If the claim is accepted, the benefit in this claim is determined and then a payment is issued. If the claim is pended, arrival of additional information leads to another round of processing. In addition to activities and entities, this process contains control nodes (AND-SPLIT, AND-JOIN, OR-SPLIT, OR-JOIN, START and END) and connectors.

In general, an activity-centric process model can contain both data flows and control flows. Wang and Kumar [25] classified two types of constraints in process activities, hard constraints that arise from data dependencies, and soft constraints caused by business rules or policies. In the absence of hard constraints, control flows become necessary to sequence activities. The process model shown in Fig. 5 contains both types of flows, but some of the control flow links are redundant. For example, information entity *claim* is an output of activity *notify claim*, and it becomes an input of activity *record claim*, implying that *record claim* has to be executed after *notify claim*.

We consider this single process as a process scope. This process scope provides an end-to-end function from creating a claim after a loss event is notified to closing the claim and managing payment. Also, our investigation shows that information entities in this process model are self-describing, non-overlapping and decoupled.

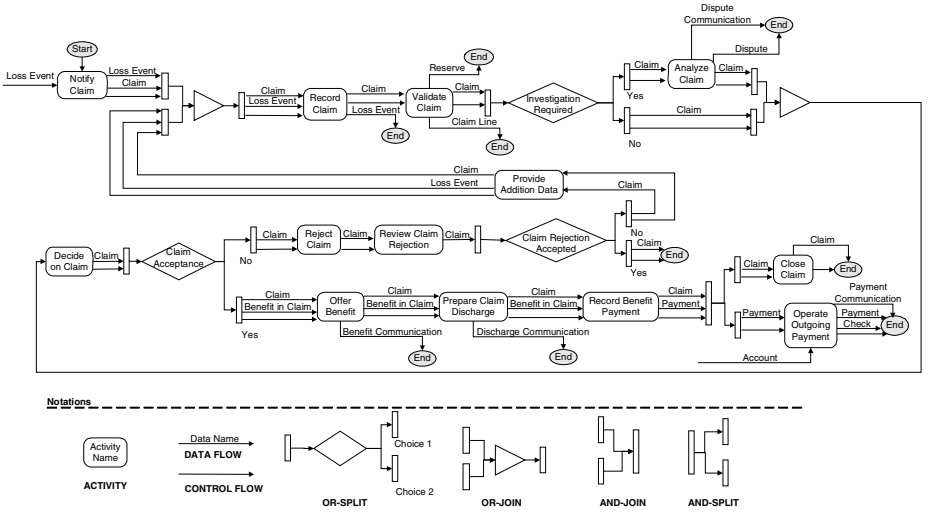

Fig. 5. Administering Property Damage Claim

Based on the concept of domination, we discover three dominant entities involved in this process: *claim*, *loss event*, and *payment*. *Claim* and *payment* business entities have several dominated entities, as shown in the entity domination graph of Fig. 6.

Next, based on the entity domination graph, we can create containment data models [27] for each business entity. Fig. 7 shows the data models. In each data model, the root container is a business entity. The *claim* business entity contains *claim line*, *dispute*, *dispute communication*, *reserve*, and *benefit in claim*, which, in turn, contains *discharge communication* and *benefit communication*. The *loss event* does not contain any entity, but it may have attributes and child items. In this paper, for simplicity, we omit the detailed attributes of each entity. In addition, there may be data relationships between business entities. For example, in Fig. 8, *claim* is created by *loss event* through activity *Record Claim*.

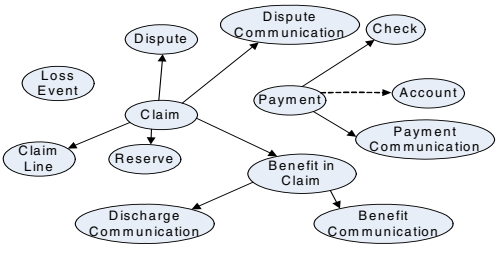

Fig. 6. Business entities – Administer Property Damage Claim

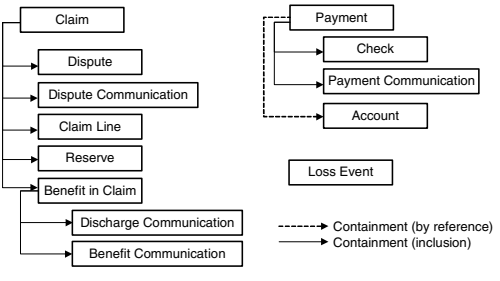

Fig. 7. Data Model of Administering Property Damage Claim

With the discovered dominant entities, we can easily construct the behavior model of each dominant entity and then convert the activity-centric model into an information-centric model using the algorithm shown in Fig. 4. Fig. 8 shows the information-centric process model consisting of three connected state machines. The state machine of *claim* describes the lifecycle of the *claim* business entity and it interacts with the other business entities, *loss event* and *payment*. For example, during *claim*'s state transition from *created* to *recorded*, the *loss event* business entity also changes its state from *notified* to the final state *recorded*. Similarly, *record benefit payment* changes the state of the *claim* business entity from *discharged* to *closed*, while creating a new instance of *payment* business entity.

