Bulletin of the Technical Committee on

Data Engineering

September 2009 Vol. 32 No. 3 **IEEE Computer Society**

Letters

Special Issue on Management of Data-Centric Business Workflows

Conference and Journal Notices

Letter from the Editor-in-Chief

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

Data management is a very nice area: research is well motivated by real world problems and techniques developed are used to make software systems better. In fact, the research area really grew out of application needs of data management in financial systems, inventory management, etc. in the 60's and early 70's.

The information technology innovations (computer hardware, software, networking, data management systems, the Internet, World Wide Web, etc.) in the last two decades is causing a fundamental change in almost all things we do, all spectrum of life, and all corners of modern societies. As computer scientists, we are now used to phrases such as "*digital* YYY" (YYY being *library*, *government*, *classroom*, *health care*—a new phase I learned in China this summer and translated here, ...) and "*e*-ZZZ" (ZZZ being *mail*, *tailer*, *science*, *book*, ...). However, *all things is* not *well here*. There are many *new* challenges arising from the real world software systems and applications, as suggested by some keywords selected from this year's SIGMOD/VLDB/ICDE/EDBT session titles: "security" and "privacy", "entity resolution", "information extraction", "data" on "modern hardware", "provenance", "uncertainty", "social networking", "mobility", "data quality", "meta data", etc. In this special issue, I would like to take a few minutes of your time to go on a tour of a *not-so-new* area but a *hidden treasure*: "business process management" or "BPM".

Roughly speaking, a *business process* is a collection of activities and services assembled together to accomplish a business goal (admitting a student, visiting doctor for an illness, reimbursing a business trip, granting a construction permit, establishing a law, ...). BPM refers to the management and support for a collection of inter-related business processes, often within an organization (government agency, real estate agency, hospital, institute, university, ...). This includes the management of all necessary resources (e.g., human) to ensure successful execution of all business processes, handling of exceptional cases, making needed changes for a range of reasons such as market competition, compliance to new laws and regulations, incorporation of new technology, and better management of resources. Clearly this is a *very old* problem!

What I am really speaking of is a new twist: when BPM meets IT! The availability of electronic storage, computer network, and advanced software development platforms is turning paper into digital documents and business processes into workflows (i.e., business processes aided by software systems). The BPM market (related to computer software) has already exceeded the billion-dollar mark. As a consequence, many management functionality now relies on software support. This is where things don't work very well. BPM practitioners today are facing enormous difficulties in many aspects due to a *profound* lack of technology related to IT.

The workflow concept is not new, definitely not to the database community. Traditional business process/workflow models focus mostly on the "control flow" aspect. In applications, the documents or data going through the workflow often play a vital role in determining whether the workflow would run correctly, effectively, and even efficiently. It's only natural that the area of workflow/BPM is embracing a significant shift from control flow-centric to *data-centric* workflow design and specification. Data-centricity is interesting and new. It has two facets. First, conceptual models for *data-aware workflow* elevate the data being manipulated by the workflows to the same level of prominence as given to control flow in traditional models. Second, the topic of *workflow as data* is emerging in recent studies on scientific workflow and business applications. A key issue is to easily represent, store, and query both workflow schemas and executions (or enactments).

I am happy to present this special issue on the interplay between data and processes in the context of BPM. These papers are authored by experts in the area and focus on three topics: *data-centric workflow models* ([Cohn and Hull], [Abiteboul, Segoufin, and Vianu], and [van der Aalst, Mans, and Russell]), *querying workflow models* ([Dumas, García-Bañuelos, and Dijkmanet] and [Deutch and Milo]), and *data and processes* ([Dayal, Wilkinson, Simitsis, and Castellanos], [zur Muehlen], and [Truong and Dustdar]). They can be a good starting point for your exploration but do not represent a comprehensive survey of the field. I hope you will enjoy this issue.

Jianwen Su
University of California, Santa Barbara

Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes

David Cohn and Richard Hull
IBM T.J. Watson Research Laboratory
Hawthorne, New York, USA
{dcohn,hull}@us.ibm.com

Abstract

Traditional approaches to business process modeling and workflow are based on activity flows (with data often an afterthought) or documents (with processing often an afterthought). In contrast, an emerging approach uses (business) artifacts, *that combine data and process in an holistic manner as the basic building block. These correspond to key business entities which evolve as they pass through the business's operation. This short paper motivates the approach, surveys research and its applications, and discusses how principles and techniques from database management research can further develop the artifact-centric paradigm.*

1 Introduction

The importance of effective Business Process Management (BPM) increases as the needs for better insight, understanding and efficiency for business operations increases. Classically, most BPM frameworks (e.g., [LRS02, vdAtHKB03]) have used meta-models[1] centered on activity-flows, with the data manipulated by these processes seen as second-class citizens. Another approach [GM05] focuses on the documents that track the business operations, with the process meta-model typically impoverished. For both, associated requirements, business rules, and business intelligence are based on conceptual meta-models only loosely connected to the base model. This disparity adds substantial conceptual complexity to models of business operations and processes, making them hard to understand. This paper focuses on (*business*) *artifacts*, rather than activity-flows or documents. Artifacts combine both data aspects and process aspects into a holistic unit, and serve as the basic building blocks from which models of business operations and processes are constructed. The approach enables a natural modularity and componentization of business operations and varying levels of abstraction. The paper motivates the approach, surveys research and applications, and highlights ways that philosophic underpinnings and selected techniques from database management research can further its development.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] Following the tradition of UML and related frameworks, we use the terms 'meta-model' and 'model' for concepts that the database and workflow research literature refer to as 'model' and 'schema', respectively.

Artifacts are business-relevant objects that are created, evolved, and (typically) archived as they pass through a business. The artifact type includes both an *information model* for data about the business objects during their lifetime, and a *lifecycle model*, describing the possible ways and timings that tasks can be invoked on these objects. A prototypical artifact is *Air Courier Package*, whose information model would include slots for package ID, sender, recipient, arrival times, delivery time, and billing information. The lifecycle model would include the multiple ways that the package could be delivered and paid for. Artifacts define a useful way to understand and track business operations, such as the locations that the package has passed through and its arrival times, as typically provided to customers.

Since 2003, IBM Research has been developing meta-models, methods, tools, user-centric paradigms, and other technologies in support of the artifact-centric paradigm [NC03, KNI+03, SNK+08, CDI+08, SSRM07, Hul08]. The methods and tools have been successfully applied in various settings [B+05, BCK+07, C+09].

Three key lessons have been learned from the work to date:

1. The artifact-centric approach enables rich, natural communication among diverse stakeholders about the operations and processes of a business, in ways that activity-flow based and document-based approaches have not. This has measurably reduced the time and staff needed to do business transformations, and enabled unexpected new capabilities.

2. The artifact-centric models, even though expressed in a way that business-level people can understand, are *actionable*, i.e., they can be mapped to execution-level models implementable with tools like IBM's WebSphere Process Server [Fer01], and can serve as an organizing foundation for related BPM capabilities, such as business rules, the development of web screens for task performers, and business intelligence.

3. There is a compelling opportunity for research into numerous aspects of the artifact-centric approach. From a core Computer Science perspective, artifacts provide a well-motivated framework that combines data and process in a manageable way; this combination has been largely missing from research on databases and knowledge representation, that has focused largely on data aspects, and also from research on programming langauges, software engineering, workflow, and verification, that has focused largely on process aspects. Specific areas for exploration include conceptual modeling (and, in particular, declarative meta-models), design methods, user-centric aspects, systems issues, integrity constraints, views and foundations. Core philosophic perspectives and techniques from database (and management information science) research can make substantial contributions to this field.

The following sections discuss each of these points in more detail.

2 Enabling understanding and communication

This section outlines the artifact-centric approach to modeling business operations, contrasts it with Entity-Relationship modeling in databases, and highlights how it facilitates stakeholder communication.

As detailed in [C+09], IBM Research has applied the artifact-centric paradigm to a problem faced by IBM Global Financing (IGF), which operates in more than 50 countries and annually finances over $40 billion in IT. After 25 years of organic growth, IGF's global operations were essentially in country "silos", each with different procedures. IGF needed operations based on a global standard with disciplined regional variations, that streamlined operations and allowed the business to expand its focus from large-scale loans to include moderately-sized deals. IGF had tried to do so using traditional techniques (e.g., process decomposition, Lean and Six Sigma), but was not succeeding.

IBM Research, working closely with IGF subject matter experts, applied the artifact method to create a high-level model of the IGF operations. This model is focused on three business artifacts:

- **Deal:** The activity around evaluating a client request, negotiating terms and conditions, signing the contract, issuing invoices for the assets to be financed, and tracking payments and completion.

- **Supplier Invoice:** The purchase and shipping of the asset(s) to the client location(s).
- **Asset:** The individual hardware asset(s) when accepted from the supplier, titled to IGF, delivered to the client, used by the client, and finally sold or disposed.

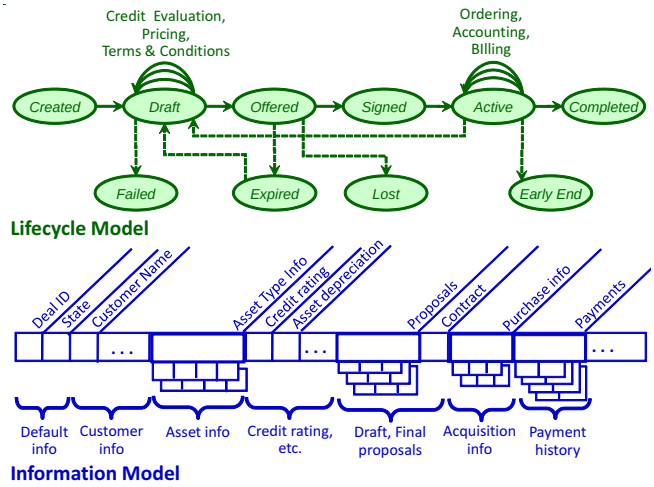


Figure 1: High-level specification of Deal artifact

Figure 1 shows an informal, high-level representation of the information and lifecycle models of the Deal artifact. The information model has slots for information gathered as a Deal artifact instance evolves, including customer details (credit ratings, etc.), types(s) of asset(s), terms and conditions, specific hardware asset(s) acquired, and payment history. The lifecycle model shows the key business-relevant states through which a Deal passes, with transition edges corresponding to tasks performed by specialists. The solid transition edges correspond to the "sunny day" state sequence from Created to multiple Draft versions, through Offered, Signed, multiple loops through Active, and, finally, Completed. The dashed edges show additional potential transitions, some going to additional states. The artifact information model starts out largely empty, and over the life of the artifact, its attributes are filled in (or overwritten). The first task in a Deal's Active state creates corresponding Supplier Invoice artifact instance(s); and when each physical asset is accepted by IGF, an Asset artifact instance is created. More generally, in the state-based approach to artifacts, instances interact through message passing as they transition between states. The artifact-based business operations model is being used by IGF to manage operations at both global and local levels. (The full Deal artifact type has about 100 attributes and 70 states.) IGF plans to automate their top-level operations around this model and expects significant efficiency gains.

There are parallels between the artifact approach to business operations modeling and the Entity Relationship (ER) approach [Che76] to modeling the data managed in a business. Both are systematic approaches that use a small set of natural and intuitive constructs. Further (as discussed in Section 3), business artifact specifications are *actionable*, in the same way that ER diagrams are actionable, i.e. the specification can be used to automatically generate an executable system. There is a contrast between how information is typically clustered in artifacts vs. in database schema design and document management systems. With database schemas, there is a tendency to break data into fairly small "chunks": ER-based techniques use separate entity types and their relationships; normal forms from relational database theory break data apart to avoid update anomolies. This is valuable when data is used by a variety of applications. Similarily, document management systems often focus on the company's literal document types rather than on the single conceptual entity which multiple document types together represent. In contrast, an artifact information model clusters the various kinds of data which correspond to the stages in the business entity's lifecycle.

Clustering data based on a dynamic entity that moves through a business's operations, rather than pieces of its lifecycle, makes a profound difference. As demonstrated in the IGF and other examples, it enables strong communication between a business's stakeholders in ways that traditional approaches do not. Experience has shown that once the key artifacts are identified, even at a preliminary level, they become the basis of a stakeholder vocabulary. Artifacts enable communication along three dimensions, which we illustrate using the Deal artifact. Along the lifecycle dimension, stakeholders who focus on one part of a lifecyle, say the Draft state, are better equipped to communicate with stakeholders focused on another part, say the Active state. All are talking about the same overall artifact and can confidently discuss attributes that are shared or produced in one part of

the lifecycle and consumed in another. Across the variations dimension, IGF stakeholders from multiple geographies could understand similarities and differences between their respective operations by comparing them to the commonly held artifact model. Communication between stakeholders at different management levels is enhanced because the artifact approach naturally lends itself to a hierarchical perspective. For example, the Deal artifact shown in Figure 1 is easily understood by executives, and a drill down is useful to stakeholders managing the detailed operations.

3 An actionable framework

The artifact-centric framework is actionable along two dimensions. An artifact model expressed in business-level terms can be automatically mapped onto a workflow engine to create a deployed system. Such a model can also be the basis for attaching a variety of traditional BPM capabilities.

There are currently three working implementations of the (state-based) artifact meta-model, each with a different purpose. Two are elements of the tooling associated with the Business Entity Lifecycle Analysis (BELA) *capability pattern* [SNK+08], that is part of IBM's Service Oriented Modeling and Architecture (SOMA) method. BELA's FastPath tool lets artifact model designers automatically generate a running model during the design process. It provides a full system shell and preliminary versions of performer web screens. Designers and executives can step through different scenarios, seeing how the artifact model behaves. The second BELA tool can map an artifact model into a workflow that runs on IBM's WebSphere Process Server [Fer01]. This has been used to deploy business processes that operate at a massive scale, with 100s of simultaneous users. The third implementation is the experimental Siena prototype [CDI+08, F.F09], that uses a direct architecture. The artifact model is represented as an XML document and execution is performed essentially by a direct interpretation of the XML. This system has been used for rapid prototyping exercises involving small- and medium-size applications, and is available to universities for teaching and research.

As noted, a business artifact is a blend of data and process for a key business-relevant dynamic entity that captures its end-to-end journey. As a result, business artifacts are a natural basis for many BPM suite capabilities. For example, [Lin07] describes a tool for using business rules expressed in OMG's SBVR standard [Obj08] with artifacts. The work shows that the vocabulary provided by artifacts is natural for specifying business rules, and shows how the rules can be mapped into the system to guide task sequencing and prevent rule violations. In the area of web screens for performers, [SMS09] describes how the basic artifact structure is the basis for automated implementation of the screens for carrying out business process tasks. A key enabler here is that the artifact model can include *CRUD* (Create-Read-Update-Delete) permissions in terms of artifact attributes. The rights of performers in a given role can depend on the artifact's state. Business artifacts also provide a natural basis for Key Performance Indicator (KPI) specification, monitoring, and response, because they correspond to the business-relevant entities the KPIs measure. Citation [K+07] describes how an artifact-centric model for a supply chain application was used for sense-and-respond monitoring and dashboarding. To summarize, the artifact-centric approach lets many BPM suite capabilities be based on a single model at both conceptual and implementation levels, rather than on several diverse conceptual models.

4 Research challenges

The combination of data and process provided by the business artifact approach raises interesting research issues ranging from conceptual modeling and design, to systems issues, to foundations. The artifact abstraction provides a vehicle for understanding the interplay between data and process in ways not supported by previous Computer Science abstractions. For example, artifacts permit the study of how a broad class of data evolves over time, providing structure and opportunity for application of old techniques and development of new ones. This section highlights challenges that may be of particular interest to the database community. Another survey

of research opportunities is [Hul08].

A central research challenge is to understand the basic building blocks and alternatives for artifact-centric meta-models. In some ways, this is analogous to research into semantic data models in the 70's and 80's [HK87]. Central to this investigation is the diversity of people involved in designing and specifying business operations, ranging from executives to business architects, business analysts, and subject matter experts, and finally to business solution designers. Typically, solution designers are comfortable with detailed artifact models, but the others often prefer high-level requirements, business rules, and scenarios. The relationship between these two levels of specification is analogous to that between semantic data models (including the ER model) and the relational model in database management. Important goals here include formal mechanisms to specify requirements, rules and scenarios, and to map and trace their links to detailed artifact models.

To date, work on the artifact-centric method and artifact meta-models, and also related work [BDW07, RDtHI09], has used a variant of finite state machines to specify lifecycles. Recent theoretical work (e.g., [BGH$^+$07, DHPV00, BHS09]), is exploring declarative approaches to specifying the artifact lifecycles following an event-condition-action and/or condition-action style. The Project $\mathrm{ArtiFact}^{\mathrm{TM}}$ team at IBM Research is developing a first practical artifact-centric meta-model along these lines. The meta-model will incorporate parallelism of human-performed tasks and explicit hierarchy in the lifecycle specification. Declarative approaches promise to enable succinct specification of variations which may arise across differing geographies or customer categories. Also, they may enable the development of multiple perspectives or views on an artifact model or portions of it, which would be useful to executives and subject matter experts. Finally, a declarative approach has already shown itself to be promising as a basis for verification of artifact model properties [BGH$^+$07, DHPV00].

Other variations in the meta-model also merit study, including the underlying data meta-model (e.g., XML-based or ontology-based [BDW07]), task models (e.g., CRUD information only, BPEL specifications, or pre- and post-conditions as in semantic web services), and association of tasks to artifacts (e.g., design time as is the tradition, or dynamically at run time). An intriguing direction is to use Active XML [ABM08] as a basis for supporting artifacts, as in [ABGM09].

