From Data-Centric Business Processes To Enterprise Process Frameworks

Jianwen Su
Department of Computer Science
UC Santa Barbara, USA
su@cs.ucsb.edu

Lijie Wen
School of Software
Tsinghua University, P.R. China
wenlj@tsinghua.edu.cn

Jian Yang
Department of Computing
Macquarie University, Australia
jian.yang@mq.edu.au

***Abstract*—**Conceptual elevation of data in business process modeling was first formulated in 2003. The research community responded to this new idea enthusiastically. In the past decade, there have been numerous research activities concerning the interactions between business processes/activities and data in many aspects of business process management. Many of the advancements have or will have impacted on design/modeling, analysis, implementation of business processes in practice. However, there is still a lack of techniques for developing a suite of "interrelated" processes realizing a business service. Interrelated processes are a cluster of processes that share data and other resources, are collectively constrained by regulations and policies, influence KPIs as a group. Many current practical applications of business workflow systems are in urgent need for tools and techniques for process clusters. In this paper, we formulate a broad notion of an "Enterprise Process Framework" (EPF) to address this need and outline several interesting research challenges arising from EPFs.

I. Introduction

Business services contribute 60-80% of GNP to most of worlds top economies [1]. A *business service* is typically accomplished by a set of interrelated business processes. Arising from enterprise software applications less than 2 decades ago, service-oriented computing aims to wrap functionalities provided by software systems into *software services* that are easily interact-able via network to help business processes. With research efforts by the software and BPM communities and standards such as WDSL, REST, BPEL, WS-CDL, etc. developed by various organizations (W3C, OMG, OASIS, etc.) in the past two decades, the bridge between software services and (collaborative) business processes is relatively clear. In this paper, we explore the relationships between business services and business processes.

A business process is an assembly of tasks performed by human participants or by computing and other devices to accomplish a business objective typically within the scope of a business service. The processes serving a common business service must be interrelated by *sharing data*. Collaborative processes share data during execution, but in general data sharing may happen after completion of processes.

Earlier business process models focus mostly on activities (or tasks or software services) and their control flow. The exclusion of data in these models limits applicability. A fundamental advancement was made when data was paid the same amount of attention as activities and incorporated into process modeling [25][5]. The inclusion of data immediately opened the door to a set of interesting and challenging research problems relevant to practice [11]. The ACSI project [2] and the CMMN standard [23] are notable developments inspired by the issues concerning data in processes. Currently issues concerning both processes and data have been studied in a wide range of topics including process mining (e.g., [12]), data management (e.g., [30]), and others.

Inclusion of both activities and data is very helpful for BPM practitioners but merely a first step. Very often many business services are accomplished by a group of business processes instead of individual processes. For example, obtaining a new Internet or wireless communication service subscription may include processes such as *order*, *shipping*, installation *scheduling*, client site *installation*, *payment*, *cancellation*, order *modification*, etc. A government agency in a Chinese city provides 20-30 services for residents concerning real estate transactions has about 500 business processes, each service is accomplished by more than a dozen processes [45].

In this paper, we argue that "clusters" of interrelated business processes collectively realizing business services deserve technical investigations from many aspects. Indeed, with data access modeled in processes, it is feasible to study the relationships between processes via the common data they access, e.g., a customer order completes, and the customer later requests to cancel. We illustrate with examples the need for considering processes as clusters, formulate and discuss a new concept of an "enterprise process framework" (EPF). We also speculate a few research challenges concerning EPFs.

This paper is organized as follows. Section II briefly reviews data-centric business processes and a few relevant developments. Section III provides examples to demonstrate the need for concerning process clusters. Section IV introduces the main notion of an EPF. Section V enumerates a few technical challenges.

II. Data-Centric Business Processes

In this section we provide a brief outline of research and development activities concerning incorporation of data into business process management (BPM) in the recent decade. The discussions center around the main ideas, the application and research problems, and the technical development.

Data plays a critical role in essentially all workflow applications. Clearly data with the associated semantics is a fundamental piece in formulating workflow semantics, defining what actually a workflow performs and how its actions are related to the environment/context. Consequently, recorded as data are progresses of individual workflow executions (instances or enactments) including execution status, resource usage and status, and correlations with other workflow instances. Moreever, executing a workflow (instance) would generates additional data for a variety of reasons such as monitoring for performance or business concerns, auditing, compliance checking, etc. Finally, even workflow schemas and configurations for the environment can be viewed as data so that they can be managed, queried, analyzed, extended (or evolved) for better serving the application needs.