The business entity behavior models provide a new perspective from which to reason about business activities. Ideally, a business activity should produce some meaningful change to a business entity, resulting in a new milestone in the lifecycle of that entity, which should be monitored and tracked for performance management. If a business activity does not bring such changes, that activity can either be removed or combined with others. Therefore, business entities actually provide guidelines for determining the right granularity of activities. For example, in Fig. 8, activity *offer benefit* is likely to notify a claimant without providing any actual changes to the *claim* business entity. This activity may be merged with the activity *prepare claim discharge*. Also, if there is a significant overlap between two behavior models, we may need to re-design the activities to decouple the business entities. Note that the algorithm in Fig. 5 does not give any state naming convention. One can name business entity states based on results produced by activities, as exemplified by Fig. 8.

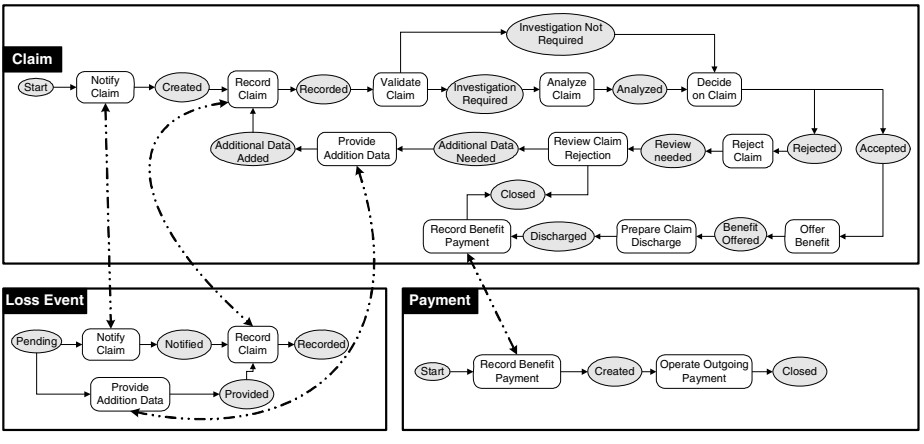

Fig. 8. Information-centric process – Administering Property Damage Claim

Compared with the original model in Fig. 5, the model in Fig. 8 has the following features. First, this information-centric process model provides better understandability because the introduced business entities highlight the focus of the process. Obviously, the process mainly deals with *claim*, tracking its behavior through its end-to-end lifecycle from creation to closure. Understandability is further improved by the decomposition of the process into three streamlined state machines each with fewer activity nodes. Empirical evidence shows that model size and average connector degree significantly affect the understandability of process models [19].

Second, the information-centric model hides IT implementation details and only describes the business entities that each business activity acts on. In Fig. 5, each business activity has detailed input and output entities. However, the information-centric model only specifies the business entities that each activity reads, updates or creates. Through the data model, an activity is able to retrieve information of the dominated entities. In practice, data access can be defined as data views by user roles and by business activities. By adding data access details for each activity, we can convert the information-centric process back to the activity-centric model. However, data access

details are an IT implementation issue. We prefer to delay the definition of data access until implementation for two reasons. First, during implementation, precise data access may be defined at the attribute level instead of the entity level. Therefore, the input and output specifications in terms of entities in Fig. 5 are not sufficient. Second, in reality, data access varies with each implementation. Without data access details, an information-centric process model can be easily adapted into different process scopes. As evidence, IBM Insurance Application Architecture [13] contains seven process models, each describing a particular type of insurance claim, such as medical expense claim, life claim, and auto claim. Our analysis shows that these models can be transformed into the same information-centric process as in Fig. 8, with slightly different data graphs. For example, in the process *administering auto claim*, *benefit in claim by insurer* is one of the final output entities, instead of *payment*.

Finally, using the Model Driven Business Transformation Toolkit [17], we can generate business applications automatically from the information-centric business process models. The development and implementation time can be greatly reduced. Also, this direct transformation from business process models to IT solutions reduces the gap between business and IT.