Similar to database management, the artifact-centric approach enables separation of logical vs. physical concerns. While an artifact's information model may cluster multiple kinds of data and permit users to query and manipulate instances as a unit, they may be physically stored across multiple databases. Further, different parts of an artifact lifecycle might be carried out by different, perhaps legacy, applications or systems. Finally, as discussed in [NC03, ABGM09], it may be beneficial to view artifact instances as traveling between organizations, either conceptually or physically. Against this background of modeling choices, several systems issues need to be addressed. Because of the possibilities of parallel processing and interactions between artifact instances, concurrency control must be provided. The interplay of materialized and virtual data raises traditional problems of fast access, query processing across diverse data sources, and maintaining consistency across redundant copies of data, but in a structured context. If considering large scale deployments, it is useful to study techniques that follow the intended semantics of a declarative artifact model, but enable optimizations according to resource availability. Initial work towards such a framework, reminiscent of the use of the relational algebra as an optimization level under SQL, is reported in [BHS09].

A fundamental and largely unexplored area for artifacts, which received considerable attention in relational databases, is the constellation of design principles and integrity constraints. What is the analog for artifacts of the relational notion of update anomalies and normal forms, and the dependencies used to study them? As noted, normal forms tend to disaggregate data, whereas artifacts encourage clustering of data around an organization's underlying dynamic entities. Citation [LBW07] develops an algorithm that analyzes the input-output properties of different tasks, in order to recommend how data should be clustered to form the key artifacts. It is natural to think in terms of integrity constraints that address the evolution of artifacts; work on dynamic constraints and evolution in the relational model (e.g., [AV89]) can provide a useful starting point. Naturally arising classes of temporal constraints for artifacts may come from part of SBVR [Obj08]

Another unexplored area for artifacts is *views*. This is important, for example, when an artifact-centric model

is used to represent the activity of an *interoperation hub* that facilitates the choreography of multiple business processes. [HNN09]. Conference management sites like EasyChair and ConfTool are such hubs although they are not (currently) artifact-centric. In these applications, stakeholders have access to varying views of the overall system which restrict data and behavioral capabilities. Citation [HNN09] develops a notion of view for state-based artifacts, including projection and selection on the information model, and a form of *condensation* of states for the lifecycle model; an analog for declarative lifecycles remains open. More generally, basic properties such as the interplay of views and integrity constraints, and translating queries and modification requests against views into the base model remain largely unexplored.

Research into foundations underlying the artifact model is at an early stage. Studies of static analysis for state-based artifacts include [GS07, KLW08], and those for declarative lifecycles are in [BGH+07, DHPV00]. Citation [FHS09] presents a first study of synthesizing declarative artifact models, and [CGHS09] presents a preliminary investigation into dominance and relative expressive power of such models. Extension of these directions and development of a theory of constraints and views in the context of dynamic behavior, are promising challenges that call for techniques from database theory, finite model theory, and temporal and other logics.

Acknowledgements

This paper is based on work that began at IBM Research in the late 1990's, and picked up momentum in 2003. Many IBMers have made substantial contributions to the effort, and prominent contributors include: Kumar Bhaskaran, Kumar Bhattacharya, Nathan Caswell, Henry Chang, Tian Chao, Fabiana Fournier, Amit Fisher, Fenno (Terry) Heath III, Santosh Kumaran, Mark Linehan, Rong (Emily) Liu, Prabir Nandi, Anil Nigam, Piyawadee Noi Sukaviriya, John Vergo, and Frederick y Wu. Researchers outside of IBM have also contributed directly or indirectly to the larger artifact-centric vision, including Serge Abiteboul, Elio Damaggio, Alin Deutsch, Christian Fritz, Fabio Patrizi, Jianwen Su, and Victor Vianu. Finally, the research by Richard Hull described here is partially supported by NSF grants IIS-0415195, CNS-0613998, and IIS-0812578.

References

[ABGM09] S. Abiteboul, P. Bourhis, A. Galland, and B. Marinoiu. The AXML Artifact Model. In *Proc. 16th Intl. Symp. on Temporal Representation and Reasoning (TIME)*, 2009.

[ABM08] S. Abiteboul, O. Benjelloun, and T. Milo. The Active XML project: An overview. *Very Large Databases Journal*, 17(5):1019–1040, 2008.

[AV89] S. Abiteboul and V. Vianu. A transaction-based approach to relational database specification. *Journal of the ACM*, 36(4):758–789, October 1989.

[B+05] K. Bhattacharya et al. A Model-driven Approach to Industrializing Discovery Processes in Pharmaceutical Research. *IBM Systems Journal*, 44(1), 2005.

[BCK+07] K. Bhattacharya, N. S. Caswell, S. Kumaran, A. Nigam, and F. Y. Wu. Artifact-centered operational modeling: Lessons from customer engagements. *IBM Systems Journal*, 46(4):703–721, 2007.

[BDW07] M. Born, F. Dörr, and I. Weber. User-friendly semantic annotation in business process modeling. In *Proc. Web Information Systems Engineering - WISE 2007 Workshops*, 2007.

[BGH+07] K. Bhattacharya, C. E. 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)*, pages 288–304, 2007.

[BHS09] K. Bhattacharya, R. Hull, and J. Su. A Data-centric Design Methodology for Business Processes. In J. Cardoso and W.M.P. van der Aalst, editors, *Handbook of Research on Business Process Management*. Information Science Publishing, an imprint of IGI Global, Hershey, PA, USA, 2009.

[C+09] T. Chao et al. Artifact-based transformation of IBM Global Financing: A case study, 2009. *Intl. Conf. on Business Process Management (BPM)*, September, 2009.

[CDI+08] D. Cohn, P. Dhoolia, F.F. (Terry) Heath III, F. Pinel, and J. Vergo. Siena: From powerpoint to web app in 5 minutes. In *Intl. Conf. on Services Oriented Computing (ICSOC)*, 2008.

[CGHS09] D. Calvanese, G. De Giacomo, R. Hull, and J. Su. Artifact-centric workflow dominance. In *Proc. Intl. Conf. on Service Oriented Computing (ICSOC)*, 2009. to appear.

[Che76] P. P. Chen. The entity-relationship model – toward a unified view of data. *ACM Transactions on Database Systems*, 1:9–36, 1976.

[DHPV00] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. Intl. Conf. on Database Theory (ICDT)*, 2009.

[Fer01] D. Ferguson. IBM Web Services: Technical and Product Architecture Roadmap. IBM Corporation, 2001. `http://www4.ibm.com/software/solutions/webservices/pdf/roadmap.pdf`.

[F.F09] F.F. (Terry) Heath III and F. Pinel. Siena user's guide, 2009. In preparation.

[FHS09] C. Fritz, R. Hull, and J. Su. Automatic construction of simple artifact-based workflows. In *Proc. of Intl. Conf. on Database Theory (ICDT)*, 2009.

[GM05] R.J. Glushko and T. McGrath. *Document Engineering: Analyzing and Designing Documents for Business Infomratics and Web Services*. MIT Press, Cmabridge, MA, 2005.

[GS07] C. E. Gerede and J. Su. Specification and verification of artifact behaviors in business process models. In *Proceedings of 5th International Conference on Service-Oriented Computing (ICSOC)*, Vienna, Austria, September 2007.

[HK87] R. Hull and R. King. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys*, 19:201–260, 1987.

[HNN09] R. Hull, N.C. Narendra, and A. Nigam. Facilitating workflow interoperation using artifact-centric hubs. In *Proc. Intl. Conf. on Service Oriented Computing (ICSOC)*, 2009.

[Hul08] R. Hull. Artifact-centric business process models: Brief survey of research results and challenges. In *On the Move to Meaningful Internet Systems: OTM Confederated Intl. Conf.s*, Monterrey, Mexico, 2008.

[K+07] S. Kapoor et al. Sense-and-respond supply chain using model-driven techniques. *IBM Systems Jornal*, 46(4):685–702, 2007.

[KLW08] S. Kumaran, R. Liu, and F.Y. Wu. On the duality of information-centric and activity-centric models of business processes. In *Proc. Intl. Conf. on Advanced Information Systems Engineering (CAISE)*, 2008.

[KNI+03] S. Kumaran, P. Nandi, F.F. (Terry) Heath III, K. Bhaskaran, and R. Das. ADoc-oriented programming. In *Symp. on Applications and the Internet (SAINT)*, pages 334–343, 2003.

[LBW07] R. Liu, K. Bhattacharya, and F. Y. Wu. Modeling business contexture and behavior using business artifacts. In *CAiSE*, volume 4495 of *LNCS*, 2007.

[Lin07] M. Linehan. Ontologies and rules in business models. In *Proc. 3rd Intl. Workshop on Vocabularies, Ontologies and Rules for the Enterprise (VORTE)*, 2007.

[LRS02] F. Leymann, D. Roller, and M.-T. Schmidt. Web services and business process management. *IBM Systems Journal*, 41(2):198–211, 2002.

[NC03] A. Nigam and N. S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.

[Obj08] Object Management Group (OMG). Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.0. , January 2008.

[RDtHI09] G. Redding, M. Dumas, A.H.M. ter Hofstede, and A. Iordachescu. Modelling flexible processes with business objects. In *Proc. 11th IEEE Intl. Conf. on Commerce and Enterprise Computing (CEC)*, 2009.

[SMS09] N. Sukaviriya, S. Mani, and V. Sinha. Reflection of a year long model-driven business and ui modeling development project. In *Proc. 12th IFIP Conference on Human-Computer Interaction (INTERACT)*, 2009.

[SNK+08] J.K. Strosnider, P. Nandi, S. Kumarn, S. Ghosh, and A. Arsanjani. Model-driven synthesis of SOA solutions. *IBM Systems Journal*, 47(3):415–432, 2008.

[SSRM07] N. Sukaviriya, V. Sinha, T. Ramachandra, and S. Mani. Model-driven approach for managing human interface design life cycle. In *Proc. Intl. Conf. on Model Driven Engineering Languages and Systems (MoDELS)*, pages 226–240, 2007.

[vdAtHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(3):5–51, July 2003.

Modeling and Verifying Active XML Artifacts *

Serge Abiteboul
INRIA Saclay & LSV - ENS Cachan, France

Luc Segoufin
INRIA & LSV - ENS Cachan, France

Victor Vianu
U.C. San Diego, USA

1 Introduction

Shared evolving data is central to an increasing range of human activities. In response to the need for computerized support of such activities, the notion of *business artifact* has been proposed at IBM as a model of such evolving data [1]. The model captures both the flow of control (workflow) of the application and the evolution of the relevant data (data cycle); see [2] for a brief survey. In the same spirit, we propose a new artifact model building upon Active XML (AXML for short), an extension of XML with embedded service calls [3]. The services are hosted by autonomous peers that evolve and interact by exchanging XML data. We claim that this can provide the foundation for an appealing artifact model, combining the advantages of semistructured data and of the Web service paradigm. With the model in place, we consider the verification of data-intensive applications, which is particularly critical for such systems due to their vulnerability to costly bugs. Despite the expressiveness of the model, we show that verification remains possible under reasonable restrictions.

Workflow and database systems are two essential software components that often have difficulties interoperating. Data-centric workflow systems are meant to integrate the control aspect of workflows with the underlying data. They allow managing data evolution by tasks with complex sequencing constraints as encountered for instance in scientific workflow systems, information manufacturing systems, e-government, e-business or healthcare systems. One can distinguish two main approaches for combining the database and workflow components. One consists in starting from a workflow approach, enriching it with data, e.g., by explicitly introducing state variables and specifying how they may evolve. The second emphasizes data placed at the center of the specification, but enriches it with means of controlling how it evolves. There is no fundamental separation between these two kinds of approaches but more a bias coming from where the emphasis is placed. However, when an emphasis is placed on the data (as we do here), one tends to prefer declarative specifications based on constraints on the evolution rather than control-based specifications.

We follow here a data-centric workflow approach where both data and tasks, but also the "actors" (humans, processes, systems) are captured by AXML *artifacts* [4]. The basis of this work is thus the Active XML model. AXML documents [3, 5] are XML documents with embedded function calls realized as Web service calls. Observe that the central notion is a document, so data, but that the model also involves computation, i.e., Web services. A main issue in the AXML technology is "when is a Web service call" evaluated. In query processing, a call may be activated because its result may impact the result of a query; this is in the spirit of recursive query processing. A call may also be activated bacause some event occurred, as in active databases. In the present

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Work supported by the Webdam Grant of the European Research Councild (FP7); n. 226513. The work of the first two authors is also supported in part by the EC FoX Grant. The work of the last is supported in part by the National Science Foundation under grant number IIS-0916515.

work, we want activation to be guided by the logic of the application, e.g. by some workflow constraints. In particular, we want to be able to specify some particular sequencings of the Web service calls inside a document.

An example of AXML documents is shown in Figure 3. Function *warehouseOrder* is used to obtain the parts from a warehouse. Function *deliveryOrder* is used to start the delivery process. It is likely that the system will delay the activation of this second function until after the computer has been built. On the other hand, the parts should be obtained from the warehouse before the construction starts.

The calls in an AXML document may be activated from inside (the artifact as client) and then receive answers in push or pull mode. Calls may also be activated from outside (the artifact as server). Rules are used to specify the logic of functions declaratively [6]. We use such documents to represent artifacts. In the spirit of [7, 1], an AXML document represents a process that evolves in time. A function call may be seen as a request to carry out a subtask whose result may lead to a change of state in the document.

An Active XML system specifies a set of interacting AXML documents. In such a system, there is an important distinction between internal and external services. An internal service is a service that is completely specified within the system whereas an external one captures interactions with other services or with users. One important goal is to statically analyze the behavior of such systems, which is especially challenging because the presence of data induces infinitely many states. We illustrate this aspect by mentioning some work on the verification of restricted centralized AXML systems [6]. These results can be easily transferred to distributed systems of AXML artifacts.

In this paper, we briefly present the AXML artifact model [4] (Section 2). We also mention some work on data-centric verification from [6] (Section 3). The last section provides brief conclusions.

2 The AXML Artifact model

Artifacts present several facets that, in our opinion, should be captured by an artifact model. An artifact is an *object* with a universal identity (e.g., URI). Its *state* is self-describing (e.g., XML data) so that it may be easily transmitted or archived. An artifact may host other artifacts as components, yielding a hierarchy of artifacts. At the physical level, each artifact at the root of the hierarchy is hosted by a peer. During its life cycle, an artifact is created, evolves in time, migrates among hosts, may hibernate and be reactivated, or dies according to a logic that is specified declaratively. Its *evolution* may be constrained to obey some laws, e.g. a *workflow*. An artifact *interacts* with the rest of the world via function calls (e.g., Web services) both as a server and a client. An artifact provides for communications, storage and processing for the artifacts it hosts. As in scientific workflows, an artifact has a *history* including time and provenance information that may be recorded and queried. These requirements have been in part motivated by [8].

To illustrate, consider a simplified view of the Dell manufacturing system [9] (Figure 1). When a new Web order arrives (1), a new *webOrder* artifact is created and creates a subartifact that is sent to a credit service (2). Once credit has been approved, the subartifact returns to the *webOrder* but now its state contains all the credit data. A plant is then selected and the artifact moves to that plant (3). It initiates a new subartifact for gathering parts, that is sent to a warehouse and another local artifact for communications with the customer (4). Once the product has been built, the artifact is sent to a delivery service (5). Finally, once the Web order has been completed, the artifact moves to an archive where it is stored as a text-based XML serialization that includes all the information it has gathered during its life cycle (6). (Subartifacts may also be archived separately.) The Dell example can be naturally modeled in AXML. See Figure 3 where the tree is represented using an XML syntax (a text-based serialization of the tree). The figure shows part of a *webOrder* artifact immediately after it enters the plant. The *creditApproval* element denotes a subartifact (the one that has been processed by the bank). The functions *?warehouseOrder* and $?comm$ will be activated next in order to create the *warehouseOrder* and *communication* subartifacts that will then work concurrently (and somewhat autonomously).

More broadly, the use of AXML as a basis for an artifact model is motivated by the fact that it can be easily

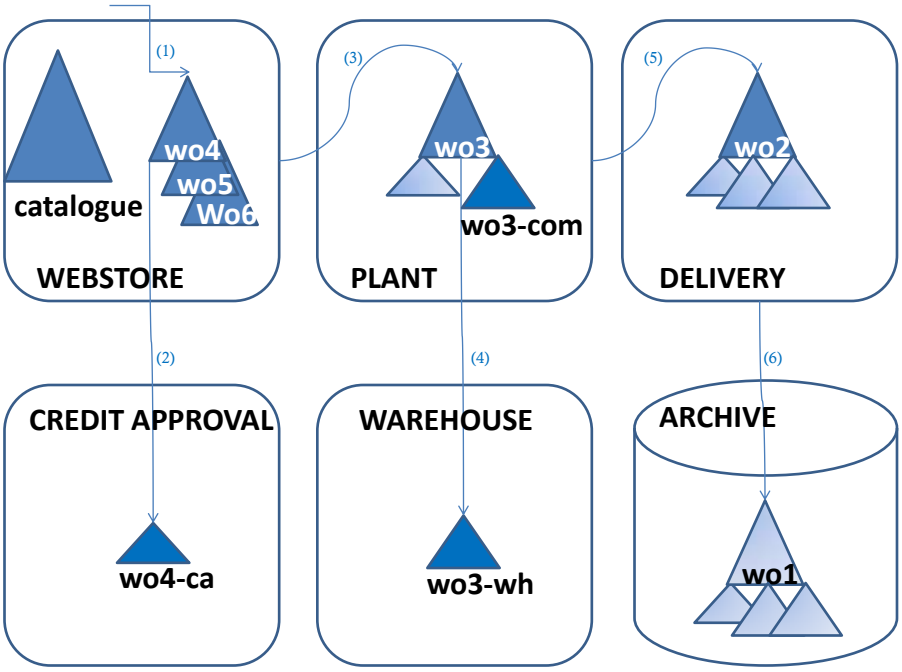


Figure 1: Artifacts in the Dell application

adapted to support all the requirements identified above. In particular, AXML naturally captures the distribution and autonomy of artifacts and provides reliable synchronous or asynchronous communication, allowing an artifact to send a message to another artifact just by knowing its ID. Also, because of its nested structure, AXML naturally supports hierarchies of artifacts. Two functionalities have to be added to AXML in order to fully support the above requirements. First, since we want artifacts to move from place to place in the system, we need an identification mechanism serving as a URI for artifacts. We also augment the rule-based workflow specification provided by AXML with workflows specified in a transition-based BPEL style that is more familiar to application designers.