Not surprisingly, workflow models indeed emerged that support a much tighter coupling of the data being manipulated and the sequencing of activities that perform those manipulations. An initial notion was originally introduced as *business artifact* by Nigam and Caswell [25] to capture the essential properties of key conceptual objects which evolve as they move through a workflow. There are two essential components in a specification of a class of business artifacts (instances): (i) a data schema (or information model) for holding information about the artifacts as they move from creation, through the workflow, and in usual cases, to archival storage, and (ii) a lifecycle schema that describes how and when tasks (or services) might be invoked on the artifacts as they move through the workflow. A prototypical example of a business artifact is the notion of "air courier package delivery", whose data schema can hold information about a package including sender, receiver, the steps occurring in transport, and the billing activity, and whose lifecycle would specify the possible ways that the delivery service might be carried out. Indeed, the typical package tracking information provided by commercial delivery services can be understood as providing a subset or "view" of the data value associated with the delivery artifact as it progresses through the courier's workflow, along with an abstracted view of the lifecycle, shown as the likely steps leading to completion of the enactment.

A formal model for business artifacts was developed in [5], where the data schema of a business artifact is represented by a set of states and a set of typed attributes, activities as semantic services with *input*, *output*, a *precondition*, and conditional *effects* (IOPEs). *Business* (or *condition-action*) *rules* are then used to assemble the services together. Thus, a business process model consists of business artifacts, services, and rules. This technical framework treats both business informational perspective (as manifested in business artifacts) and control flows (as manifested by business services and rules) as equally important. A 4-step data-centric business process design methodology was then formulated [6] in which critical business artifacts are first identified and along with the key stages of their lifecycles, a business operations model is then designed that includes both components of the business artifacts and event-condition-action (ECA) rules, in the last two steps the conceptual flow diagram is designed and workflow realization is acquired for the targeted business process.

The declarative feature in the technical modeling language of [5][6] is desirable, however *states* for artifacts are atomic and there is a mismatch between instantaneous state transitions in the formal model (adopted from typical state machine formalisms) and actual business processes in which state transitions are made while activities are performed. Also, the ECA rules lean closer to execution than specification. These weaknesses are overcome in the Guard-Stage-Milestone (GSM) modeling language [20][21][10]. A GSM process is modeled in the form of artifacts, each of which has a data schema with data attributes and state attributes, and a lifecycle. A lifecycle is formulated based on *guards*, *stages*, and *milestones*. A guard defines when a particular stage may become active, a milestone defines when a stage is completed (e.g. a business goal is reached). A stage can be *atomic* (containing a service to be executed) or *composite* (contains other stages). A composite stage groups stages that are executed in order to achieve a common goal collectively. Although GSM semantics [10] is based on ECA rules, the change from instantaneous state transition to lasting stages to accomplish activities and other subgoals, and the incorporation of conditions into stage activation and completion make GSM processes easier to understand for business applications, a key reason that GSM was the basis for a recent case management standard CMMN [23].

The artifact-centric approach to business process has spawned a very active research program in formal verification in the context of data and process considered together [8][7]. This area is well motivated, because it is often the case that process executions must satisfy certain conditions derived from laws, governmental policies, institution policies, contextual requirements, etc. To apply formal verification approaches, it is typical to express these conditions in a temporal logic language. The verification problem is to check if the temporal

properties are satisfied by every process execution. When a business process schema (model) does not include data explicitly, a usual choice of the language for properties is propositional linear temporal logic (PLTL). In this case, checking if every execution of a business process schema would satisfy a given PLTL property can be done by, e.g., model checking. When data is modeled explicitly in the business process schemas, first-order (FO) LTL language and μ-calculus have been used for specifying properties of process execution. There are two issues, one concerns the modeling of data and the other concerns complexity of the verification problem. In some studies the data in a single artifact instance has been modeled as a set of scalar attributes and possibly with ordered domains and even with arithmetic operations [14][9], alternatively, [3] permits artifacts whose attributes hold entire relational databases. Both approaches provide insight into the impact of structured data on verification. In general, checking if every possible execution allowed by a business process will satisfy a given FO-LTL property is undecidable. The research has discovered restrictions on artifact-centric business process models that yield decidable verification of temporal properties.

For a variety of reasons such as changes of business policies, operational routines, and the environment, business processes often need to be changed or restructured [19]. Declarative process specifications make changes easier, but are far from solving problem of change or evolution. A common feature of declarative languages is that a specification is often composed from individual elements; changes of elements or composition rules appear to be easy. However, a fundamental difficulty is whether the changes can be "pushed down" to the implementation level consisting of data accesses. The study reported in [45] developed a novel mechanism for translating process changes to data access changes in order to handle ad hoc and just-in-time changes at runtime. Specifically, Runtime execution variations are specified as execution modification rules consists of constructs *retract*, *skip*, *add*, and *replace*. The change rules are applied to execution at runtime to achieve the desired changes while maintaining consistency of data and execution.

An important point learned from the above described study on dynamic changes is that data management during process execution must be systematically planned, as opposed to accessing data typically in enterprise databases at will. Indeed, Jointfounder Info. Tech. Inc. (`http://www.hzzfxx.com/`) ran into a roadblock when attempting to develop a Business-Process-as-a-Service (BPaaS) framework as a cost saving mechanism for its numerous clients that are government agencies for housing management [31]. At the core of the technical challenge is how to manage data for the clients consistently while cleanly separating data from different clients during process execution.