4 Analysis and Discussion

The domination concept reveals the deep structure of the information domain of a business function. This deep structure is represented as entity domination graphs in the form of *directed acyclic graphs (DAG)*, with the dominant information entities at the source nodes of the DAGs serving as the driving force for the process flows that constitute the business function. The dominated information entities form the non-source nodes of the DAGs and play a subsidiary role in the execution of business activities. Usually, the dominated entities are created during the processing of the dominant entity and their existence depends on the dominant entity. But the dominated entities do play an important role in the lifecycle of the dominant entity. It is analogous to the growth of a tree in nature, with new branches being added as the tree progresses through its lifecycle. For example, in the insurance claim process discussed earlier, *dispute*, *claim line* and *benefit in claim* are created during the processing of the *claim* entity and their existence depends on the *claim* entity. And when a *claim* is *accepted*, one can expect to see *benefit in claim* added to the data graph associated with *claim*.

An intuitive explanation of the domination concept can be derived from the Pareto principle which states that, for many events, 80% of the effects come from 20% of the causes. When applied to business process analysis, we observe that a few information entities serve as key drivers of the flow of most activities. Using our algorithm, we are able to select the dominant entities and model their behaviors, thus leading to significant reduction in model complexity and better understanding of business operations. For example, we have observed that among 320 information entities used in IBM Insurance Application Architecture [13], only 90 qualify as dominant entities. We can also view domination as a special association rule mining [2] for discovering antecedent and consequent information entities and establishing associations between them.

The algorithm presented in this paper leverages the domination concept to transform an activity-centric business process model into an information-centric model. There are several advantages to be gained from creating such an information-centric model as discussed below.

As we have seen above, an activity-centric model of a business process enumerates all the activities in the process and defines a control flow and a data flow over these activities. As the processes grow in size and complexity, it becomes increasingly difficult to understand the business behavior using these models [3]. The traditional approach to dealing with this complexity is to resort to a hierarchical representation of business processes. It has been shown that static, hierarchical representations of business processes are not conducive to in-depth analysis and prediction of the behavior of the system under dynamic conditions [20]. The information-centric modeling approach presents an attractive alternative as it helps to analyze and predict system behavior using the lifecycle models of a few business entities in a flat structure.

Designing user-friendly human interfaces from activity-centric models to drive business process execution has been known to be a challenge [28]. Business users are knowledge workers who need contextual information to make the decisions needed in performing a business activity. This contextual information is not part of the activity-centric models of business processes and thus it becomes hard to design effective human-interfaces from such models to drive process execution. Information-centric models help in this regard since the deep structure of the information domain represented as a DAG holds this contextual information. The lack of contextual information in activity-centric process models has business-level implications as well. Since such models emphasize control flows and look at information usage only within the context of a single activity, business actors tend to focus on "what should be done instead of what can be done", hindering operational innovations [1,12].

The issue with interface design to business process execution systems gets more complex when conversational interfaces are involved. In a conversational interface, the input to a process has to be defined and refined incrementally as part of the process execution [28]. The unpredictability of the input makes it hard to precisely determine the execution order of activities during the modeling phase. For example, Zimmermann et al. [28] reported difficulties in designing conversational interfaces in a large order management process using BPEL [9]. However, an information-centric model of a process naturally supports such constraints because it models the business process as business entity lifecycles, and the set of client interactions that move a business entity though its lifecycle becomes a conversation.

Information-centric models of business processes have important implications with respect to implementations of business process management systems. Activity-centric models such as BPEL [9] and Event-driven Process Chains (EPC) [24] are executed by flow-driven workflow systems. In such systems, the process execution complexity can be classified into three types: control-flow complexity, data-flow complexity, and resource complexity [10]. Typically, the complexity increases rapidly as the number of control nodes (e.g. Pick, Flow and Switch nodes in BPEL) increases [10], thus severely impacting the scalability of this approach. In contrast, information-centric models enable the execution of the process as a set of communicating finite state machines, which significantly improves process execution efficiency [21].