The core of an application is a *schema* specifying a set of of peers and a set of classes (e.g., webOrder, financialService). The definition of a class provides typing of the data (document types), dynamic constraints on artifacts evolution (their workflows) and the interface of functions that the artifacts in this class export. From an implementation viewpoint, a peer provides storage, communications and computing resources for the artifacts it hosts. Artifacts are allowed to exchange data with other peers or to move to other peers. AXML data (e.g., in function arguments and results) is sent as strings and reconstructed at the receiving peers.

The semantics of functions is specified by rules. The declarative semantics facilitates reasoning about the runs of such systems and performing optimization. Function call activation is controlled by *call guards* that are specified by *Boolean combinations of tree-patterns* over the documents. Observe that the guards impose constraints on the evolution of documents in the style of condition-action rules. This may be seen as specifying *workflow* constraints on the runs of the system. Alternatively, one might prefer a more standard workflow approach in the style BPEL. The workflow is then specified by defining *stages* in the evolution of the artifact and admissible transitions between them. We are currently working on a comparison of the two styles of workflow specifications.

The notions of *task*, *service*, *state*, *stage*, and *activity*, that are essential in the artifact context, can all be

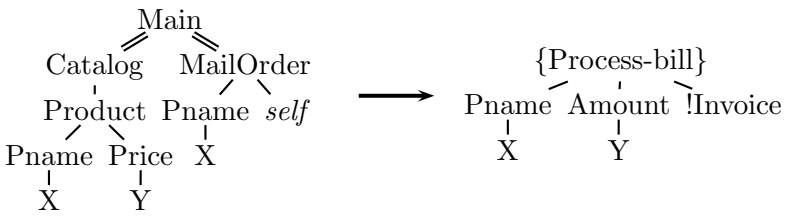


Figure 2: A GAXML query

```
< plant artID="plant02" >
...
< webOrder artID="wo3" >
    < client >
      < name > Sue Leroux < /name >
      < address > ... < /address >
    < /client >
    < order > ... < /order >
    < order > ... < /order >
  < creditApproval artID="wo3-ca" >
   ...
  < /creditApproval >
  < fun funID="?warehouseOrder"/ >
  < fun funID="?deliveryOrder"/ >
  < fun funID="?comm"/ >
< /webOrder >
...
< /plant >
```

Figure 3: An AXML artifact

formally captured in the AXML artifact model. The notion of *activity*, often arising in functional decompositions of business processes, may be seen as a view over a system of artifacts. The notions of *time* and *provenance* that are central to scientific workflows can be captured as well, because this information can be recorded and maintained in XML documents.

Related work Although the notion of artifact has been recently articulated by [1], similar ideas of data centric workflows have been around, e.g., in AXML [3], in the Vortex system [10] or scientific workflows [11]. The models that are considered are often restricted, e.g., [12, 8]. For instance, a single artifact is usually considered, vs. a system of artifacts in the present paper. Also, these models are often based on the relational model so have difficulties with collections of artifacts or nested tasks/artifacts. Formal models for data-centric workflows have been considered in [13] (that focuses on verification) and [14] (that discusses the synthesis of artifacts).

3 Verification

The need for reasoning about artifact systems arises in many contexts and is particularly challenging because of the presence of data. We briefly summarize results obtained in [6] on static analysis of AXML systems, in particular on automatic verification of temporal properties of their runs.

Classical automatic verification techniques operate on finite-state abstractions that ignore the critical semantics associated with data in such applications. The need to take into account data semantics has spurred interest in studying static analysis tasks in which data is explicitly present. We have started an investigation of the automatic verification of Active XML systems. We consider properties expressed in Tree-LTL, an extension of LTL where propositions are interpreted as tree patterns. For instance, one may want to verify whether some static property (e.g., all ordered products are available) and some dynamic property (e.g. an order is never delivered before payment is received) always hold. Tree-LTL allows to express a rich class of such properties. An example of Tree-LTL formula can be found in Figure 4.

We have identified a significant fragment of Active XML, called non-recursive Guarded AXML for which the verification of Tree-LTL properties is decidable. This fragment is expressive enough to describe meaningful

applications. We use also it as a convenient formal vehicle for studying decidability and complexity boundaries for verification in AXML in general. We briefly describe next Guarded AXML (GAXML for short) and its non-recursive fragment.

Every product for which a correct amount has been paid is eventually delivered (note that the variable Z is implicitly existentially quantified in the left pattern):

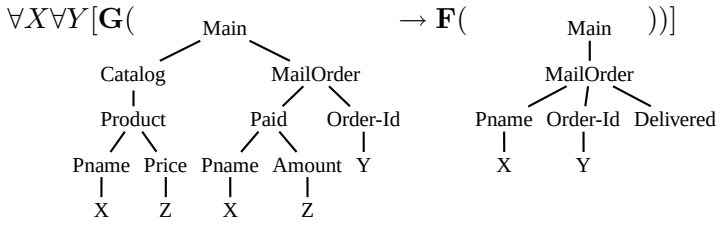


Figure 4: A Tree-LTL formula

In GAXML, document trees are unordered. With ordered trees, verification quickly becomes undecidable. Finally, the most novel feature of the model in the AXML context is a *guard* mechanism for controlling the initiation and completion of subtasks (formally function calls). Guards are Boolean combinations of tree patterns. They facilitate specifying applications driven by complex workflows and, more generally, they provide a very useful programming paradigm for active documents.

We obtain decidability by disallowing recursion in GAXML systems, which leads to a static bound on the total number of function calls in runs. We prove that for such non-recursive GAXML, satisfaction of Tree-LTL formulas is CO-2NEXPTIME-complete. We also consider various relaxations of the non-recursiveness restriction and show that they each lead to undecidability. This establishes a fairly tight boundary of decidability of satisfaction of Tree-LTL properties by GAXML systems.

Related work Most of the previous work on static analysis on XML (with data values) deals with documents that do not evolve in time (static constraints). This motivated studies of automata and logics on strings and trees over infinite alphabets, see [15] for a survey. Previous work on AXML also considered the evolution of documents. For instance, this is considered in [16] for a monotone AXML language, *positive* AXML. The setting is very different from ours, as their systems are monotone but possible recursive. In contrast, we consider verification for nonmonotone systems. Static analysis is also studied in [17] using a model based on tree rewriting. Verification of temporal properties of Web services has mostly been considered using models abstracting away data values (see [18] for a survey). Verification of data-aware Web services was studied in [19, 20], and a verifier implemented [21]. While this is related in spirit to the present work, the technical differences stemming from the AXML setting render the two investigations incomparable.

4 Conclusion

We briefly mention some remaining issues related to the work presented here. A most interesting direction of research is to enrich beyond non-recursive GAXML the class of AXML artifact systems that can be verified. For example, one would also like to be able to reason about time (this is complicated for several reasons, including the absence of a global clock). When full verification cannot be performed, abstraction may be useful. Recent work considers a related approach based on *interfaces* in the context of AXML [22]. Besides verification, a main issue for such systems is monitoring. A P2P monitoring system for AXML is studied in [23, 24]. Finally,

largely unexplored in this context are access control mechanisms, allowing different actors to keep control over their own data without imposing unacceptable constraints on the system.

References

[1] A. Nigam and N. Caswell, "Business artifacts: An approach to operational specification," *IBM Systems Journal*, vol. 42, no. 3, pp. 428–445, 2003.

[2] R. Hull, "Artifact-centric business process models: Brief survey of research results and challenges," in *OTM*, 2008.

[3] S. Abiteboul, O. Benjelloun, and T. Milo, "The Active XML project: an overview," *VLDB J.*, 2008.

[4] S. Abiteboul, P. Bourhis, A. Galland, and B. Marinoiu, "The axml artifact model," in *16th International Symposium on Temporal Representation and Reasoning (TIME-2009)*, 2009.

[5] Active XML, "http://activexml.net."

[6] S. Abiteboul, L. Segoufin, and V. Vianu, "Static analysis of Active XML systems," in *PODS*, 2008, pp. 221–230, full paper in *ACM TODS* 34:3, 2009.

[7] R. Khalaf, A. Keller, and F. Leymann, "Business processes for web services: Principles and applications," *IBM Systems Journal*, no. 45:2, 2006.

[8] K. Bhattacharya, R. Hull, and J. Su, "A data-centric design methodology for business processes," in *Handbook of Research on Business Process Management*, J. Cardoso and W. van der Aalst, Eds. N.A., 2009.

[9] R. Kapuscinski, R. Q. Zhang, P. Carbonneau, R. Moore, and B. Reeves., "Inventory decisions in Dell's supply chain," *Interfaces*, vol. 34, no. 3, pp. 191–205, 2004.

[10] G. Dong, R. Hull, B. Kumar, J. Su, and G. Zhou, "A framework for optimizing distributed workflow executions," in *DBPL 1999*, 2000.

[11] S. Davidson and J. Freire, "Provenance and scientific workflows: Challenges and opportunities," in *ACM SIGMOD Int. Conf. on Management of Data*, 2008.

[12] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su, "Towards formal analysis of artifactcentric business process models," in *Int. Conf. on Business Process Management*, 2007.

[13] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu, "Automatic verification of data-centric business processes," in *ICDT*, 2009.

[14] C. Fritz, R. Hull, and J. Su, "Automatic construction of simple artifact-based business processes," in *ICDT*, 2009.

[15] L. Segoufin, "Static analysis of xml processing with data values," *Sigmod Record*, no. 36:1, 2007.

[16] S. Abiteboul, O. Benjelloun, and T. Milo, "Positive Active XML," in *ACM PODS*, 2004, pp. 35–45.

[17] B. Genest, A. Muscholl, O. Serre, and M. Zeitoun, "Tree pattern rewriting systems," in *ATVA, LNCS 5311*, 2008.

[18] R. Hull, M. Benedikt, V. Christophides, and J. Su, "E-services: a look behind the curtain," in Proc. *ACM PODS*, 2003.

[19] A. Deutsch, L. Sui, and V. Vianu, "Specification and verification of data-driven web applications," *J. Comput. Syst. Sci.*, vol. 73(3), 2007.

[20] A. Deutsch, L. Sui, V. Vianu, and D. Zhou, "Verification of communicating data-driven web services," in Proc. *ACM PODS*, 2006.

[21] D.-D. W. A. A Verifier for Interactive, "A. deutsch and m. marcus and l. sui and v. vianu and d. zhou," in Proc. *ACM SIGMOD*, 2005.

[22] A. Benveniste and L. Helouet, "Interface for Active XML," private communication.

[23] S. Abiteboul and B. Marinoiu, “Distributed Monitoring of Peer to Peer Systems,” in *Workshop On Web Information And Data Management*, 2007, pp. 41–48.

[24] S. Abiteboul, P. Bourhis, and B. Marinoiu, “Satisfiability and Relevance for Queries over Active Documents,” in *PODS 2009*, 2009.

Workflow Support Using Proclets: Divide, Interact, and Conquer

W.M.P. van der Aalst and R.S. Mans and N.C. Russell
Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
w.m.p.v.d.aalst@tue.nl

Abstract

Classical workflow notations primarily support monolithic processes. They are able to describe the life-cycle of individual cases and allow for hierarchical decomposition. Unfortunately, real-life processes are fragmented and are often composed of separate but intertwined life-cycles running at different speeds and coping with different levels of granularity. The proclets framework was one of the first formalisms to acknowledge this. Proclets are lightweight interacting processes *that can be used to divide complex entangled processes into simple fragments and, in doing so, place increased emphasis on interaction-related aspects of workflows. This paper describes the proclets approach and presents an application of this approach to the gynecological oncology workflow process at a major Dutch hospital.*

1 Introduction

Although most information systems are "process aware" the support for various aspects of operational processes leaves much to be desired. For example, workflow technology is mostly used to automate repetitive well-structured processes. There is little support for less structured processes that require more flexibility. As a consequence of the widespread adoption of database technology in the seventies, the development of information systems is predominantly *data-centric*, i.e., the design and implementation starts with object/information modeling. However, since the nineties, consultants and vendors have been advocating more *process-centric* approaches. Today, the majority of larger organizations spend considerable time identifying and modeling processes. Business Process Management (BPM) techniques and tools support these more process-centric approaches and have received considerable attention. However, when looking at the actual implementations of information systems there is still a *mismatch between the processes modeled and reality* (i.e., the real systems and processes).

This mismatch has several reasons. One is that most actors have a simplistic and often incorrect view of the processes in which they are involved. Process mining techniques can be used to provide a more realistic view of their actuality [3]. It is often the case that processes are more complex and "spaghetti-like" than we expect. Reality cannot be captured in a structured monolithic workflow model. Another reason is that an effective balance/integration between/of the data perspective and the process perspective is missing. It is impossible to separate these perspectives. Moreover, it is obvious that the process-centric approaches used in the initial phases of workflow specification do not fit well with the predominant data-centric implementation approaches.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Currently, there is a renewed interest in the mismatch described above. This is illustrated by the recent NSF Workshop on Data-Centric Workflows that took place in Arlington (Virginia) in May 2009. (See [5] for the workshop report.) During this workshop there was consensus that processes cannot be straightjacketed in monolithic workflows and that the interplay between data and control-flow is essential.

In this paper, we advocate the use of *proclets*, one of the first modeling languages to address these problems [1, 2]. Proclets can be seen as lightweight interacting processes. The proclets framework be used to integrate data-centric and process-centric approaches at both the design and implementation level [1, 2]. To illustrate the framework and the ideas behind it, the gynecological oncology workflow at the AMC hospital in The Netherlands is modeled in terms of proclets.

In the remainder of this paper we first discuss the limitations of "monolithic workflows" (Section 2), followed by a brief introduction to the proclets framework (Section 3). In Section 4, we describe the application of proclets at a Dutch hospital. Section 5 concludes the paper.

2 Limitations of Monolithic Workflows

Proclets aim to address the following problems that existing workflow approaches are currently facing:

- Models need to be *artificially flattened* and are unable to account for the mix of *different perspectives and granularities* that coexist in real-life processes.
- Cases need to be *straightjacketed into a monolithic workflow* while it is more natural to see processes as intertwined loosely-coupled object life-cycles.
- It is impossible to capture the fact that *one-to-many* and *many-to-many* relationships exist between entities in a workflow, yet such relationships are common as can be seen in any data/object model.
- It is difficult to model interactions between processes, i.e., *interaction is not a first-class citizen* in most process notations.

In the remainder, we use proclets to address the problems that are experienced in monolithic workflows.

3 Proclets: Lightweight Interacting Processes

A *proclet* can be seen as a lightweight workflow process able to interact with other proclets that may reside at different levels of aggregation [1, 2]. One can think of proclets as objects equipped with an explicit life-cycle or as active documents. Recently, this has been referred to as artifact centric workflows/processes [4]. Proclets interact via *channels*. A channel is the medium used to transport messages from one proclet to another. Via a channel, a message can be sent to a specific proclet or a group of proclets (i.e., multicast). Such messages are called *performatives* since they correspond to explicit actions such as those found in speech act theory and the language/action perspective. Based on the properties of the channel, different kinds of interaction are supported, e.g., push/pull, synchronous/asynchronous, and verbal/non-verbal. Proclets are connected to channels via *ports*. Each port has two attributes: (a) its *cardinality* and (b) its *multiplicity*. The cardinality specifies the number of recipients of performatives exchanged via the port. The multiplicity specifies the number of performatives exchanged via the port during the lifetime of any instance of the class. The life-cycle of a particular type of proclet and its ports are specified in terms of a *proclet class*. Using these concepts, complex monolithic workflow definitions describing the control flow of an entire process can be broken up into smaller interacting proclets, i.e., there is a *shift in emphasis from control to communication*.

Proclets were introduced in the late nineties [1, 2]. In the original publications a variant of Petri nets, called *workflow nets*, was used as a basis. However, the main ideas are independent of the control-flow language

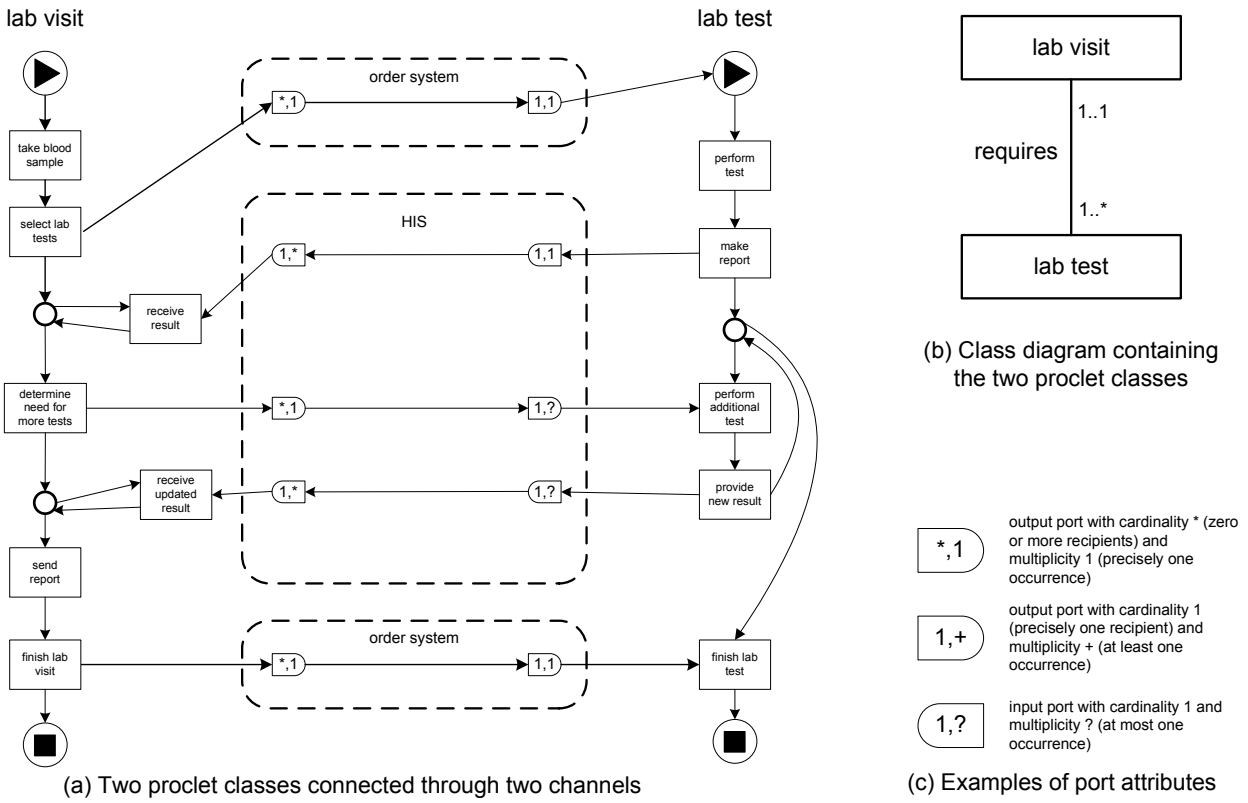


Figure 1: Example using two proclet classes: *lab visit* and *lab test*.

utilized. Therefore, we use the YAWL language rather than workflow nets, because YAWL is more expressive and supported by an extensive set of tools (editor, workflow engine, process mining, services, verification, simulation, etc.). See `www.yawl-system.com` for more information on the language and supporting tools.