To clearly understand the issues in managing data accesses during process execution, it is helpful to know the types of data usually present in a process management system. References [29][32] defined five classes of data involved during an execution of a business process: (1) specification of *process models* that serve as road maps for execution, (2) *business data* (e.g., about the applicant, shopping cart, total cost), (3) *execution status* (e.g., the initial review is completed and an auxiliary background check process instance is initiated), (4) *correlation* relations of the current instance and its collaborative instances, and (5) *resource* information and *status* (e.g., credit report request is made but the report has not arrived).

Along the data dimension, business process modeling languages can also be placed into five levels [29][32]: (A) *Data agnostic* models essentially ignore all data other than class 1 but focus on activities possibly with arrangement of activities with execution status partially available and implicit through the execution semantics (examples include Petri nets, workflow nets, UML activity diagrams, and BPMN). (B) *Data aware* models use variables to hold a small subset of business data (class 2) and are capable of specifying detailed control flow logic for processes (e.g., BPEL and YAWL [36]). (C) *Storage aware* models also include the concept of persistent data stores and modeling of enterprise data (e.g., jBPM, UML with both class and activity diagrams). (D) *Artifact-centric* models include conceptual data modeling for each process, which is a clear improvement. Both data/storage aware and artifact-centric models lack the support for mappings between persistent data in the enterprise database and process data are defined through e.g., SQL expressions (jBPM) or Java Hibernate. (E) *Semantic process* models supports the modeling of not only data of all five classes and provides the ability to specify mappings between all data in business processes and the enterprise database [30].

A closer examination of data management especially data modeling, storage, accesses, and control functions in process/workflow management systems reveals that the popular "textbook" software architecture (as presented in [37][42]) lacks essential functions of sound data management and is thus seriously flawed [29]. A key observation made in [29] is the over-simplified method of using a persistent database system *inside* the process/workflow management system for some data in the five classes and accessing/updating typically business data and also data in the five classes in the enterprise database system *outside* the process/workflow management system. (While the union of the inside and outside data covers all data needed for process execution, they are not necessarily

disjoint.) This distributed placement of data is the main obstacle to a sound BPaaS framework and a cause to many difficult problems in BPM [29].

A solution was developed based on a universal artifact that captures all necessary data for execution and allows a process/workflow engine to manage execution without the need to access data outside of the universal artifact [34][31][32]. With universal artifacts, a revised process/workflow management system does not need to maintain persistent data internally [29][32]. An interesting framework and prototype named SeGA was developed in [34][31] (also [32]). SeGA can wrap an existing process/workflow engine into "stateless" service provider and serve as a broker between the process/workflow engine and the environment. The work gives an evidence that process/workflow management systems with or even without artifact-centric models can be lifted to systems that support semantic process models.

SeGA requires a universal artifact repository so that the dispatcher can fetch universal artifacts. In general, an enterprise stores the data in one or more persistent data stores (e.g., relational databases). A general approach of data mappings can bridge the relationships between artifacts and databases [30]. These mappings allow updates to be propagated between artifact instances the database (in both directions) [30]. The SeGA framework significantly improves the support for collaborative business processes (execution and management) and makes it easy to turn BPM systems to operate in the BPaaS fashion. Recently, Jointfounder Inc. successfully implemented a BPaaS prototype system based on the SeGA framework to provide for their clients.

In summary, the research community has investigated a range of problems concerning artifact-centric business processes for more than a decade and made a significant progress.

III. Challenges From Applications

In this section we argue that existing results and techniques concerning data-centric business processes are insufficient to address application difficulties. In fact, "insufficiency" here means that the existing formulations of the problems studied in the literature are too weak to define some of the application difficulties. We illustrate the difficulties using examples that are abstracted from the real life examples.

Example 1: *Purchase, tax refund, and return*

A tourist named Sandy from the US purchased a dress from MaxCo Sydney in a recent trip to Australia. Before boarding the return flight Sandy applied and successfully got a tax refund at the Sydney airport for the purchase. Upon arriving home, Sandy found that the dress was not in the right size. She then asked her friend Jaimie who lives in Sydney to help for an exchange for the right size. But the attempt failed since the dress of the size was sold out. Jaimie consequently asked for a return and refund. When Sandy eventually received the refund from the MaxCo, to her surprise, she got a full refund instead of the amount she paid minus the tax refund she already received. ∎

Example 2: *Purchase, tax refund, and again*

Zack was born and grew up in Sydney, Australia. He now attends a college in Huddersfield, UK. During a summer break he bought his first iPhone—an iPhone 6 —from an Apple store in Sydney. Since he lives and will use it overseas, Zack requested and got a tax refund at the Sydney airport. After getting back to his college, Zack found that the screen size is less ideal for his daily use for email and note-taking. So on the next trip back to Sydney, he went to the Apple store and exchanged his iPhone 6 for an iPhone 6 Plus and paid the exact price difference between the two phones. Before heading back to college, Zack again got another tax refund for the full price of an iPhone 6 Plus instead of the refund for the price difference. After he got back, he decided to donate the extra refund to a local charity. ∎