The advantages of a modular design in building complex systems are well known [4], but the challenge lies in identifying the right modules. Business entities provide a natural way to modularize a business process management system. Each module implements the behavior of a business entity as a state machine and manages the information entities associated with that business entity. This approach to modularization leads to a new way to decompose business processes and implement them using service oriented architecture (SOA) [11]. With increasing industrialization of services, companies tend to decompose their business processes for selective outsourcing, selective automation, or restructuring to create decoupled composite business services which may be flexibly integrated to support end-to-end business processes [16]. Therefore, there is a need for a systematic way for companies to analyze and decouple their processes. Our algorithm does precisely this analysis and decoupling. Intuitively, each business entity along with its state machine defines a decoupled, composite business service [11]. In addition, service interfaces can be derived from the connections between business entities and the communication between these entities can be implemented as service invocations. For example, in Fig. 8, a company can define the *claim* portion as the core process which drives customer value, but outsource the *payment* portion. Both *claim* and *payment* may now be implemented using business entities as composite business services and the end-to-end claim business process can be realized via service invocations on these entities. The details about implementing information-centric business processes using SOA principles can be found in [8].

5 Related Work

Recently, information-centric modeling has become an area of growing interest. Nigam and Caswell [22] introduced the concept of business artifacts and information-centric processing of artifact lifecycles. Kumaran et al. [17] developed adaptive business documents as the programming model for information-centric business processes and this model later evolved into adaptive business objects [21]. Further studies on business artifacts and information-centric processes can be found in [6, 7, 8, 15]. [6] describes a successful business engagement which applies business artifact techniques to industrialize discovery processes in pharmaceutical research. More engagements using information-centric modeling can be found in [8]. Liu et al. [15] formulated nine commonly used patterns in information-centric business operation models and developed a computational model based on Petri Nets. [7] provides a formal model for artifact-centric business processes with complexity results concerning static analysis of the semantics of such processes. While previous work mainly focuses on completing the framework of information-centric process modeling from theoretical development to practical engagements, our work bridges the gap between activity-centric and information-centric models and shows the duality between them.

Other approaches related to information centric modeling can be found in [1, 25]. [1] provides a case-handling approach where a process is driven by the presence of data objects instead of control flows. A case is similar to the business entity concept in many respects. In [25], document-driven workflow systems are designed based on data dependencies without the need for explicit control flows. In this paper, in addition to tracking the behavior of data objects, we are interested in their deep structure.

Another related thread of work is the use of state machines to model object lifecycles. Industries often define data objects and standardize their lifecycles as state machines to facilitate interoperability between industry partners and enforce legal regulations [23]. [18] gives a technique to generate business processes which are compliant with predefined object lifecycles. Instead of assuming predefined business objects, our approach discovers business entities from process models and then defines their lifecycles as an alternative representation of process models. In addition, event-driven process modeling, for example, Event-driven Process Chains (EPC) [24], also describes object lifecycles glued by events, such as "material in stock". Our approach in this regard is also event-driven, as each business entity state can be viewed as an event. However, EPC is still an activity-centric approach as objects are added to functions as inputs or outputs and an event can be defined concerning a group of objects.

Some other notable studies that are related to our work are in the area of process decomposition and service oriented architecture. Basu and Blanning [5] presented a formal analysis of process synthesis and decomposition using a mathematical structure called metagraph. This study gives three useful criteria: full connectivity, independence, and redundancy, for examining process synthesis and decomposition. However, the metagraph approach is not applicable to process models, such as the one shown in Fig. 6, which contains many cycles when formulated as a metagraph.

6 Conclusion and Future Work

In this paper, we have presented an approach to discovering business entities from activity-centric process models and transforming such models into information-centric business process models. An algorithm was provided to achieve this transformation automatically. We illustrated this approach with a comprehensive example and tested it using reference processes from the insurance industry.

Our approach provides an alternative way to implement activity-centric process models. Instead of transforming them into BPEL processes or workflows, our approach generates information-centric models from them and implements these models using the Model-Driven Business Transformation Toolkit [21], thereby improving both the understandability of process models and their execution efficiency. Additionally, this approach provides a new way to decompose business processes into connected business entities, each of which can be implemented as a business service using SOA principles.

We are currently developing a tool based on this algorithm and applying this approach to best practice processes in the IT service delivery industry. We expect our future work to extend this algorithm to other types of process models, including BPEL and EPC models. Another research direction is to relax the concept of domination so that business entities can be discovered from models with incomplete or incorrect specifications of input or output information entities.

Acknowledgments. The authors thank Kumar Bhaskaran, David Cohn, Anil Nigam, John Vergo and other colleagues for their helpful discussion and comments.