Figure 1(a) shows two proclet classes. Proclet class *lab visit* consists of seven tasks and five ports and describes the process of taking a blood sample, ordering lab tests, and consolidating the results into a report. Proclet class *lab test* has five tasks and five ports and describes the life-cycle of a particular test. Note that for one blood sample many lab tests may be initiated. Hence there is a one-to-many relationship between *lab visit* and *lab test* as shown by the relationship *requires* in the class diagram in Figure 1(b). The two proclet classes are connected through two channels (*order system* and *HIS*). The mapping of ports to channels is shown in Figure 1(a).

The control-flow in each proclet class is expressed in terms of the YAWL notation. First, an instance (i.e. proclet) of class *lab visit* is created. After creation a blood sample is taken and lab tests are ordered. The output port of *select lab tests* has cardinality $*$, indicating that the performative is sent to potentially multiple recipients (i.e., lab tests). We will use $*$ to denote an arbitrary number of recipients, $+$ to denote at least one recipient, 1 to denote precisely one recipient, and *?* to denote no or just a single recipient. The performative is passed on via the channel *order system* and instantiates the proclet class *lab test* potentially multiple times, i.e., one proclet is created for every lab test that needs to be executed. The multiplicity of the output port of *select lab tests* is denoted by the number 1. This means that during the lifetime of an instance of class *lab visit* exactly one performative is sent via this port. The input port of the input condition of the *lab test* proclet has cardinality 1 and multiplicity 1. In each created *lab test* proclet a test is performed and the report is sent back to the "parent" *lab visit* proclet via the channel *HIS*. Note that the input port of task *receive result* has cardinality 1 and multiplicity $*$, indicating that multiple results may be received. Each performative received is stored in a knowledge base.

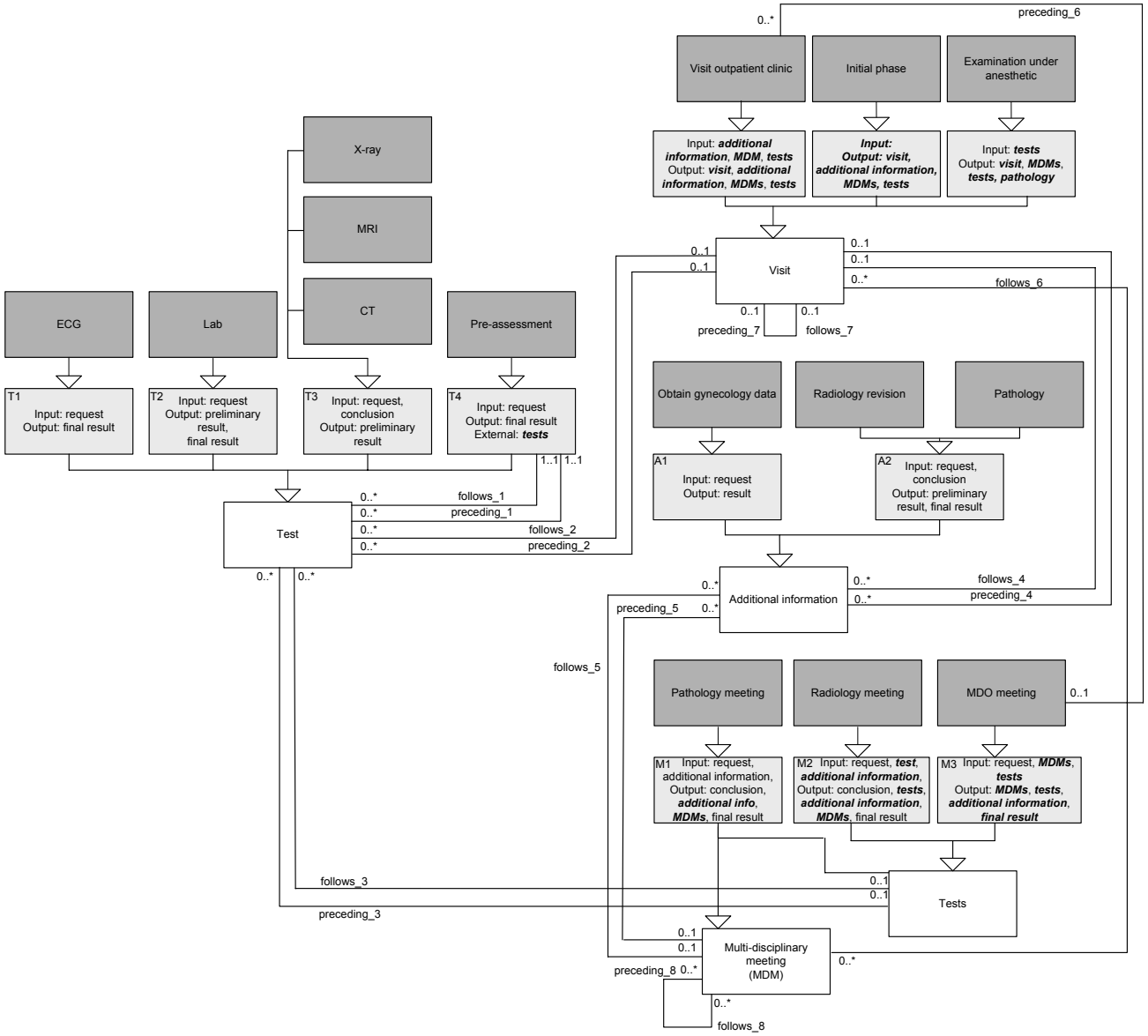


Figure 2: Class diagram outlining the concepts that exist within the healthcare process and their relationships.

The *lab visit* proclet continuously inspects this knowledge base and may decide to start analyzing the results to see if more tests are needed. If so, these are ordered in one go by the task *determine need for more tests*. Note that the cardinality of the output port of this task is $*$, i.e., in one step all relevant *lab test* proclets are triggered in order to perform any additional tests. After this the new results are sent from the various *lab test* proclets to the "parent" *lab visit* proclet. Finally, the task *finish lab visit* triggers the completion of all child *lab test* proclets that may have been initiated.

The example in Figure 1 is rather simplistic and hides many details, but at the same time it compactly illustrates the main features of proclets. For more details on the formalism we refer the reader to [1, 2]. Note that in Figure 1 interaction is modeled explicitly and there is no need to artificially flatten the process into a monolithic workflow, instead, the different levels of granularity are preserved. For more complex situations involving not only one-to-many relationships (as in Figure 1(b)) but also many-to-many relationships, it is still possible to model the overall process as a collection of intertwined loosely-coupled object life-cycles whilst it

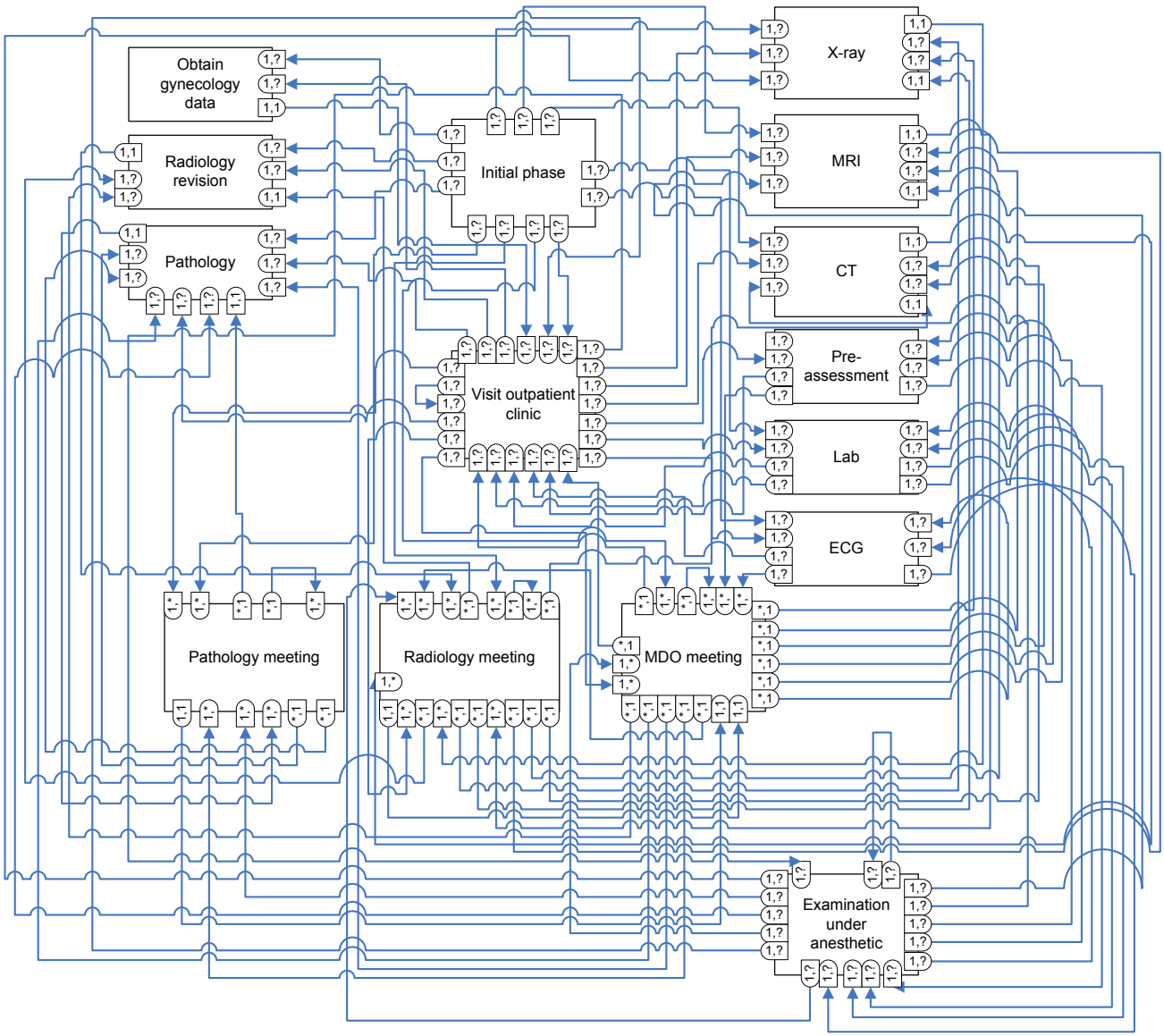


Figure 3: The proclets that are defined for the healthcare process and all the possible interactions between them.

would be virtually impossible to straightjacket the desired behavior into a monolithic workflow process.

Note that there is a strong correspondence between proclet classes and classes in a class diagram carrying the same name. A class in a class diagram outlines the data a proclet class carries with it and its relationship with other proclets. Via Object Constraint Language (OCL) expressions it is possible to access data of different proclets.

4 Application: Gynecological Oncology Workflow at the AMC

We have used proclets to model the gynecological oncology workflow at the Academic Medical Center (AMC) in Amsterdam. The AMC is the most prominent medical research center in the Netherlands and one of the largest hospitals in the country.

Given the complexity of the process and space limitations, we focus only on the main results. In total, 15

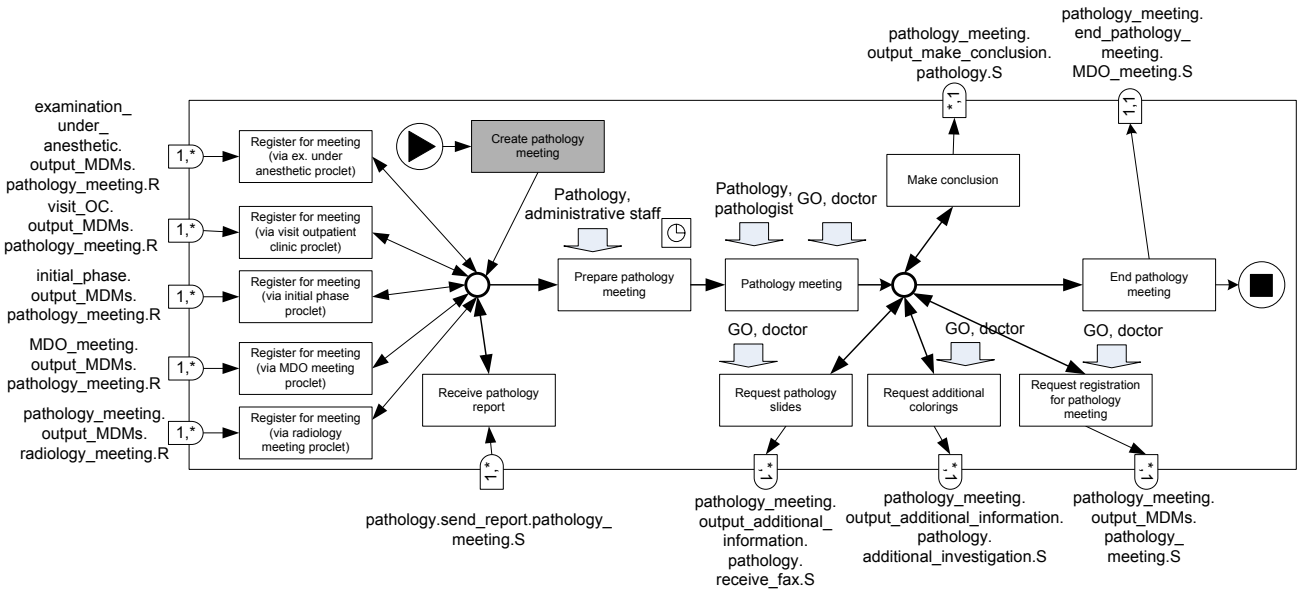


Figure 4: The *Pathology meeting* proclet.

proclet classes have been identified for the gynecological oncology workflow. Figure 2 shows a class diagram illustrating the relationships between the proclet classes. The dark rectangles correspond to concrete proclet classes. The inheritance relationships show which proclet classes have common features, i.e., the gray and white rectangles can be seen as abstract classes used to group and structure proclets. Moreover, as in Figure 1(b) the relationships between the various classes are depicted.

The 15 proclet classes identified in Figure 2 are connected to other proclet classes via the port and channel concepts. Figure 3 shows a high-level view of the interconnection structure. This diagram shows the complexity of the process. Given the different levels of granularity it is difficult (if not practically intractable) to flatten this structure into a monolithic workflow model.

Each of the rectangles in Figure 3 represents a proclet class and its ports. Figure 4 shows one example. Here the control-flow and the names of the ports and their cardinalities and multiplicities are shown. The proclet class models the weekly meeting in which the gynecological oncology doctors and a pathologist discuss the tissues examined by the pathologist that require further consideration. During this meeting, the tissues of multiple patients are discussed. For each weekly meeting, a separate proclet is created (*create pathology meeting*). In order to discuss a tissue of a patient, it first needs to be registered (*register for pathology meeting*). This can be done at different points in the process. However, as is indicated by the cardinality 1 and multiplicity $*$ of the associated ports, multiple patients can be registered using the same port. Note that after the weekly meeting (*Pathology meeting*), pathology examinations can be triggered for multiple patients. For example, as is indicated by the cardinality $*$ and multiplicity 1 of the associated port of task *Request additional colorings*, multiple tissues may be reinvestigated by a pathologist.

In this paper, it is impossible to give a more comprehensive description of the process and its 15 proclet classes. Instead, we refer the reader to [6] for an extensive description of the model.

5 Conclusion: Divide, Interact, and Conquer

In this paper we have advocated the use of proclets to overcome the problems related to monolithic workflows. Proclets are particularly suited to environments where processes are fragmented, interaction is important, and

tasks are done at different levels of granularity, e.g., healthcare processes where a visit to a doctor can trigger a wide range of tests and experiments. The next challenge is to provide advanced tool support for the design, analysis, and enactment of proclets. For example, proclets-based verification and process discovery pose interesting and challenging research questions.

References

[1] W.M.P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Workflow Modeling using Proclets. In O. Etzion and P. Scheuermann, editors, *7th International Conference on Cooperative Information Systems (CoopIS 2000)*, volume 1901 of *Lecture Notes in Computer Science*, pages 198–209. Springer-Verlag, Berlin, 2000.

[2] W.M.P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Proclets: A Framework for Lightweight Interacting Workflow Processes. *International Journal of Cooperative Information Systems*, 10(4):443–482, 2001.

[3] W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business Process Mining: An Industrial Application. *Information Systems*, 32(5):713–732, 2007.

[4] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su. Towards Formal Analysis of Artifact-Centric Business Process Models. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 288–304. Springer-Verlag, Berlin, 2007.

[5] R. Hull and J. Su. Research Challenges in Data-centric Workflow. Arlington, Virginia, 2009 (to appear).

[6] R.S. Mans, N.C. Russell, W.M.P. van der Aalst, A.J. Moleman, and P.J.M. Bakker. Proclets in Healthcare. BPM Center Report BPM-09-05, BPMcenter.org, 2009.