In Examples 1 and 2, two independent business processes are involved, namely *purchase* and *tax return*. The laws and regulations expect the two processes run independently in a sequential order and on the same purchase(s) made. There are no issues when they are carried out in the expected manner. When the expected conditions are not met, inconsistency could occur as illustrated in the two examples. An interesting observation is that if we examine processes (and their models) *individually* in isolation, each of them is (almost) perfectly compliant and meets the business goals, and their instances should run correctly according to their models. Inconsistency problems could only be found when two processes (models) are inspected *jointly*. In these scenarios, the processes belong to two different organizations and are typically not going to be verified together. Nevertheless, this presents a challenge.

There are application examples similar to the scenarios in Examples 1 and 2, i.e., individually "correct" processes belonging to different organizations will not be compliant when executing together. A keynote speaker with travel expenses covered by a conference would also seek reimbursement in her own university; a subscription (process) followed by a cancellation (process) but somehow the cancellation didn't update the appropriate data record and the customer got a surprise invoice. Of course there are procedures one can develop to guard against such inconsistencies, e.g., government approved receipts in China. Our concern here is how to find such inconsistency potentials that we call *anomalies*.

Example 3: *Degree program change*

A new degree program Digital Business is offered in 2017 that replaces the old program eBusiness. Students in the existing eBusiness degree has offered the option to either stay with the old eBusiness or graduate with the new Digital Business degree. The differences between the old and new degrees are in the core units in second and third year of study. A third year student Wendy decides to switch to the new degree program. So she needs to take a core third year unit that requires a second year unit as a prerequisite which she has not completed. The university later recognizes the issue and change the prerequisite rule so that one of her second unit can be used as a prerequisite for the third year core unit in the new degree program. ∎

In the above example, the degree process and unit process are independently developed. When a degree process changes, it may affect other processes including unit process. By establishing *dependencies* between these processes, we would be able to identify the change impact on other processes caused by one process change.

More generally, there are demands from application development to consider a group of processes as a whole in the design and development. Company J sells software to housing management agencies to help managing approval processes related to real estate related licensing, titles, permits, etc. J does very well and attracted many potential new clients. However, J is facing an enormous pressure to serve its new clients. A major reason is that for each new client (an HMA), software engineers must devote a lot of time to modify and adapt the current software to the local environment of the city: local laws and regulations, data management within the agency as well as other agencies in the city. As another example, vendor H provides software for tele-communication service providers with many of the providers are actually in the same franchise. Again, there is a challenge to deliver the software to a new client almost every time. One aspect of the re-design and re-development in J and H involves moving fragments from processes to processes.

A challenge arising from the discussions of processes in applications is how to formulate technical models so that techniques can be developed and lead to solutions to application problems.

IV. Enterprise Process Frameworks

In this section we introduce the concept of an "enterprise process framework" (or EPF) consisting of a set of processes that collectively accomplish a business service. We also discuss briefly advantages of EPFs, differences between EPFs and existing concepts including collaborative business processes, and some known techniques that are relavent to EPFs.

In many business applications, a business service is often a single product provided to the user, and the service is often composed of several steps, each of which is a business process.

Example 4: *Subscription of Internet service*

Consider an example of ordering an Internet service. The service includes processes such as ordering by the user, scheduling of installation, delivery and installation, billing, payment processing, and additional processes such as customer service, cancellation, etc. The collection of processes constitutes the service, i.e., only when a "reasonable" set of processes finish, the service completes. Here a reasonable set could include a payment (process), or a cancellation, or contain three processes: a payment, a cancellation, and a refund. In present practice, the concept of a service such as the above reasonable sets is often partially represented in process models, and partially in documentation. The ability to precisely specify such a concept is desirable for business services. ∎

Conceptually, an *enterprise process framework* (*EPF*) for a business service consists of the following basic or core elements:

- A *data model* and a *model for resources* needed for the business service. The data model should capture all data needed for processes including business data, status of processes or services, resource uses, correlations between process instances.
- A set of *processes* and *systems*. The processes are an implementation or realization of the business service and the systems may include database systems (managed inside and outside the EPF), and other systems the service will need, e.g., sensing devices, payment machines in Example 4. (Note that effects on data are more relevant than behaviors of processes.)
- A set of *relationships* between processes and systems especially with respect to data. For example, the cancellation process updates the order status and the invoice process must read the latest order status so that all actions by the user will be incorporated and reflected in the invoice.
- A set of *key performance indicators* (KPIs) and/or *quality of service* (QoS) metrics for the business service.

In addition, an EPF may also include other related elements: Specification of *administrative domains* may make cross-organizational information transfers explicit, *aggregation* mechanisms to allow more complex business services constructed hierarchically.