Reference

1. Aalst, W.M.P., Weske, M., Grunbauer, D.: Case handling: a new paradigm for business process support. Data and Knowledge Engineering 53, 129–162 (2005)
2. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Database. In: Proceedings of ACM-SIGMOD 1993, May 1993, pp. 207–216 (1993)
3. Alonso, G., Agrawal, D., El Abbadi, A., Mohan, C.: Functionalities and Limitations of Current Workflow Management Systems. IEEE Expert 12, 5 (1997)
4. Baldwin, C.Y., Clark, K.B.: Design Rules. The Power of Modularity, vol. 1. MIT Press, Cambridge (2000)
5. Basu, A., Blanning, R.W.: Synthesis and Decomposition of Processes in Organizations. Information Systems Research 14(4), 337–355 (2003)
6. Bhattacharya, K., Guttman, R., Lyman, K., Heath, I.F.F., Kumaran, S., Nandi, P., Wu, F., Athma, P., Freiberg, C., Johannsen, L., Staudt, A.: A model-driven approach to industrializing discovery processes in pharmaceutical research. IBM Systems Journal 44(1), 145–162 (2005)
7. Bhattacharya, K., Gerede, C., Hull, R., Liu, R., Su, J.: Towards Formal Analysis of Artifact-Centric Business Process Models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 288–304. Springer, Heidelberg (2007)
8. Bhattacharya, K., Caswell, N., Kumaran, S., Nigam, A., Wu, F.: Artifact-centric Operatonal Modeling: Lessons learned from engagements. IBM Systems Journal 46(4) (2007)
9. BPEL, Business Process Execution Language for Web Services, version 1.1, joint specification by BEA, IBM, Microsoft, SAP and Siebel Systems (2003)
10. Cardoso, J.: Complexity Analysis of BPEL Web Processes. Software Process: Improvement and Practice Journal 12, 35–49 (2007)
11. Ferguson, D.F., Stockton, M.L.: Service-oriented architecture: programming model and product architecture. IBM Systems Journal 44(4), 753–780 (2005)
12. Hammer, M.: Deep change: How operational innovation can transform your company. Harvard Business Review, 84–93 (April 2004)
13. IBM Insurance Application Architecture (IAA), version 7.1 (2004), http://www-03.ibm.com/industries/financialservices/doc/content/solution/278918103.html
14. Kumaran, S.: Model Driven Enterprise. In: Proceedings of Global Integration Summit 2004, Banff, Canada (2004)
15. Liu, R., Bhattacharya, K., Wu, F.Y.: Modeling Business Contexture and Behavior Using Business Artifacts. In: Krogstie, J., Opdahl, A., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 324–339. Springer, Heidelberg (2007)
16. Karmarkar, U.: Will You Survive the Services Revolution? Harvard Business Review 82(6), 100–107 (2004)
17. Kumaran, S., Nandi, P., Heath, T., Bhaskaran, K., Das, R.: ADoc-oriented programming. In: Symposium on Applications and the Internet (SAINT), pp. 334–343 (2003)
18. Küster, J.M., Ryndina, K., Gall, H.: Generation of Business Process Models for Object Life Cycle Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 165–181. Springer, Heidelberg (2007)
19. Mendling, J., Reijers, H.A., Cardoso, J.: What Makes Process Models Understandable? In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 48–63. Springer, Heidelberg (2007)
20. Modarres, M.: Predicting and Improving Complex Business Processes: Values and Limitations of Modeling and Simulation Technologies. In: Winter Simulation Conference (2006)
21. Nandi, P., Kumaran, S.: Adaptive business objects – a new component model for business integration. In: Proceedings of International Conference on Enterprise Information Systems, pp. 179–188 (2005)

22. Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification. IBM Systems Journal 42(3), 428–445 (2003)
23. Ryndina, K., Küster, J.M., Gall, H.: Consistency of Business Process Models and Object Life Cycles. In: Kühne, T. (ed.) MoDELS 2006. LNCS, vol. 4364, pp. 80–90. Springer, Heidelberg (2007)
24. Scheer, A.W.: Business Process Engineering: Reference Models for Industrial Enterprises, 2nd edn. Springer, Heidelberg (1997)
25. Wang, J., Kumar, A.: A Framework for Document-Driven Workflow Systems. In: Proceedings of Business Process Management, pp. 285–301 (2005)
26. WfMC, The Workflow Reference Model, Issue 1.1, Document Number TC00-1003, Work-flow Management Coalition, Winchester, UK (1995)
27. Whitehead, E.J.: Uniform comparison of data models using containment modeling. In: Proceedings of the thirteenth ACM conference on Hypertext and hypermedia, Maryland, USA, pp. 182–191 (2002)
28. Zimmermann, O., Doubrovski, V., Grundler, J., Hogg, K.: Service-oriented architecture and business process choreography in an order management scenario: rationale, concepts, lessons learned. In: Proc. of the 20th SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 301–312 (2005)