Similarity Search of Business Process Models

Marlon Dumas, Luciano García-Bañuelos*
University of Tartu
Estonia
marlon.dumas@ut.ee, lgarcia@ut.ee

Remco Dijkman
Eindhoven University of Technology
The Netherlands
r.m.dijkman@tue.nl

Abstract

Similarity search is a general class of problems in which a given object, called a query object*, is compared against a collection of objects in order to retrieve those that most closely resemble the query object. This paper reviews recent work on an instance of this class of problems, where the objects in question are business process models. The goal is to identify process models in a repository that most closely resemble a given process model or a fragment thereof.*

1 Introduction

As organizations reach higher levels of Business Process Management (BPM) maturity, they tend to accumulate considerable amounts of business process models – reportedly in the hundreds or thousands in the case of multinational companies. These models constitute a valuable asset to support business analysis and system design activities. In organizations with high degrees of BPM maturity, such process models are centrally managed in dedicated process model repositories that provide advanced browsing and search features.

In this paper we review recent developments pertaining to one particular search feature over process model repositories, namely similarity search [3]. In this context, similarity search is defined as follows: given a process model P (the *query*) and a collection of process models C, retrieve the models in C that are most similar to P and rank them according to their degree of similarity. Similarity search is relevant in the context of model maintenance. For example, before adding a model to a repository one needs to check that a similar model does not already exist so as to prevent duplication. Similarly, in the context of company mergers, process analysts need to find overlapping processes across the merged companies in order identify opportunities for consolidation.

Similarity search queries are defined with respect to a similarity measure between pairs of process models. The similarity between pairs of process models can be measured on the basis of three complementary aspects of process models: (i) the labels attached to tasks, events and other model elements; (ii) their graph structure; (iii) their execution semantics. The next three sections discuss a number of similarity search techniques classified according to these three criteria. We then summarize the results of an experimental evaluation covering a representative subset of these techniques. Finally, we outline some interesting open research directions.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*On leave from Autonomous University of Tlaxcala, Mexico, with funding from the European Regional Development Fund.

2 Label Similarity

One way of measuring the similarity between a pair of process models is by first computing an alignment between these models, that is, a relation between elements in one model and elements in the other model. For example, consider the two process models in Figure 1, captured using the Business Process Modeling (BPMN) notation.[1] A possible alignment is the one that matches task "Order" in the first model with the task with the same label in the second model, and task "Verify Invoice" with "Verification Invoice". Given such an alignment, the similarity between two models can be defined as a ratio between the size of the alignment, and the size of the process models. For example, we could define a similarity measure as follows: $\frac{2\times|A|}{|P|+|P'|}$, where A is an alignment and $|P|$ and $|P'|$ denote the number of compared model elements in models P and P' respectively. In the above example, this formula gives 0.66 if we only compare tasks (gateways are ignored).

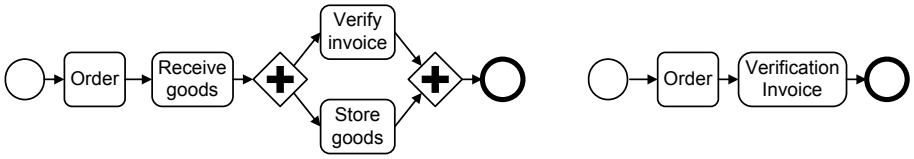


Figure 1: Sample pair of process models

Note that "Verify Invoice" does not perfectly match "Verification Invoice". So instead of giving a weight of one to this match, we should give it a weight equal to the similarity between these two labels (a number between zero and one). The resulting similarity measure is then $\frac{2\cdot\Sigma_{(n,m)\in A}\mathrm{Sim}_l(n,m)}{|P_1|+|P_2|}$ where Sim_l is a similarity measure between pairs of model elements. The similarity between model elements can be computed from their labels using syntactic similarity measures, semantic measures, or a combination of both. Syntactic measures are based on string-edit distance, n-gram, morphological analysis (stemming), and stop-word elimination techniques, whereas semantic techniques are based on synonym and other semantic relations captured in thesauri (e.g. Wordnet).[2] For example, "Verify Invoice" and "Verification Invoice" have a high syntactic similarity because they are almost identical after stemming, whereas "Verify Invoice" and "Check Invoice" have a high semantic similarity since "Verify" and "Check" are synonyms.

Variations of the above similarity measure have been proposed by Ehrig et al. [7] and Dijkman et al. [6]. In this latter work, the problem of finding a mapping between model elements is reduced to the linear assignment problem [9] for which efficient algorithms exist. In other work, label similarity measures are used in conjunction with structural or behavioural similarity measures (e.g. [8]).

3 Structural Similarity

Since process models are graphs, we can use graph matching as a basis for defining similarity measures. Specifically, given two process models, we can define their similarity as the opposite of their graph-edit distance [12] (i.e. one minus their normalized graph-edit distance). The graph-edit distance of two graphs is the minimum number of operations (insertions, deletions, substitutions) needed in order to transform one graph into the other. Unfortunately, the problem of computing the graph-edit distance is NP-hard. Thus, one needs to strike a tradeoff between computational complexity and accuracy. Below we review several possible tradeoffs.

[1] http://www.bpmn.org

[2] These linguistic matching techniques are also widely employed in the field of schema matching [14].

3.1 A* Algorithm

An exact method to compute the graph edit distance is by applying the A* search algorithm [12]. In the context of graph-edit distance calculation, the algorithm aims at constructing an alignment (mapping) between the nodes of the two graphs. Given a mapping, the graph-edit distance is calculated by considering that all non-mapped nodes have been deleted (or added), and that all pairs of nodes in the mapping correspond either to identical nodes or to node substitutions. The goal is to find the mapping with the smallest graph-edit distance. To this end, the A* graph matching algorithm starts with an empty mapping, which has a maximal edit distance since it implies that to go from one graph to the other, all nodes in one graph should be deleted and all nodes in the second graph should be added (no identical nodes and no substitutions). The algorithm incrementally constructs partial mappings of larger size, until it can no longer find a way of creating a larger mapping with a lower edit-distance. At each step, a number of new mappings are constructed by using the current partial mapping with the smallest edit-distance and adding new possible pairs to this mapping. It can be proved that this strategy leads to an optimal final mapping.

A major issue with this algorithm is the large number of partial mappings that must be maintained during the search – $O(m^n)$ in the worst case [12], where m and n are the numbers nodes in the two graphs that are being compared. Our own experiments have shown that this is problematic for process models with over 20 nodes [4]. We addressed this issue by forbidding pairs of nodes to be added to a mapping if their labels are too different from one another – but this breaks the optimality property of the algorithm.

In its basic form, the A* graph-edit distance algorithm only considers elementary graph-edit operations (deletion, insertion and substitution), meaning that only 1-to-1 node mappings are constructed. In [8] the set of graph-edit operations is extended with node splitting and node merging, i.e. N-to-M mappings are constructed. In [5], we observed that the number of partial mappings grows combinatorially when node splitting/merging operations are integrated and proposed an alternative two-step approach. First, the basic A-star algorithm is used to obtain a 1-to-1 mapping. In the second step, all deleted/inserted nodes are combined with adjacent nodes trying to improve the mapping. Combining a deleted node with a matched one is similar to node merging, while combining an inserted node with a matched one is similar to node splitting. In this way, the N-to-M mapping can be computed while maintaining the memory requirements of the 1-to-1 mapping approach.

3.2 Heuristic Search

Given the scalability limitations of exact graph matching techniques, several heuristics have been developed. In [4], we studied three such heuristics. The first one is a greedy heuristics, in which a mapping is created starting from an empty mapping and adding, at each iteration, the pair of most similar nodes that do not yet appear in the current mapping. The other two heuristics are closer to the A* graph matching algorithm in the sense that they explore several possible mappings (instead of just one current mapping as the greedy approach). However, these heuristics include a pruning function which is triggered when the number of currently considered mappings is larger than a given threshold. Only the most promising alignments are kept after a pruning phase.

3.3 Similarity Flooding

Given that process similarity can be measured on the basis of an alignment, the problem of similarity search can be related to alignment problems such as schema matching [14]. Similarity flooding is a graph matching technique that has been shown to yield good accuracy when applied to the problem of schema matching [11]. The idea behind similarity flooding is that a pair of nodes or edges of two graphs are similar when their adjacent elements are similar. The algorithm builds a matrix for similarity propagation which is updated iteratively by fixpoint computation. At the end, the matrix can be used to construct an alignment, e.g. construct a mapping with pairs of nodes whose similarity is greater than a given threshold. However, as stated in [11] the algorithm works fine for directed labelled graphs and degrades when edge labelling is uniform or absent.

Madushan et al. [10] have studied the application of similarity flooding for process model similarity search. In their work, process models are represented using an ontology-based notation in which process models are represented as graphs with labelled nodes and labelled edges. However, mainstream process modeling notations (e.g. BPMN) are such that edges have no labels, thus hindering the applicability of similarity flooding.

4 Behavioral Similarity

Process models define a part of the behavior of an organization. Therefore, another possibility for measuring their similarity is by measuring the similarity of their behavioral semantics. There are a number of behavioral semantics of business processes. For each of these behavioral semantics similarity metrics can be defined.

4.1 Comparison of traces

A simple way to define the execution semantics of a process model is in terms of the set of (completed) traces that it can accept. Assuming process models with finite sets of traces, we can define similarity measures in terms of this trace-based semantics. For example, we can define the similarity between two process models $P1$ and $P2$ as the ratio $\frac{2\times|T(P1)\cap T(P2)|}{|T(P1)\cup T(P2)|}$ where $T(P)$ is the set of traces generated by process P. However, this simple measure leads to unsatisfactory results. For example, the sets of traces of the two models shown in Figure 1 have an empty intersection. A more suitable alternative is to consider partial traces. Wombacher [16] studied the use of N-grams (partial traces consisting of n-items) as a basis to compare process models. Another technique based on a notion of "partially fitting traces" is presented in [1].

4.2 Simulation

A second way of defining behavioral semantics is in terms of a labelled transition system that captures all the states in which the process model can be, and all transitions that can cause the process model to change state. To determine if two process models are equivalent, we can then take the state-space of two process models and check if they can simulate one another (i.e. if they allow the same transitions in equivalent states). If the process models are not equivalent, we will find states which can be reached through the same sequence of transitions, and yet do not allow the same transitions. By counting such states, we can measure how dissimilar two process models are. This idea is applied by Nejati et al [13] in order to match statechart diagrams and could in principle be applied to process models.

4.3 Causal footprints

Trace- and state-based semantics aim to describe the behavior of a process as precisely as possible. However, their use can lead to performance problems due to large sets of traces and state explosion, while their level of precision is not required for measuring similarity. An approximation of the behavioral semantics of business processes would be sufficient for similarity measure. A possible approximation of this behavioral semantics is given by the concept of causal footprint [15].

A causal footprint of a business process is a triple (E, L_{lb}, L_{la}), where: E is the set of elements of the business process (e.g. tasks); $L_{lb} \subseteq \mathcal{P}(E) \times E$ is the set of look-back links, such that (lb, e) denotes that at least one element from lb must have occurred before e can occur; and $L_{la} \subseteq E \times \mathcal{P}(E)$ is the set of look-ahead links, such that (e, la) denotes that after e has occurred at least one element from la must occur. For example, if we identify the tasks in the example in figure 1 by the first letter of their label, a causal footprint for the leftmost process could be: $(\{O, R, V, S\}, \{(\{O\}, R), (\{R\}, V), (\{R\}, S)\}, \{(O, \{R\}), (R, \{V\}), (R, \{S\})\})\}$. Note that, if we add $(\{O\}, V)$ to the look-back links, the resulting causal footprint is still a valid footprint for the process. This illustrates that causal footprints are an approximate semantics.

The similarity search method based on causal footprints differs from other similar search methods in that it does not compute the similarity between each pair of process models. Instead, each process model in a collection is represented as a point in a vector space, and the problem of process model similarity search is reduced to the nearest-neighbour problem, that is, finding the nearest points to the query. This technique is often employed in the field of information retrieval for text documents. In the case of causal footprints, the dimensions (also known as the terms) of the vector space are the elements (e.g. tasks), the look ahead-links and the look-back links that appear in at least one business process in the collection. For a given business process, the values for the dimensions (also known as the weights of the terms) are determined based on the presence of the term in the process in question and the 'importance' of that term. Look-back and look-ahead links that consist of fewer elements are considered more important than those that consist of more elements.

5 Comparison

Table 1 summarises the results of an empirical evaluation of 8 similarity search techniques covering the three categories reviewed above. The evaluation involved 10 queries executed over a set of 100 process models. Details of the dataset and the evaluation method are given in [6, 4]. The table shows the mean average precision obtained for each technique across all 10 queries. Average precision is a measure commonly used to evaluate the quality of search techniques that return ranked lists of results [2]. The mean average precision for a given technique is the arithmetic mean of the average precisions obtained for each query using that technique.

Table 1: Mean average precision of representative search techniques (adapted from [4, 6])

Algorithm	**Mean avg. precision**	**Algorithm**	**Mean avg. precision**
Syntactic label sim.	0.8	A-star GM	0.86
Semantic label sim.	0.78	Sim. Flooding	0.56
Greedy GM	0.84	Causal Footprint	0.86
Heuristic GM	0.83	Text search engine	0.76

The table suggests that structural and behavioural techniques slightly outperform pure label-based ones. An exception is similarity flooding, which performs poorly, whereas it is known to have good performance in the context of schema matching. This can be explained by the fact that similarity flooding heavily relies on edge labels (in addition to node labels) whereas process models generally lack edge labels. The last row shows the result obtained by using a full-text search engine on the same set of queries. As expected, the average precision is lower than that obtained using any of the reviewed similarity search techniques (except similarity flooding).

6 Outlook

Existing process model similarity search techniques focus on process models composed of atomic tasks and connectors. Little attention has been paid to other process modelling constructs such as sub-process invocation and exception handlers. Perhaps more limiting is the fact that existing process model similarity search techniques tend to focus on the control-flow view of process models, neglecting data manipulation (e.g. data inputs/outputs) and resource allocation. Addressing this limitation is an avenue for further work.

As emphasized in this paper, the problem of similarity search of process models can be related to that of schema matching. Although some differences exist between these problems – particularly the general lack of edge labels in process models – there is an opportunity to transpose schema matching techniques to the process model similarity search problem. Several techniques reviewed in this paper are also found in automated schema matching tools. However, many other schema matching techniques have not yet been considered in the context of process model similarity search.

All process model similarity search techniques we know of employ linear search. In other words, the query model is compared to each model in the collection. An avenue for future work is to study the applicability and performance gains of graph indexing techniques [12] in the context of process model similarity search.

References

[1] W. van der Aalst, A.K. Alves de Medeiros, and A. Weijters. Process Equivalence: Comparing two process models based on observed behavior. In *Proc. of BPM 2006*, volume 4102 of *LNCS*, pages 129–144. Springer, 2006.

[2] C. Buckley and E.M. Voorhees. Evaluating evaluation measure stability. In *Proc. of the ACM SIGIR Conference*, pages 33–40, 2000.

[3] E. Chávez, G. Navarro, R.A. Baeza-Yates, and J.L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[4] R. Dijkman, M. Dumas, and L. García-Bañuelos. Graph matching algorithms for business process model similarity search. In *Proc. of BPM 2009*, Ulm, Germany, September 2009.

[5] R. Dijkman, M. Dumas, L. García-Bañuelos, and Reina Käärik. Aligning business process models. In *Proc. of EDOC 2009*, Auckland, New Zealand, September 2009.

[6] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling. Similarity of business process models: Metrics and evaluation. Working Paper 269, BETA Research School, Eindhoven, The Netherlands, 2009.

[7] M. Ehrig, A. Koschmider, and A. Oberweis. Measuring similarity between semantic business process models. In *Proc. of APCCM 2007*, pages 71–80, 2007.

[8] D. Grigori, J.C. Corrales, and M. Bouzeghoub. Behavioral matchmaking for service retrieval: Application to conversation protocols. *Inf. Syst.*, 33(7-8):681–698, 2008.

[9] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[10] Therani Madhusudan, J. Leon Zhao, and Byron Marshall. A case-based reasoning framework for workflow model management. *Data and Knowledge Engineering*, 50:87–115, 2004.

[11] S. Melnik, H. García-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm (extended technical report). Technical Report 2001-25, Stanford InfoLab, 2001.

[12] B. Messmer. *Efficient Graph Matching Algorithms for Preprocessed Model Graphs*. PhD thesis, University of Bern, Switzerland, 1995.

[13] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. Matching and merging of statecharts specifications. In *Proc. of ICSE 2007*, pages 54–63, 2007.

[14] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[15] B. F. van Dongen, R. M. Dijkman, and J. Mendling. Measuring similarity between business process models. In *Proc. of CAiSE 2008*, volume 5074 of *LNCS*, pages 450–464. Springer, 2008.

[16] A. Wombacher. Evaluation of technical measures for workflow similarity based on a pilot study. In *Proc. of CoopIS 2006*, volume 4275 of *LNCS*, pages 255–272. Springer, 2006.

Querying Future and Past in Business Processes *

Daniel Deutch Tova Milo

Tel Aviv University
{danielde,milo}@post.tau.ac.il

Abstract

A business process (BP for short) consists of a group of business activities undertaken in pursuit of some particular goal. Analysis of BPs bears two main flavors, namely analysis of future *and* past *executions. We intuitively explain these analysis goals and the models and algorithms employed to achieve them.*

1 Introduction

A business process (BP for short) consists of a group of business activities undertaken by one or more organizations in pursuit of some particular goal. It usually operates in a cross-organization, distributed environment and the software implementing it is fairly complex. *Standards* facilitate the design, deployment, and execution of BPs. In particular, the BPEL [5] standard (Business Process Execution Language), provides an XML-based language to describe the interface between the participants in a process, as well as the full operational logic of the process and its execution flow. BPEL specifications are automatically compiled into executable code that implements the described BP and runs on a BPEL application server. Processes execution is traced (logged), and their run-time behavior can be recorded in standard XML formats.