The main objective of EPFs is not that different from that of service composition or process modeling that has been studied variously in the services computing and

BPM fields. However, the earlier work focused mostly on runtime executions, and most of them paid little attention on modeling data and data accesses. Not surprisingly, the examples discussed earlier suggest that data provide key links (dependencies) between processes (instances).

An EPF provides a technical means to model business services. It not only allows a technical specification of services, but also brings possibility of algorithms and tools to help service design. For example, for the Internet service in Example 4, if the data relationships between cancellation, payment, invoice processes are present, it is possible to detect whether there are potential scenarios where a payment is never made but a refund is given; if process models are also available more detailed verification can be performed to ensure the intended semantics for the service is faithfully implemented by the processes.

Another advantage of EPFs is that KPIs can be formulated over a set of process at design time to reflect services more accurately. For the Internet service example, a salesperson may recruit many clients for subscriptions. A simple count of service orders only indicates the efforts by the salesperson, since many clients may cancel their subscriptions before installation. Modeled as an EPF, a KPI could be formulated over ordering, payment, and refund processes to better reflect the actual contributions by a saleperson.

In many cases government policies and regulations often applies to a group of processes. The tax refund in Examples 1 and 2 in Section III should only be used for goods purchased in the country and at most once for each eligible item. When an item that received prior tax refund is returned, or exchanged for another item, the completed refund should be taken into consideration. (Regulations such as these are complicated and hard to enforce since it involves multiple organizations.)

In services provided by a government, there are often opportunities for fraudulent uses and loopholes. The current strategy dealing with such issues is rather ad hoc, and passive, i.e., once misuse cases are found, measures are then taken to "patch" the hopes. Having the ability to group relevant process into an EPF, it is possible, at least theoretically, to analyze the EPF concerning potential abuses with formal verification techniques.

In general service compositions may be designed with intended business goals, or assembled arbitrarily often by a third party possibly with an intension for abuse. In either cases, EPFs provides a basis for developing algorithms, techniques, and tools to ensure desired design or to detect abuse anomalies.

Collaborative business processes have been a research focus in BPM study [22]. EPFs and collaborative processes are different but complementary. Research on collaborative processes focuses on runtime coordination and messaging aspects, usually with known participants. Existing specification languages such BPEL (for orchestration) and WS-CDL (for choreography) all require participants, at least their roles, to be defined in the collaboration, and behaviors of each participant (at least the messaging actions) are typically explicitly specified. An EPF, on the other hand, emphasizes on more general interactions especially via data in addition to runtime interactions. An EPF may focus less on behavior interactions and messaging.

Consider again the subscription service in Example 4. In an extreme, it is possible to design a single process model for the service. Such a crude approach suffers from several weaknesses. One is the added complexity of design, analysis (e.g., verification), validation, and changes since all activities are bundled into a single model. Also, it is also unclear when the process ends since the provider has no pior hints on whether a cancellation would come from a client. Finally, for scenarios in at least Examples 1-2, it may not even be possible to have a single process. EPFs easily avoid such problems.

Case handling is a new paradigm for supporting flexible and knowledge intensive business processes [13]. Case handling treats data as the typical product of processes and focuses on what can be done but not only what should be done from control flow perspective. In other words, case handling prefers assisting rather than guiding in doing so. However, compared to EPFs, case handling deals with each process instance or case separately and it assumes that one process instance should be independent with others. Generally, case handling does not take into consideration relationships between processes and nor beyond a single process.

EPFs naturally extend process models and collaborative process models. Technical specification languages for EPFs therefore might be obtained by augmenting choreography languages with needed features. In the following, we describe a couple of developments that can serve as the basis for technical specifications of EPFs.

There are two approaches to business process interoperation. In the *orchestrated* approach, a designated "mediator" communicates and coordinates with all participating business processes. This approach is widely used in practice (e.g. via BPEL) but it loses autonomy of participating processes and does not scale well. The *choreography* approach specifies global behaviors (e.g. in WS-CDL) among participating processes but otherwise leaves the processes to operate autonomously and communicate in the peer-to-peer fashion. Reference [33] presents a choreography language with two distinct features relevant to EPF specification: (1) Each participant has a tree-structured data model (representing the data to be accessed by the participant) with a selected sub-part of the data model visible to choreography specification.

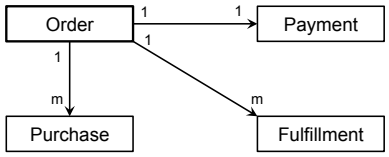

Fig. 1. A Correlation Graph with Four Processes

(2) Correlations between participants and instances are explicitly specified along with cardinality constraints on correlated instances in a *correlation graph*. Fig. 1 shows a correlation graph where rectangles represent processes (*Order*, *Payment*, *Purchase*, and *Fulfillment*) edges represent correlations between processes with the cardinalities on each edge indicating that at runtime a number of instances of one process will be correlated to a number of instances of the other. For example, an *Order* instance may be correlated to multiple *Purchase* instances, but only one *Payment* instance. The direction of an edge represents that at runtime, instances will be explicitly created by a correlated instance.