These standards not only simplify software development, but, more interestingly from an information management perspective, they also provide an important new *mine of information*. Queries about the BPs, that were extremely hard (if not impossible) to evaluate when the business rules were coded in a complex program, are now potentially much easier, for a declarative specification of the BP. Furthermore, sophisticated querying, that interleaves static analysis of the BP specification with queries over execution traces, can now be used for a variety of critical tasks such as fraud detection, SLA (service level agreement) maintenance, and general business management. This provides an essential infrastructure to both companies and customers: the former may optimize their business processes, reduce operational costs, and ultimately increase competitiveness. The latter may be presented with personalized analysis of the process, allowing them to make an optimal use of it.

For instance, consider a Business Process of an on-line store, that suggests electrical products of various kinds, brands and vendors. The web-site owner, on one hand, may wish to make sure that some business logic is kept, e.g. that no customers can make a product reservation without logging-in with their credit card number first; or in identifying the DVD brand that is most popular among customers buying a certain TV brand. The users, on the other hand, may wish to analyze the BP for identifying compatible TV and DVD of the lowest total price, or the most common choice of combined products.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*The research has been partially supported by the European Project MANCOOSI and the Israel Science Foundation.

In general, analysis of such BPs bears two main flavors. First, analysts are interested in analyzing executions that occurred in the *past*, for instance to identify trends, to make sure that business logic is maintained, etc. We note that executions are usually logged, forming *execution traces* that are kept in a repository. Thus, analysis over past executions is translated into queries over traces. There are two cruxes here: first, the size of a typical execution traces repository is extensively large, calling for query optimization techniques. Second, the traces often contain only partial information on the activities that were performed at run time, due to confidentiality, lack of storage space, etc. Thus query evaluation must be performed under terms of uncertainty.

The second flavor of analysis considers *future* executions. Namely, given the static BP specification, this kind of analysis, again operating under terms of uncertainty, aims at predicting the behavior of future users, e.g. to characterize *common* behavior of users, or to identify executions where the total induced cost to the customer is the cheapest, etc. This latter kind of analysis is in fact a *top-k* analysis, as it aims to find the k "best" execution flows, under some weighting function, and the main difficulties here stem from (1) the fact that the number of possible execution flows is very large, or even infinite in presence of recursion and (2) that the weight (e.g. likelihood, monetary cost, etc.) induced by actions made during the flow (e.g. product purchase), may be inter-dependent (due to probabilistic dependency, combined deals etc.).

To enable such reasoning, we first define models for capturing Business Process specifications, their execution flows and traces. These models should account for partial information and uncertainty of various flavors. Second, we define an intuitive query language that allows to rapidly form queries of interest over BP execution flows and traces. Third, we provide algorithms that allow for efficient query evaluation over BPs/execution traces under these models of uncertainty, and forth, we develop implementations that exploit these sound theoretical foundations for practical needs. We Intuitively describe here some of the main models and results.

2 Models

We give next a brief intuitive review of the main models standing at the center of our research on Business Processes analysis, and refer the reader to [12, 13] for precise definitions.

BP specifications and their executions. A BP specification is modeled as a set of node-labeled DAGs. Each DAG, intuitively representing a function, has a unique *start* (*end*) node with no incoming (outgoing) edges. Nodes are labeled by activity names and directed edges impose ordering constraints on activities. Activities that are not linked via a directed path are assumed to occur in parallel. The DAGs are linked through *implementation relationships*: an activity a in one DAG is realized via another DAG. We call such an activity *compound* to differentiate it from *atomic* activities having no implementations. Compound activities may have multiple possible implementations; the choice of implementation is controlled by a condition over user choices, variable values, etc., referred to as a *guarding formula*. A distinguished DAG, containing a single activity, stands as the BP root.

Figure 1(a) shows an example BP specification. The root S_0 has precisely one activity named `ShoppingMall`. The latter has as its implementation the DAG S_1, which describes a group of activities comprising user login, the injection of an advertisement, the choice of a particular store, and the user exit (possibly by paying). Within S_1, `Login` and `chooseStore` are *compound* activities; the `Login` activity has two possible implementations S_2 and S_3; the idea is that exactly one formula is satisfied at run-time, e.g., the user either logins as a regular or premium user and thus `Login` is implemented either by S_2 or S_3 respectively. Then the user chooses to pay with Mastercard or Visa, and authentications checks are made according to her identity (regular or premium), etc. Note that the specification is recursive as e.g. S_9 may call S_1.

An *execution flow* (abbr. EX-flow) of such BP is modeled as a nested DAG reflecting the execution order and implementation relation. Exactly a single implementation is chosen for each compound activity node. We model each activity occurrence by two nodes, the first (second) standing for its *activation* (*completion*) point. The chosen implementation appears in-between these two nodes (connected by specially marked *zoom-in edges*).

An example EX-flow is given in Figure 1(b). Regular (dashed) arrows stand for flow (zoom-in) edges. The user here logins as a regular customer and pays with a *Visa* Credit Card, then shops at the *BestBuy* store. There,

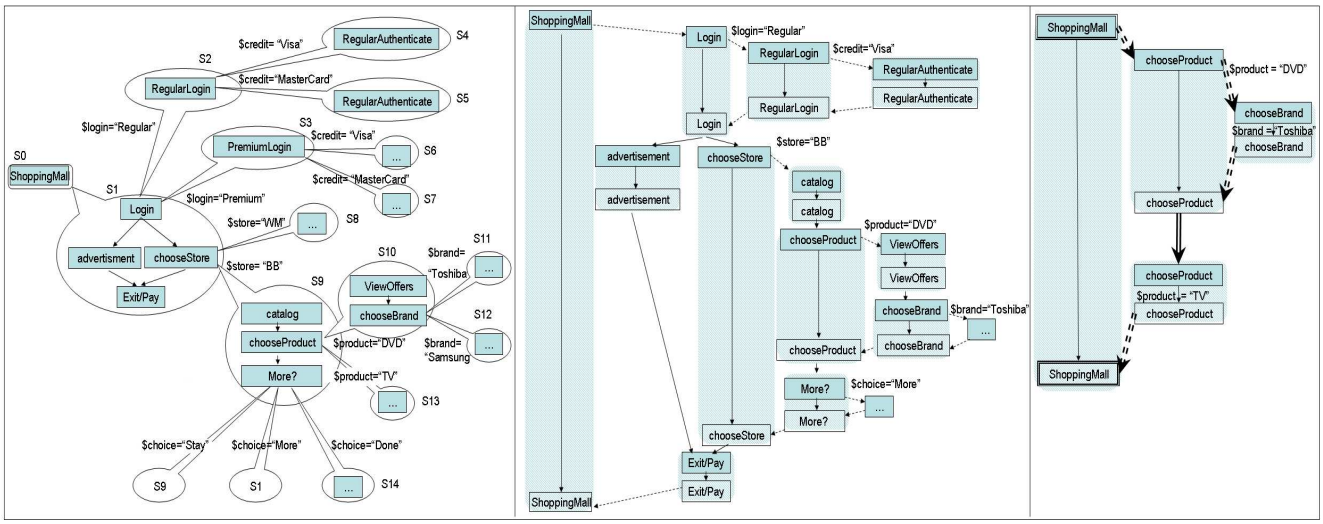


Figure 1: (a) Business Process (b) Execution Flow (c) Execution Pattern

she chooses to look for a *DVD player* and selects one by *Toshiba*, and continues shopping at the same store (we omitted the further purchases from the figure). Finally she exits and pays.

Tracing Systems. Tracing systems are employed for recording execution flows of Business Processes, and may vary in the amount of information that they record on the flow. In general, one can distinguish three families of tracing systems with decreasing amount of information: (i) *naive* tracing provides a complete record of the activation/completion events of all activities that had occurred during the EX-flow, (ii) *semi-naive* tracing where the activation/completion events are all recorded, but possibly with only partial information about their origin activity. For instance, such tracing system may record all login activities (Premium and Regular) as a generic `Login` name (and similarly for `Authenticate`), thus removing track of the different treatment of Premium and Regular clients, and (iii) *selective* tracing where, additionally, events for some selected subset of activities are not recorded at all. Such system may, for instance, completely omit the occurrences of login-related activities, thus removing all record of the fact that there are two types of users in this system.

Queries. Queries select EX-flows of interest using *execution patterns*, an adaptation of the tree/graph patterns offered by existing query languages for XML/graph-shaped data [9], to BP nested DAGs. Execution patterns may be uniformly interpreted as queries over past or future executions. An execution pattern is a nested DAG of shape similar to that of an EX-flow, but its edges may be either regular, i.e. match a single edge in the EX-flow, or transitive, i.e. match a path. Similarly, activity pairs may be regular or transitive for searching only in their direct implementation or zooming-in inside it, resp. Activity nodes may be marked by a special "Any" symbol, and then may be matched to BP nodes with any label, and finally some query part may be marked as output and is projected out (the query language may be enhanced by joins, negation etc. [12]). An example query is given in Fig. 1(c). The double-lined edges (double-boxed nodes) are *transitive* edges (activities). The query looks for EX-flows where the user chooses a DVD of brand Toshiba (possibly after performing some other activities, corresponding to the transitive edges), then chooses also a TV (of any brand). The `ShoppingMall` activity is transitive indicating that its implementation may appear in any nesting depth; `chooseProduct` is not transitive, requiring the brand choice to appear in its direct implementation.

3 Querying Future and Past Executions

We have reviewed the models standing at the center of our research on analyzing Business Processes, and we next (informally) define several main research problems in this context and give intuition for their solutions.

Querying Future Executions Recall that a BP specification defines a set of possible executions. We studied in [3] query evaluation over BP specifications, selecting execution flows of interest. The set of query results may be infinite, due to recursion. Continuing with our running example, there are infinite number of flows ending with the choice of Toshiba DVD, as the user may first make any sequence of choices and selections.

Thus the evaluation algorithm obtains a compact representation of all results, describing this recursive nature of qualifying flows. We note, however, that the query results are not equally interesting, and users are thus interested only in some subset of these, namely the *top-k weighted flows*, according to some weight function that fits the user interests. We thus introduce in [13] a refined, weighted model for execution flows of Business Processes, and compute the *top-k* execution flows of the given process, out of these conforming to the user query. The weighted model bears the following ingredients: first, we define a weight function $cWeight$ (corresponding to e.g. products prices, popularity of a link, etc.) over all possible implementation choices of compound activities; observe that the $cWeight$ of a given choice may vary at different points of the EX-flow and may depend on the course of the flow so far and on previous choices, e.g. the likelihood (price) of choosing a product may depend on previously purchased products, registration to membership club etc. Thus $cWeight$ accounts not only for the choice itself but also information about the history of the EX-flow thus far. Then, we aggregate $cWeight$ values of choices throughout a flow to obtain the weight of the entire flow (denoted $fWeight$, for flow weight). Following common practice [20], we require the aggregation to be monotonic w.r.t. the progress of the flow.

Results. We have shown that the extent to which the $cWeight$ of a given choice is dependent on the preceding choices affects the complexity of our problem. We use the term "history size" to measure this extent. In the general case where the history size is unbounded, top-k query evaluation is undecidable. Fortunately, such case is very rare, and moreover studies on the behavior of typical Web applications indicate this size to be relatively small (approximately 4) [28]. The complexity of our algorithm is then exponential in this (small) size (but we show this is unavoidable, unless P=NP), polynomial in the BP size, and linear in the output size.

We have also examined *optimality* properties of top-k algorithms for BP flows, and found that with plausible assumptions over $fWeight$, intuitively corresponding to "how strongly monotone" it is, one may provide (instance) optimal [20] algorithms for top-k query evaluation. We refer the reader to [14] for details.

Practical Applications. We have exemplified some of the practical applications of possible future flow analysis in [14], where the theoretical background explained above was exploited to design `ShopIT` (ShoppIng assitanT), a system that assists on-line shoppers by suggesting the most effective navigation paths for their specified criteria and preference. When the user starts her navigation in the site, she specifies her constraints and her weighting function of interest, and have the system compute and propose (an initial set of) top-k ranked navigation flows, out of these conforming to the constraints. The user then continues her navigation taking into account the presented recommendations, but may also make choices different than those proposed by the system, in the latter case `ShopIT` adapts its recommendations to the actual choices made by the user.

Querying Past Executions So far we have considered analysis of possible executions that have not happened yet. Naturally, much information can be also obtained from executions that had occurred in the past. Analysis of such information may be either done at run-time, *monitoring* [4] the execution, or over repositories of execution traces [12]. The latter kind of analysis is often done in two steps: the repository is first queried to select portions of the traces that are of particular interest. Then, these serve as input for a finer analysis that further queries and mines the sub-traces to derive critical business information [29]. Not surprisingly, *type information*, i.e., knowledge about the possible structure of the queried (sub-)traces, is valuable for query optimization [4]. Its role is analogous to that of XML schema for XML query optimization: it allows to eliminate redundant computations and simplify query evaluation. Such type information is readily available, as the BP specification, for the original traces, but not for the intermediary traces selected by queries. This calls for *Type Inference*. When the analysis tool expects particular data type, we would also like to verify that the sub-traces selected by queries conform to the required type. Such *Type Checking* is thus a second challenge.

An additional kind of queries over past executions considers *recovery* of the flow that is most likely to actually happened at run-time, given a partial trace. There are two practical cases to consider here: first, the simpler case where the tracing system itself, e.g. which activities are omitted or renamed are known. Note that there may still be (infinitely) many origins to a given log. Second, the tracing system itself may be unknown, in which case there may also be exponentially many tracing systems to consider.

Results. We showed in [12] that the less detailed (and thus less restrictive) the execution traces are, the more efficient type inference can be: it can be done in time polynomial in the size of the input type (with the exponent determined by the size of the query) for selective trace types, but may require time exponential in the size of input type (even for small queries) with semi-naive trace types, and may not be possible at all if all trace types are naive. This signals *selective trace types* as an "ideal" type system for BP traces, allowing both flexible description of the BP traces as well as efficient type inference. Type checking, on the other hand, incurs exponential data complexity for (semi-)naive trace types , and is undecidable for selective trace types. This indicates that static type checking is probably infeasible, and calls for run-time analysis [4].

Retrieval of execution flows given their trace was studied in [13], where we show that our query evaluation algorithms can be adapted to retrieve the most likely flow given a trace. We then consider the case of an unknown tracing system, and avoid enumeration of the exponentially many tracing systems by proving a *small world* theorem, showing that only a polynomial number of representative options need to be tested.

Practical Applications. We have demonstrated in [4] a query language and system for monitoring business processes, that allows users to visually define monitoring tasks, using a simple intuitive interface similar to those used for designing BPEL processes. The monitoring tasks are translated to very efficient BPEL processes that run on the same execution engine as the monitored processes.

4 Related Work

First, let us consider our choice of data model and query language. These are argued [12] to be more intuitive for BP developers than e.g. temporal logics and process algebras, as they are based on the same graph-based view used by commercial vendors for the specification of BPs.

A variety of formalisms for (probabilistic) process specifications exist in the literature, with applications in Verification [19], Natural Language Processing, Bioinformatics, etc. Among those, we mention Hidden Markov Models (HMMs) [18], (Probabilistic) Recursive State Machines (PRSMs) [2], and (Stochastic) Context Free (Graph) Grammars (CFG, CFGG) [10]. While HMM extends Finite State Machines, PRSMs and SCFGG describe nested structures similar to that of BPs. There are several differences between works on these models and analysis of BPs. First, in terms of expressive power, probabilistic variants of these models typically assume independencies (markov property, context freeness) between probabilistic events. Second, most of the analysis works over such processes (e.g. [15, 16, 6]) use temporal logic, which may not capture our query language. Intuitively, this is because our query language bears *structural features* (allowing e.g. to capture graph homomorphism). In contrast, work on querying CFGGs [10] generally uses strongly expressive (structural) logics such as MSO (Monadic Second Order Logic), incurring high evaluation complexity. Also note the analogy between Naive (Semi-naive, Selective) trace types and *bracketed* (*parenthesis* [25], *context free* [31]) string languages.

Type Checking and Type Inference, discussed here for BP execution traces, are well studied problems in functional programming languages for database queries [27, 7], and for XML [26]. The unique structure of BP traces incur additional difficulties (e.g. in contrast to XML, here type checking is harder than type inference).

We have also discussed above top-k queries for execution flows of BPs. Top-k queries were studied extensively in the context of relational and XML data [22]. Notably, [20] presented an instance-optimal algorithm for top-k queries that aggregate individual scores given to joining tuples. Difficulties specific to the BP settings are that (1) the size of a given flow, thus the number of aggregated scores, is unbounded (2) the particular properties of the weight functions are unique to EX-flows and (3) the number of items (EX-flows) that are ranked is infinite. Note that while an infinite setting also appears in top-k queries over *streamed* data [24], works in this context aggregate over a *bounded size sliding window*, whereas we consider aggregation over flows of unbounded size.

Ranking by likelihood was also studied in several other settings, e.g. *Probabilistic Databases* (PDBs) [11, 30] and *Probabilistic XML* [1, 23]. For example, [30] and [23] study the problem of retrieving the top-k query results for queries over PDBs and Probabilistic XML, resp. Note that in contrast to relational data and XML, our model for BP flows allows representation of an *infinite* number of items, out of which the top-k are retrieved.

Last, we briefly mention a complementary line of tools whose input is a set of run-time generated *traces*

(logs), and may generate a probability distribution over the events affecting the flow; such distribution may then serve as input for our analysis. The common OLAP (online analytical processing)-style analysis [17] offers users various multi-dimensional views of data, and correlations in-between. The Business Process Intelligence (BPI) [21] project is another branch of the work on analyzing execution flows, inferring causality relationships between execution attributes using data mining techniques such as classification and association rule mining. Such retrieved relationships may be used as input to our analysis.

5 Conclusion

We have depicted here models and algorithms for capturing and analyzing Business Processes and their past and future executions, and demonstrated that declarative languages for specifying and querying such processes allow for important analysis and optimization tasks, that were not possible in the absence of such languages. Further challenges include, among others, extensions of the query language to include additional useful features such as negation, joins, etc.; considering other settings for top-k analysis; and designing further practical applications.

References

[1] S. Abiteboul and P. Senellart. Querying and updating probabilistic information in xml. In *Proc. of EDBT*, 2006.

[2] R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T. Reps, and M. Yannakakis. Analysis of recursive state machines. *ACM Trans. Program. Lang. Syst.*, 27(4), 2005.

[3] C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes. In *Proc. of VLDB*, 2006.

[4] C. Beeri, A. Eyal, T. Milo, and A. Pilberg. Monitoring business processes with queries. In *Proc. of VLDB*, 2007.

[5] Business Process Execution Language for Web Services. http://www.ibm.com/developerworks/library/ws-bpel/.

[6] T. Bultan, J. Su, and X. Fu. Analyzing conversations of web services. *IEEE Internet Computing*, 10(1), 2006.

[7] J. Bussche, D. Gucht, and S. Vansummeren. A crash course on database queries. In *Proc. of PODS*, 2007.

[8] F. Casati, M. Castellanos, N. Salazar, and U. Dayal. Abstract process data warehousing. In *Proc. of ICDE*, 2007.

[9] D. Chamberlin. Xquery: a query language for xml. In *Proc. of SIGMOD*, 2003.

[10] B. Courcelle. The monadic second-order logic of graphs. *Inf. Comput.*, 85(1), 1990.

[11] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *Proc. of VLDB*, 2004.

[12] D. Deutch and T. Milo. Type inference and type checking for queries on execution traces. In *Proc. of VLDB*, 2008.

[13] D. Deutch and T. Milo. Top-k projection queries for probabilistic business processes. In *Proc. of ICDT*, 2009.

[14] D. Deutch, T. Milo, and T.Yam. Goal-oriented web-site navigation for on-line shoppers. In *Proc. of VLDB*, 2009.

[15] A. Deutsch, M. Marcus, L. Sui, V. Vianu, and D. Zhou. A verifier for interactive, data-driven web applications. In *Proc. of SIGMOD*, 2005.

[16] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. Verification of communicating data-driven web services. In *Proc. of PODS*, 2006.

[17] J. Eder, G. E. Olivotto, and W. Gruber. A data warehouse for workflow logs. In *Proc. of EDCIS*, 2002.

[18] Y. Ephraim and N. Merhav. Hidden markov processes. *IEEE Trans. Inf. Theory*, 48(6), 2002.

[19] K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic state machines. In *Proc. of TACAS*, 2005.

[20] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4), 2003.

[21] D. Grigori, F. Casati, M. Castellanos, M.Sayal U .Dayal, and M. Shan. Business process intelligence. *Comp. in Industry*, 53, 2004.

[22] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4), 2008.

[23] B. Kimelfeld and Y. Sagiv. Matching twigs in probabilistic xml. In *Proc. of VLDB*, 2007.

[24] N. Koudas and D. Srivastava. Data stream query processing: A tutorial. In *Proc. of VLDB*, 2003.

[25] R. McNaughton. Parenthesis grammars. *J. ACM*, 14(3), 1967.

[26] T. Milo and D. Suciu. Type inference for queries on semistructured data. In *Proc. of PODS*, 1999.

[27] J. C. Mitchell. *Foundations for Programming Languages*. MIT Press, 1996.

[28] P. L. T. Pirolli and J. E. Pitkow. Distributions of surfers' paths through the world wide web: Empirical characterizations. *World Wide Web*, 2(1-2), 1999.

[29] D. M. Sayal, F. Casati, U. Dayal, and M. Shan. Business Process Cockpit. In *Proc. of VLDB*, 2002.

[30] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *Proc. of ICDE*, 2007.

[31] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

Business Processes Meet Operational Business Intelligence

Umeshwar Dayal, Kevin Wilkinson, Alkis Simitsis, Malu Castellanos
HP Labs, Palo Alto, CA, USA

Abstract

As Business Intelligence architectures evolve from off-line strategic decision-making to on-line operational decision-making, the design of the backend Extract-Transform-Load (ETL) processes is becoming even more complex. We describe the challenges in ETL design and implementation, and the approach we are taking to meet these challenges. Our approach is centered around a layered methodology that starts with modeling the business processes of the enterprise, and their information requirements and service level objectives, and proceeds systematically through logical design to physical implementation. A key element of this approach is the explicit specification of a variety of quality objectives (we call these collectively the QoX objectives) at the business level, and the use of these objectives to drive the optimization of the design at the logical and physical levels.

1 Introduction

Today's Business Intelligence (BI) architecture typically consists of a data warehouse that consolidates data from several operational databases and serves a variety of querying, reporting, and analytic tools. The back-end of the architecture is a data integration pipeline for populating the data warehouse by extracting data from distributed and usually heterogeneous operational sources; cleansing, integrating, and transforming the data; and loading it into the data warehouse. The traditional data integration pipeline is a batch process, usually implemented by extract-transform-load (ETL) tools [1, 2]. Traditionally, BI systems are designed to support off-line, strategic "back-office" decision-making where information requirements are satisfied by periodic reporting and historical analysis queries. The operational business processes and analytic applications are kept separate: the former touch the OLTP databases; the latter run on the data warehouse; and ETL provides the mappings between them. We have learnt from discussions with consultants who specialize in BI projects that often 60-70% of the effort goes into ETL design and implementation. As enterprises become more automated, data-driven and real-time, the BI architecture must evolve to support *operational Business Intelligence*, that is, on-line, "front-office" decision-making integrated into the operational business processes of the enterprise [3]. This imposes even more challenging requirements on the integration pipeline. We describe some of these challenges and propose a new approach to ETL design to address them.

To motivate our approach, we use a simple, example workflow. Consider a hypothetical, on-line, retail enterprise and a business process for accepting a customer order, fulfilling and shipping the order and booking the revenue. Such an *Order-to-Revenue* process involves a number of steps, utilizing various operational (OLTP) databases and an enterprise data warehouse (Figure 1(a)). Assume a customer has been browsing the retailer web site and adding items to a shopping cart.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

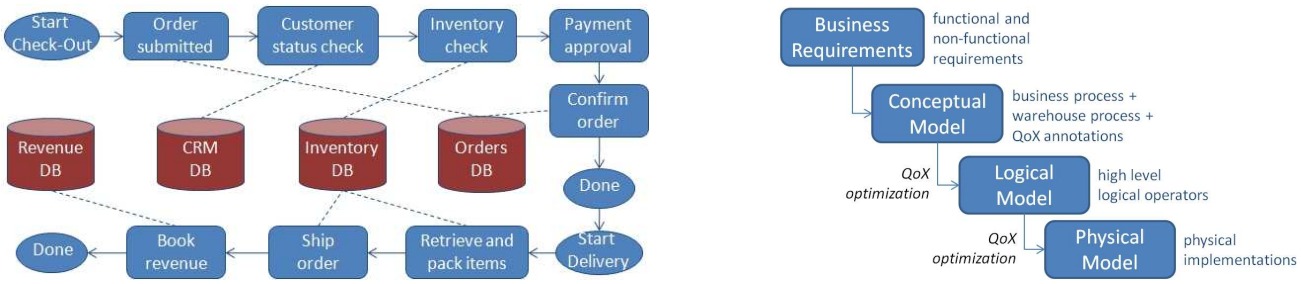


Figure 1: (a) Order-to-Revenue business process and (b) Layered methodology for ETL

Eventually, the customer is ready to make a purchase, which initiates the CheckOut process. This submits an entry to the order database and then the customer status is checked to validate the order. Next, the inventory database is checked to ensure the product is in stock. At this point, the order can be fulfilled so the customer payment is processed. Once confirmed, the Delivery process is initiated. The items are retrieved from inventory and packed. Finally, the order is shipped and the order revenue is added to the financial revenue database. In our example, revenue is not counted until the order is shipped.

Operational BI imposes new requirements on ETL that are difficult to meet using today's conventional approach. We describe three challenges.

End-to-end operational views of the enterprise. In the conventional BI architecture, the data warehouse provides an historical view of the enterprise; e.g., it can be used to provide reports on weekly sales, the top selling items, seasonal trends or to build customer segmentation models. The architecture may even incorporate an operational data store to provide a near-real time view of transactional data in the OLTP databases. Still, it does not provide the integrated, near real-time view of the entire (end-to-end) enterprise needed by the new breed of operational BI applications. For example, suppose we want to make special offers to particular customers based on their purchasing history, recent browsing actions, today's revenue, and current inventory. This operation requires data from the OLTP databases (current inventory, customer's recent browsing actions), the data warehouse (customer segment, purchasing history), and in-flight data (today's revenue, including orders that haven't yet shipped) that may be in staging areas on its way to the data warehouse. Such enterprise views are very complicated to design, implement, and maintain, and current BI tools provide little support for them.

Design by objective. The focus for ETL so far has been on correct functionality and adequate performance, i.e., the functional mappings from data sources to warehouse must be correct and their execution must complete within a certain time window. However, a focus on just functionality and performance misses other important business objectives (e.g., recoverability, maintainability, reliability) that, while harder to quantify, are needed for a successful ETL deployment. This is especially true for operational BI where there may be a wide range of competing objectives. Fraud detection may require a high degree of provenance for certain parts of the ETL flow. High reliability may be needed for parts of the flow related to revenue, e.g., the loss of click-stream data is acceptable whereas the loss of payment is not. Consequently, what is needed is a more general approach where the ETL design is driven by objectives and can be optimized considering their tradeoffs.

Design evolution. A typical ETL engagement consumes many months starting with business requirements and design objectives, infrastructure surveys, conceptual and logical design, and culminating in a physical design and implementation. In an ideal world, the requirements never change. In the real world and especially in operational BI, requirements change rapidly as the business evolves and grows. For example, assume the order-to-revenue process was implemented with an expected latency for the (external) payment approval process, but months later the credit agency doubles the latency. This affects the entire downstream ETL pipeline and perhaps substantial redesign to maintain service level objectives. A methodology that requires many additional months to adapt to such a change would not be useful in operational BI.

In [4], we describe a layered methodology that proceeds in successive, stepwise refinements from high-

level business requirements, through several levels of more concrete specifications, down to execution models (Figure 1(b)). At each level of design, different qualities (QoX objectives) are introduced or refined from higher levels [5]. This layered approach presents opportunities for QoX-driven optimization at each successive level. By connecting the designs and objectives at successive levels of refinement we are better able to track objectives and rapidly generate new designs as the business evolves.

An important feature of our design methodology is the use of business process models for the conceptual (high level) design. This has several advantages. It provides a unified formalism for modeling both production (operational) processes such as Order-to-Revenue as well as the processes that populate the data warehouse and intermediate enterprise states. It enables ETL design starting from a business view that hides the low-level implementation details and therefore facilitates the specification of SLAs (Service Level Agreements) and metrics by business analysts.

In brief, our approach leverages business process models to enable operational business intelligence. It captures end-to-end views of enterprise data and associates them with high-level design objectives, which are used to optimize the ETL processes and implementation. In the following sections, we elaborate on these ideas.

2 QoX-driven integrated business views

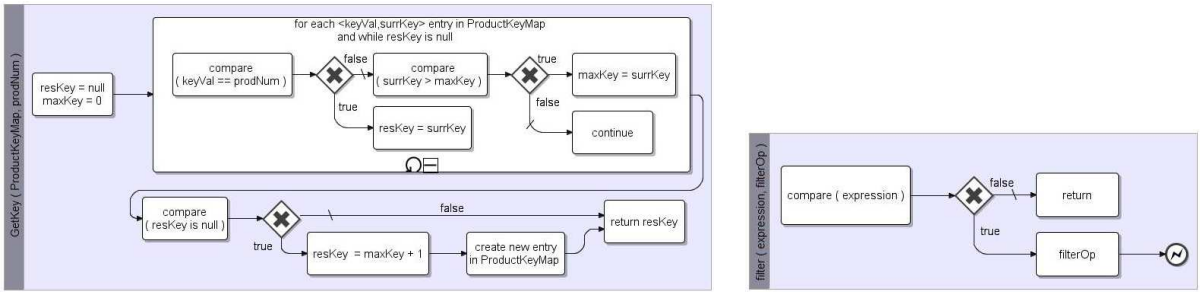
This section discusses our approach to obtain an integrated, end-to-end view of the enterprise at the conceptual level and, from that, how to get a logical ETL design. The facts in a data warehouse define the business objects and events of interest for decision-making. The data sources for these facts are objects in the OLTP databases manipulated by operational business processes, e.g., CheckOut, Delivery. ETL flows define the mappings between the source objects and the facts in the warehouse. However, ETL tools today do not support the modeling of these mappings at the conceptual level (i.e., in terms of business objects). Rather, they support only logical ETL design at the level of objects such as tables, indexes, files, communication links. We believe that modeling ETL at a conceptual level can benefit operational BI in a number of ways. First, it enables users of the warehouse to see the provenance of the warehouse data in business terms they understand. Second, it provides an up-to-date view of the enterprise by exposing the intermediate state of the ETL pipeline, the data under transformation before it is loaded to the warehouse. This intermediate view creates new opportunities for real-time operational applications in that it can be used at any time for operational decision-making, avoiding a wait for the warehouse to be refreshed. Third, this conceptual model can be used to derive the logical model for ETL.

Our approach is to use BPMN[1] for the conceptual model. Since BPMN can also be used to model operational business processes, this provides a common formalism to model the complete information supply chain for the enterprise. For each fact (and dimension, view, etc.) object in the warehouse, there is a corresponding BPMN *business fact process* that shows how that object is created out of the operational business processes. Probes inserted in the operational business process are used to send messages to all fact processes that need data from that point in the process. But, there are three challenges with using BPMN.

The first challenge is that to derive the logical ETL flow, we need a way to map fragments of a BPMN fact process to the corresponding logical ETL operators. To do this, we employ three techniques: (1) an expression language to specify method invocation in the nodes of a fact process; expressions can be easily mapped to logical ETL operators; (2) macro expansion; e.g., Figure 2(a) illustrates an example for the frequent ETL operation of surrogate key generation; and (3) templates for mapping specific patterns to ETL operations; e.g., a compare method followed by a true/false branch where one branch terminates the flow is recognized as an ETL filter operator (see Figure 2(b)). These examples for macro expansion and templates are discussed later. Note that a legend for BPMN notation is provided in Figure 3.

The second challenge is that BPMN models process flow, but ETL is fundamentally a data flow. So, we need to augment our BPMN diagrams to convey the necessary data flow information, in particular, the input,

[1] Business Process Modeling Notation, http://www.bpmn.org/



(a) (b)

Figure 2: Example (a) macro expansion for a SK operator and (b) template for filters

output, and parameters of ETL operators. To do this, we augment our BPMN fact process with input, output and parameter schemas as follows: we assume that each BPMN message is described by an XML schema; when a message is received by a method, it is included as part of the node's input schema. We use the BPMN annotation capability to annotate methods with XML schemas for other inputs, outputs and any other parameters. In going from process flows to data flows, we also have to consider that a business process typically creates a new process instance per business event or transaction, whereas an ETL process typically uses a single data flow to process a batch of records. Consequently, to create a batch of objects, we introduce a spool method that inputs a sequence of objects and outputs a set of objects.

The third challenge derives from our ultimate goal of producing an optimized physical ETL implementation. For this purpose, we must also incorporate the QoX objectives into the BPMN fact processes and the derived logical ETL flows.

Example. We illustrate our ideas by presenting the BPMN diagram[2] for a *DailyRevenue* fact process (see Figure 3). This process computes the revenue for each product sold per day. We assume that the business requirements include a QoX measure for freshness. This specifies how frequently the warehouse should be updated (e.g., daily for high freshness, monthly for low freshness, etc.). Thus, the DailyRevenue process is initiated once per freshness interval. Recall that revenue is counted when a product is shipped. Thus, a probe must be added to the Delivery business process to send the order details to the DailyRevenue process. The spool method separates order details into an order summary and its constituent lineitems and accumulates this data for the freshness period. Afterward, the set of spooled lineitems is forwarded to the partitioning method which groups lineitems by date and product number. For each group, it creates an instance of the DailyRevenue process and sends the process its lineitems.

The DailyRevenue process does the work of creating one new fact. It iterates through the lineitems, aggregating their details. The GetKey method converts production keys to surrogate keys. Note that GetKey method is a macro expansion and the corresponding BPMN diagram is shown in Figure 2(a). Internal orders, denoted by a null shipping address, should not be counted as revenue so they are filtered out. The template that is used to recognize this pattern as a filter operation is shown in Figure 2(b). When all lineitems in the group are processed, the totals are added to the warehouse as a new fact.

Given the DailyRevenue process description along with annotations for the data flow, the logical ETL flow, DailyRevenue_ETL, can be generated using a relatively straightforward translation. The details are omitted in this paper. The logical ETL flow is depicted in Figure 4(a). Here we use the notation of an open source ETL tool (i.e., Pentaho's Kettle). Designing a tool-agnostic, logical ETL flow language is itself an interesting challenge.

Operational BI example. As discussed, the BPMN fact process enables new opportunities for real-time decision-making without waiting for warehouse refresh. As an example, suppose the on-line retailer wants to include special offers in the shipping package such as product rebates, free shipping or discounts on new

[2]Our BPMN diagrams are intended for presentation and are not necessarily entirely consistent with the specifications.