The language developed in [33] is for choreography specifications and thus only focuses on runtime interactions. It could nevertheless serve as a basis for further development for EPF specifications. Data modeling is essential for EPF specifications. There may be multiple conceptual data models to deal with in an EPF: a data model for each process, the global data model for the EPF, and possibly one or more data models in the persistent database system in the enterprise. These data models are closely related but yet different. It is necessary to define the logical relationships between them clearly.

[30] developed an approach to modeling data for business processes: representing data used by a process as a hierarchically structured business entity with keys, local keys, and update constraints, and a set of data mapping rules defining exact correspondence between entity data values and values in the enterprise database. Their data mapping language is based on path expressions, and schema mapping languages [16]. The study in [30] also includes "updatability" (translating data updates by a process to updates on the database, and vice versa) and "isolation" (updates by one process execution not altering data used by another running process). Properties such as these are quite useful for the study on EPFs.

V. Research Challenges

From the discussions in the previous sections, there is a clear need to model or technically specify a process framework so that both runtime and offline inter-process interactions can be captured and analysis to help address a range of application problems. In this section we discuss several research challenges concerning EPFs.

Modeling EPFs

An immediate research problem is to develop languages to specify EPFs. Given a wealth of existing modeling techniques for data, data mappings, processes, and (entity) relationships, a key challenge is to select modeling constructs from these different aspects and tailor them to EPFs and application problems. One interesting technical issue concerns precise formulation of KPIs for EPFs, which depends on the modeling choices for data, processes, and relationships.

Choreography languages only focus on runtime interactions between processes, but provide a basis for modeling more general relationships between processes. Causality such as an instance of one process spawns one or more instances of another is interesting [33], and more general occurrence-based relationships such as the constraints permitted by DecSerFlow [38] are relevant for runtime relationships. Concerning data dependencies, it is obvious to consider "read-from" relationships which would specify that one process must read the data written by another. One could also consider relationships that are conditional depending on data contents. When multiple instances are involved, it seems necessary to develop aggregate constructs so that one process instance may read from *all* instances of another process. We are currently developing a language to specify relationships between processes [28].

In applications, there is often a need to split a process, e.g., for out-sourcing [15]. Such split operations or more general projections are more frequently seen for business services. It would be desirable to develop algebras to perform projections, and perhaps joins for EPFs.

Anomaly detection and incident mining

From Examples 1 and 2 in Section III, it is apparent that anomalies are one of the focus of study. There are two technical aspects: detecting anomalies from EPF specifications, and mining undesirable incidents. Concerning detection a possible approach is to decompose the detection problem into individual processes and develop efficient techniques for "gluing" together processes and partial detection results. Concerning incident mining, based on well-known process mining techniques there could be at least two scenarios for incident analysis, i.e., incident mining (i.e., given logs of executed business processes, what EPF could be discovered?) and conformance checking (i.e., given logs of executed business processes and an EPF, what derivation could be found and what is the degree of the conformance between them?).

For the first scenario, compared to traditional process discovery, the following new challenges involving EPF should be considered carefully.

- What process logs should be taken into consideration? Are the given process logs complete enough

to discover process clusters for incident mining?
- How are the process instances from different process logs correlated together for mining with the help of business data being accessed? Furthermore, which kinds of pre-processing should be issued?
- Provided with the accessed data, which kinds of mining algorithms could be applied, adapted, or designed for discovering process clusters as well as the relationships between them?
- From the mining result, how could we evaluate the effectiveness and efficiency of the mining algorithm for EPF towards incident mining?
- How could the mining result be used to provide insights into the current running status of process clusters or even help analysts make decisions?

Although there are many process mining algorithms and even a few cross-organizational process mining algorithms, existing algorithms mainly focus on control flow perspective, seldom consider accessed data, and never take process clusters into consideration.

For the second scenario, all kinds of quantitative metrics for the conformance between process logs and EPFs need be developed as well as the evaluation framework for their effectiveness and efficiency.

For both scenarios, how could cloud computing, big data, artificial intelligence (especially machine learning and deep learning) techniques be effectively utilized remains an open problem.

Optimization and (semi)-automation

From the discussions in the previous section, an EPF is an improved representation and in fact, fairly completely most aspects of a business service. This brings the opportunity for the study of optimization issues concerning resources for business services or EPFs. Consider the subscription service in Example 4, From a single instance (i.e., one client orders a subscription), it may be cheaper (e.g., less human performer's time) to combine the payment process into the order process, as opposed to having a separate process. Similarly, a fragment of the packaging process (tools and parts for a new installation) could be combined with the shipping process (for a new router).

Database query evaluation is based on a "re-formulation" framework based on the evaluation cost. For example, the parser translates an SQL query from the user into a tree-representation of a relational algebraic expression, the optimizer then searches through (tree of) equivalent algebraic expressions, and selects one with the least cost to execute. One can imagine that an optimizer for EPFs could employ the same approach:

$$\text{EPF} + \text{recourse cost model} \Rightarrow \text{optimized EPF}$$

Finally, resource optimization must take into consideration of serving multiple service requests (i.e., multiple instances). This is because often instances are competing for resources. Simple strategies such as "FIFO queues" may be appropriate for some resources, while modification of an existing EPF may achieve more desirable KPIs while keeping resource consumption low. In BPM research, optimization with such competing instances has not been extensively looked at. For a starting point of exposition, the scheduling algorithms presented in [27] [41] tackle batched executions, the process model used in [18] may provide hints for non-batched cases.

Process change impact analysis and propagation

As illustrated in Example 3, the changes made in the degree process may affect other processes such as the unit program. Various work have been done in business process change management, most of them focused on managing changes in a single process or service [40][26], between processes [17][43][24] or between services [35][4], and analyzing change impacts between services and processes [39][44]. Verification techniques are also provided in these works in terms of consistency and compatibility. However all existing works deals with technical services/processes, which means there are explicit message/activity dependencies defined among them. In an EPF, we are dealing with a cluster of processes that are independently administered to support common a business service. In the setting of EPFs, within a business service, several processes are related not in terms of messages or having overlapping activities, and most of time they have nothing to do with each other like purchase and tax return two processes in Examples 1 and 2. Instead they become relevant to each other when changes happen in some processes. Some of the challenges are: how we shall specify these process relevance, and how to detect and evaluate the impact.

REFERENCES

[1] U. Apte, U. Karmarkar, and H. Nath. The U.S. information economy: Value, employment, industry structure, and trade. *Found. and Trends in Tech., Info. & Operations Manag.*, 6(1), 2012.

[2] Artifact-centric service interoperation (ACSI) web site, 2011. `http:/acsi-project.eu/`.

[3] B. Bagheri Hariri, D. Calvanese, M. Montali, G. De Giacomo, and A. Deutsch. Verification of relational data-centric dynamic systems with external services. In *Proc. ACM Symp. on Principles of Database Systems (PODS)*, 2013.

[4] K. Becker, A. Lopes, D. Milojicic, J. Pruyne, and S. Singhal. Automatically determining compatibility of evolving services. In *Proc. IEEE Int. Conf. on Web Services (ICWS)*, pages 161–168, 2008.

[5] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su. Towards formal analysis of artifact-centric business process models. In *Proc. Int. Conf. on Business Process Management (BPM)*, Brisbane, Australia, September 2007.

[6] K. Bhattacharya, R. Hull, and J. Su. A data-centric design methodology for business processes. In *Handbook of Research on Business Process Modeling*. Information Science Publishing, 2008.

[7] D. Calvanese, G. D. Giacomo, and M. Montali. Foundations of data-aware process analysis: A database theory perspective. In *Proc. ACM Symp. on Principles of Database Systems (PODS)*, pages 1–12, 2013.

[8] E. Damaggio, A. Deutsch, R. Hull, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. Int. Conf. on Business Process Management (BPM)*, pages 3–16, 2011.

[9] E. Damaggio, A. Deutsch, and V. Vianu. Artifact systems with data dependencies and arithmetic. In *Proc. Int. Conf. on Database Theory (ICDT)*, pages 66–77, 2011.

[10] E. Damaggio, R. Hull, and R. Vaculin. On the equivalence of incremental and fixpoint semantics for business artifacts with Guard-Stage-Milestone lifecycles. *Information Systems*, 38:561–584, 2013.

[11] NSF Workshop: Research Challenges on Data-Centric Workflows, May 2009. http://dcw2009.cs.ucsb.edu.

[12] E. de Murillas, W. van der Aalst, and H. Reijers. Process mining on databases: Unearthing historical data from redo logs. In *Proc. 13th Int. Conf. Business Process Management (BPM)*, 2015.

[13] W. V. der Aalst, M. Weske, and D. Grünbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.

[14] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. Int. Conf. on Database Theory (ICDT)*, pages 252–267, 2009.

[15] R. Eshuis, R. Hull, Y. Sun, and R. Vaculín. Splitting GSM schemas: A framework for outsourcing of declarative artifact systems. In *Proc. Int. Conf. on Business Process Management (BPM)*, pages 259–274, 2013.

[16] R. Fagin, L. M. Haas, M. Hernandez, R. J. Miller, L. Popa, and Y. Velegrakis. Clio: Schema mapping creation and data exchange. In *Conceptual Modeling: Foundations and Applications*, pages 198–236, 2009.

[17] W. Fdhila, C. Indiono, S. Rinderle-Ma, and M. Reichert. Dealing with change in process choreographies: Design and implementation of propagation algorithms. *Information Systems*, 49:1–24, 2015.

[18] C. E. Gerede and J. Su. Specification and verification of artifact behaviors in business process models. In *Proc. 5th Int. Conf. on Service-Oriented Computing (ICSOC)*, Vienna, Austria, September 2007.

[19] G. Group. Gartner Newsroom. http://www.gartner.com/it/page.jsp?id=1740414, 2011.

[20] R. Hull, E. Damaggio, F. Fournier, M. Gupta, F. Heath III, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, and R. Vaculín. Introducing the guard-stage-milestone approach to specifying business entity lifecycles. In *Proc. Workshop on Web Services and Formal Methods (WS-FM)*. Springer, 2010.

[21] R. Hull, E. Damaggio, R. D. Masellis, F. Fournier, M. Gupta, F. Heath III, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, and R. Vaculín. Business artifacts with guard-stage-milestone lifecycles: Managing artifact interactions with conditions and events. In *Proc. ACM Int. Conf. on Distributed Event-Based Systems (DEBS)*, 2011.

[22] C. Liu, Q. Li, and X. Zhao. Challenges and opportunities in collaborative business process management: Overview of recent advances and introduction to the special issue. *Information Systems Frontiers*, 11(3):201–209, 2009.

[23] M. Marin, R. Hull, and R. Vaculín. Data centric BPM and the emerging case management standard: A short survey. In *BPM 2012: Business Process Management Workshops*, pages 24–30, 2012.

[24] A. Martens. Consistency between executable and abstract processes. In *Proc. IEEE Int. Conf. on e-Technology, eCommerce and e-Service (EEE)*, pages 60–67, 2005.

[25] A. Nigam and N. S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.

[26] M. Papazoglou. The challenges of service evolution. In *Proc. Int. Conf. on Advanced Information Systems Engineering (CAiSE)*, pages 1–15. Springer, 2008.

[27] L. Pufahl and M. Weske. Batch activities in process modeling and execution. In *Proc. Int. Conf. on Service-Oriented Computing (ICSOC)*, pages 283–297, 2013.

[28] J. Su, L. Wen, and J. Yang. Processes and relationships—an approach to modeling business services. In preparation, 2017.

[29] J. Su and J. Yang. Yank your data out of my engine: A new approach to workflow system design. In *Proc. 8th Int. Workshop on Evolutionary Business Processes (EVL-BP)*, 2015.

[30] Y. Sun, J. Su, B. Wu, and J. Yang. Modeling data for business processes. In *Proc. International Conf. on Data Engineering (ICDE)*, 2014.

[31] Y. Sun, J. Su, and J. Yang. Separating execution and data management: A key to Business-Process-as-a-Service (BPaaS). In *Proc. Int. Conf. on Business Process Management (BPM)*, 2014.

[32] Y. Sun, J. Su, and J. Yang. Universal artifacts: A new approach to business process management (BPM) systems. *ACM Trans. Management Information Systems*, 7(1), 2016.

[33] Y. Sun, W. Xu, and J. Su. Declarative choreographies for artifacts. In *Proc. Int. Conf. on Service Oriented Computing (ICSOC)*. Springer, 2012.

[34] Y. Sun, W. Xu, J. Su, and J. Yang. SeGA: A mediator for artifact-centric business processes. In *Proc. Int. Conf. on Cooperative Information System (CoopIS)*, 2012.

[35] W. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, and K. Wolf. From public views to private views: correctness-by-design for services. In *Proc. Workshop on Web Services and Formal Methods (WS-FM)*, pages 139–153, 2007.

[36] W. van der Aalst and A. ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.

[37] W. van der Aalst and K. van Hee. *Workflow Management: Models, Methods and Systems*. The MIT Press, 2004.

[38] W. M. P. van der Aalst and M. Pesic. DecSerFlow: Towards a truly declarative service flow language. In *Proceedings of Workshop on Web Services and Formal Methods*, 2006.

[39] Y. Wang, J. Yang, W. Zhao, and J. Su. Change impact analysis in service-based business processes. *Service Oriented Computing and Applications*, 6(2):131–149, 2012.

[40] B. Weber, M. Reichert, and S. Rinderle-Ma. Change patterns and change support features – Enhancing flexibility in process-aware information systems. *Data & Knowledge Engineering*, 66(3):438–466, 2008.

[41] Y. Wen, Z. Chen, and T. Chen. An improved scheduling algorithm for dynamic batch processing in workflows. In *Proc. IEEE Int. Conf. on Cloud and Green Computing*, pages 502–507, 2013.

[42] M. Weske. *Business Process Management: Concepts, Languages, Architectures (2nd Ed.)*. Springer, 2012.

[43] A. Wombacher, P. Fankhauser, B. Mahleko, and E. Neuhold. Matchmaking for business processes based on choreographies. In *Proc. IEEE Int. Conf. on e-Technology, eCommerce and e-Service (EEE)*, pages 359–368, 2004.

[44] P. Xiu, J. Yang, and W. Zhao. Change management framework for service based business process. In *Proc. Australasian Computer Science Week Multiconference (ACSW)*, 2017.

[45] W. Xu, J. Su, Z. Yan, J. Yang, and L. Zhang. An artifact-centric approach to dynamic modification of workflow execution. In *Proc. Int. Conf. on Cooperative Information Systems (CoopIS)*, pages 256–273. Springer, 2011.