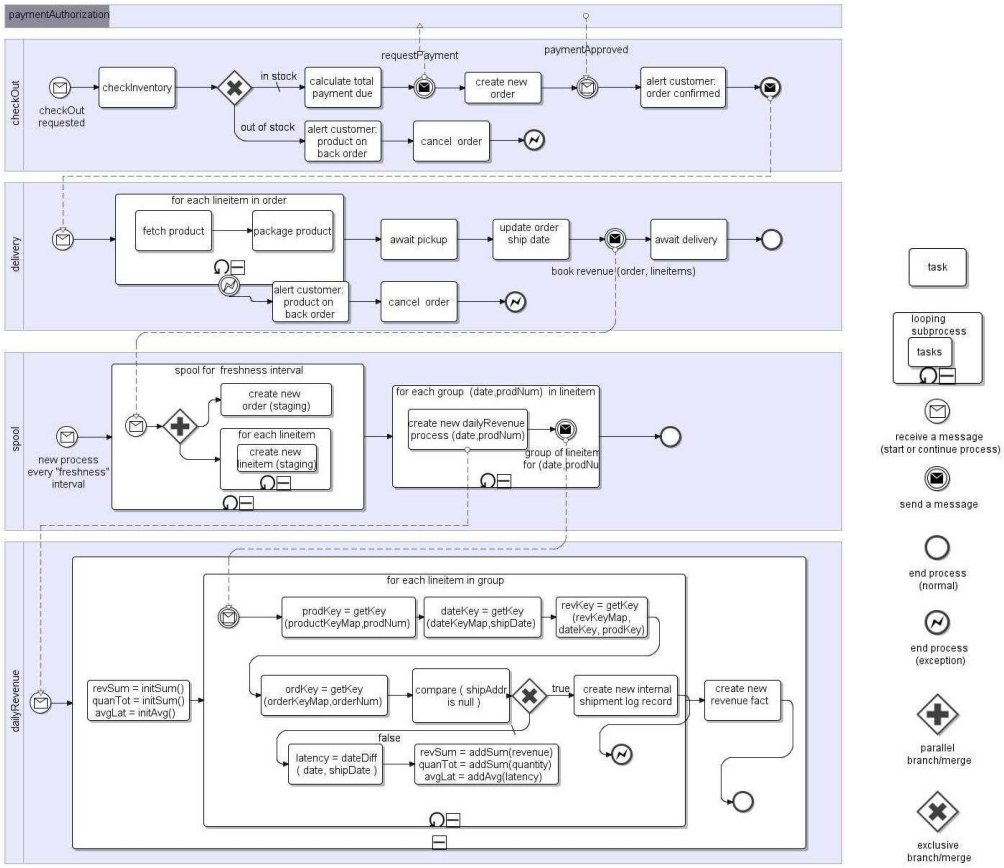


Figure 3: Example operational business processes (CheckOut, Delivery) and business fact process (DailyRevenue, spool)

products. And suppose these offers depend on today's current daily revenue. The current day's revenue is not available in the warehouse so the special offers process must access the intermediate state of the enterprise.

To accomplish this, we need to link the RetrieveAndPack method in the Delivery process to a new process, ShippingInserts (Figure 4(b)). This new process returns a set of offers to include in the shipping package according to the business rules. In our example, we assume rebates are offered for orders that had an exceptional delay and free shipping is provided for orders that exceed twice the average order amount. We need to adjust the freshness interval to ensure that the DailyRevenue is updated hourly (or possibly more frequently) so that the running totals can be tracked. Note this requires a slight modification of the DailyRevenue fact process (not shown) to maintain a running total, i.e., it should process multiple groups from the spooler and only update the warehouse once in each refresh cycle.

3 QoX-driven optimization

After having captured the business requirements and produced an appropriate logical ETL design, the next step involves the optimization of the ETL design based on the QoX metrics. The challenges in doing this include the definition of cost models for evaluating the QoX metrics, definition of the design space, and algorithms for searching the design space to produce the optimal design. In [5], we showed how tradeoffs among the QoX objectives can lead to very different designs. Here, we summarize some of the optimization techniques and tradeoffs.

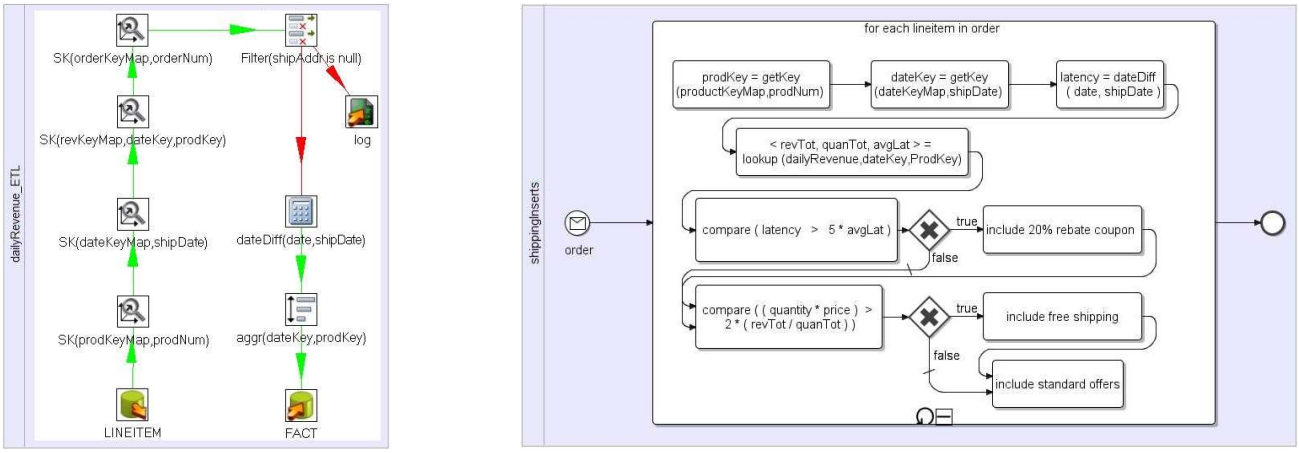


Figure 4: (a) Logical ETL flow and (b) Real-time offers

Optimizing for *performance* (i.e., improving the execution time of an ETL flow) typically exploits algebraic rewriting (e.g., postponing the getKey method for revKey until after the aggregation to decrease the amount of data) or flow restructuring (e.g., partitioning the flow for parallelization). The optimizer must select among many choices for rewriting and restructuring the flow (e.g., how and where to partition). An ETL workflow may fail due to operational or system errors. Designing for *recoverability* typically involves the addition of recovery points at several places in the workflow from which the ETL process resumes after a failure and continues its operation. However, I/O costs are incurred for maintaining recovery points, and hence there are tradeoffs between recoverability and performance. The optimizer must decide on the number and placement of recovery points. Sometimes, we cannot afford to use recovery points, as for example when high freshness is required. In such cases, it might be best to design for *fault-tolerance* through the use of redundancy (i.e., replication, failover, diversity). There are many challenges such as determining which parts of the workflow to replicate and achieving a balance between the use of recovery points and redundancy. *Freshness* is a critical requirement for operational BI, and designing for freshness is an important area of research [6, 7]. Alternative techniques here include the use of partitioned parallelism, the avoidance of blocking operations and recovery points, streaming implementations of transformation operators such as joins (e.g., [8]) or the loading phase (e.g., [9]). Also, scheduling of the ETL flows and execution order of transformations becomes crucial [10].

Optimizing for each of the QoX metrics is a challenge by itself because of the large design space. However, the main challenge is to consider these implementation alternatives together in order to optimize against a combination of QoX objectives specified by the business requirements. Figure 5 illustrates some of the tradeoffs in optimizing for freshness, performance, recoverability, and fault-tolerance for a specific flow [5]. The solid blue line represents the baseline performance of the original flow. For improving freshness (i.e., reducing the latency of an update at the target site - y axis), we need to increase the number of loads (x axis). In doing so, the best performance (i.e., lowest latency) may be achieved with parallelization (black dotted line). Using recovery points hurts freshness more or less depending on whether we use a high (green line with larger dashes) or a low (red line with smaller dashes) number of recovery points, respectively. The alternative of using triple modular redundancy (red line with larger dashes) for fault-tolerance achieves nearly the same level of freshness as the original design.

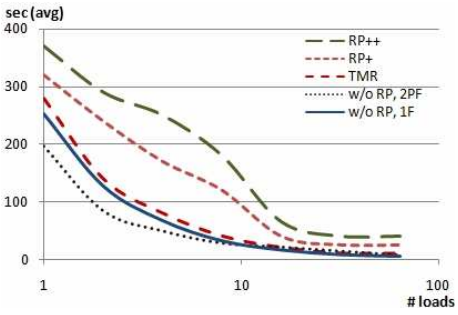


Figure 5: Example design space for QoX metrics [5]

4 Summary

We described a layered methodology for designing ETL processes in operational Business Intelligence systems. A key feature of this methodology is the use of a unified formalism for modeling the operational business processes of the enterprise as well as the processes for generating the end-to-end information views (e.g., business facts) required by operational decision-making. The methodology starts with a conceptual specification from which the logical definition and physical implementation are systematically derived. Included in the conceptual model is the specification of QoX objectives, which drive the design and optimization at the logical and physical levels.

Our ongoing research addresses the following problems: (1) Conceptual modeling formalism: We have illustrated our approach using BPMN. However, as we discussed, BPMN is not especially well suited to expressing the data flow mappings for constructing information views. Also, the modeling formalism must support annotation of the process and data flows with quality objectives. (2) Logical modeling formalism: This must include the typical operators required by the mappings, but must be agnostic to any specific implementation engine, and it must enable QoX-driven optimization. (3) Automatic derivation of the logical model from the conceptual model. (4) QoX-driven optimization: This includes a cost model for expressing the QoX metrics, and algorithms for optimizing against these metrics. (5) Techniques for validating the design against the business level specifications. (6) Techniques for evolving the design as business level requirements change.

References

[1] W. Inmon, *Building the Data Warehouse*. John Wiley, 1993.

[2] R. Kimball and J. Caserta, *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. John Wiley, 2004.

[3] C. White, "The Next Generation of Business Intelligence: Operational BI," DM Review Magazine, May 2005.

[4] U. Dayal, M. Castellanos, A. Simitsis, and K. Wilkinson, "Data Integration Flows for Business Intelligence," in *EDBT*, 2009, pp. 1–11.

[5] A. Simitsis, K. Wilkinson, M. Castellanos, and U. Dayal, "QoX-driven ETL Design: Reducing the Cost of ETL Consulting Engagements," in *SIGMOD Conference*, 2009, pp. 953–960.

[6] D. Agrawal, "The Reality of Real-time Business Intelligence," in *BIRTE (Informal Proceedings)*, 2008.

[7] P. Vassiliadis and A. Simitsis, *New Trends in Data Warehousing and Data Analysis*. Springer, 2008, ch. Near Real Time ETL, pp. 1–31.

[8] N. Polyzotis, S. Skiadopoulos, P. Vassiliadis, A. Simitsis, and N.-E. Frantzell, "Supporting Streaming Updates in an Active Data Warehouse," in *ICDE*, 2007, pp. 476–485.

[9] C. Thomsen, T. B. Pedersen, and W. Lehner, "RiTE: Providing On-Demand Data for Right-Time Data Warehousing," in *ICDE*, 2008, pp. 456–465.

[10] L. Golab, T. Johnson, and V. Shkapenyuk, "Scheduling Updates in a Real-Time Stream Warehouse," in *ICDE*, 2009, pp. 1207–1210.

Volume versus Variance: Implications of Data-intensive Workflows

Michael zur Muehlen
Stevens Institute of Technology
Hoboken, NJ 07030
mzurmuehlen@stevens.edu

Abstract

A data-intensive workflow is a process that is faced with a large volume or highly variable forms of information. The increasing digitization of office processes, the use of workflows to integrate publicly available data sources, and the application of workflow technology to scientific problem solving have led to an increased interest in the design and deployment of data-intensive workflows. This paper discusses the notion of data-intensive workflows and outlines the implications of increasing data volumes and variances for the design of process-aware applications.

1 Introduction

Process-oriented Information Systems have been developed for more than 30 years [6]. Their development is based on a behavioral view of the enterprise as a system. This view defines an organization as an information processing entity that transforms inputs into outputs according to a set of procedural rules. These procedural rules can be observed, (re-)defined, and managed. This process perspective on the organization is not a new concept. In management science its roots can be traced back to the early 1930s in Europe [8] and the late 1950s in the United States [7]. The restructuring of organizations along their core processes has demonstrated benefits in particular among functionally fragmented organizations that were striving to offset the side-effects of worker specialization and functionally-oriented departments. The efficiency benefits of process-driven application design have made workflow systems a readily available application in many organizations, to the extent that many middleware systems and packaged applications contain workflow technology.

In contrast to this focus on organizational behavior, the development of functional Information Systems has traditionally been dominated by data management concerns. Beginning with accounting and record-keeping systems, the need to make large data sets accessible and manageable has led to significant innovation in areas such as database technologies, query languages, and lately, the semantic markup of information using technologies such as *RDF* and *OWL*. The increasing maturity of data access standards such as *RSS* and *SOAP*, combined with authentication technologies for distributed environments is making significant data sets easily accessible. In the United States new data sharing initiatives such as *data.gov*, *usaspending.gov* and *recovery.gov* make government information publicly available using standardized access mechanisms and data formats.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

The physical distribution of audio and video materials on optical or magnetic media is continually diminishing in favor of digital downloads. The increasing availability of digital information poses questions for the design of workflow applications, which are traditionally based on the notion of a well-defined and limited set of information that has to be routed between process participants, be they people, applications, or services. What does it mean for a workflow to be *data-intensive*?

2 Classes of Workflow Data

To answer this question we need to consider the types of data that surround a typical workflow application. An established classification for data handled in the context of workflow applications has been defined by the Workflow Management Coalition Glossary [1]:

Content Data (sometimes referred to as *application data*) relates to the (user-defined) payload of a workflow instance. This data is either supplied to the workflow management system by the initiator of the workflow when the workflow instance is created (i.e., the initial /payload), or it is created by individual activities throughout the life of the workflow instance. Content data in its most general form has no bearing on the execution path of a workflow instance. A typical example would be the line item description of an order or the content of a photograph submitted as evidence in an insurance claim.

Workflow Data refers to those data objects that are produced by the workflow execution environment itself during the enactment of the workflow instance. This class of data relates to technical information, such as audit trail information that documents the instantiation, invocation and completion of activity instances [2], user log-on and log-off information, or recovery data that a workflow server might generate in order to be able to recover after a failure situation. While this information is generally not considered for decision-making at the instance level, it can be used for applications such as server health checks, load balancing, and - combined with content data - process analytics.

Workflow-relevant Data relates to those data objects that can affect the routing logic of a workflow application, both in terms of control flow decisions (such as which outgoing sequence flow of a data-based XOR gateway to activate), as well as in terms of task assignment (i.e., which performer a particular work item should be offered to). If the decision logic of a workflow application is based on few stable attributes it is often encoded in the process model itself. If the decision logic requires the evaluation of multiple attributes, rules, or changes frequently it is increasingly located in a separate rules management system. Workflow-relevant data may be part of the externally generated payload (such as the status of a customer) or it can be generated by the workflow application during the execution of the workflow instance itself. A typical example is information about the starting user of the workflow instance. This data is not known until the workflow instance has been created, but in many cases it is being used to assign activities to the initiator of the workflow instance.

In many cases a workflow application plays the role of a mediation system that enables disparate systems or services to interact. If in the process of mediation the data generated by the source system or service is transformed so that it can be read by the destination system or service the management of provenance information plays an important role, and the traceability of transformations may become a requirement. In this sense, the workflow application may become an author of data that would otherwise be classified as application data. The boundary between data that is exposed to the workflow application for routing decisions, and pure application data is increasingly blurry, so that the main distinction in this taxonomy is between data that is generated by the workflow application and data that is consumed by the workflow application.

3 Data Volume versus Content Variance

The data surrounding different workflow applications varies both in terms of *volume* and *variance*. A data-intensive workflow application can be defined as a process-oriented information system that is designed to process data in large volumes and/or data with highly variable characteristics.

Item	Data Volume		Content Variance	
	Low Volume	**High Volume**	**Low Variance**	**High Variance**
Workflow Data	The workflow application generates a small amount of audit data (low fidelity)	The workflow application generates a large amount of audit data (high fidelity)	The structure of the workflow audit trail is similar from one workflow instance to the next	The structure of the workflow audit trail can vary widely between instances
Workflow-relevant Data	The control flow of a workflow instance is determined based on a limited set of data	The control flow of a workflow instance is determined based on a large set of data	The control flow of a workflow instance is determined based on predictable datatypes	The control flow of a workflow instance is determined based on varying data types
Content Data	Each workflow instance processes a small amount of data	Each workflow instance processes a large amount of data	The data types are stable from one workflow instance to the next	The data types can vary widely between workflow instances
Performer Data	Few performers participate in the execution of a workflow instance	Many performers participate in the execution of a workflow instance	The set of performers is stable between workflow instances	The set of performers can vary widely between workflow instances
Context Data	The execution of the workflow is relatively independent of context information	The execution of the workflow is highly dependent on context information	Workflow instances are executed under similar circumstances	Workflow instances are executed under highly varied circumstances

Figure 1: Data Volume versus Content Variance

Large data volumes can relate to the type of data that as well as to the number of data objects that are routed by the workflow engine to different processing stations. Examples for large data types are applications that process large images or movie files, such as digital scans from medical devices, satellite imagery, or applications that post-process video streams. Even though each workflow instance may only transport a limited number of these objects, the size of each object can be in the *MB* to *GB* range. If the workflow application moves these objects across a network the requirements for network throughput increase with the number of concurrent workflow instances. If no mediation is required, the workflow application may refer to these objects using URIs without moving them physically. However, if the workflow application has to mediate data formats (e.g. encoding of materials for different end user devices) it may be necessary to physically transport large data volumes. Examples for a large number of data objects are high-volume workflow applications such as trading systems, traffic monitoring applications, or telephony applications. Even though the size of each data object is very limited, the number and frequency of these objects, combined with requirements for low latency information flow puts an emphasis on the data processing capacities of a workflow applications.

Content variance relates to the rate with which the structure of content data changes. A workflow application can be regarded as data-intensive if it has to operate in an environment where the payload varies highly between workflow instances. A typical example are intelligence applications where a large variety of information sources are routed to analysts based on content correlation and the context in which they were gathered. An analyst may be presented with textual, visual, and auditory information, and the composition of data in each workflow instance may differ widely. Workflow applications with a high degree of content variance tend to favor the use of case-management techniques, where an individual actor is provided with the total set of information related to the workflow instance, but is given some leeway to decide the appropriate course of action.

4 Participant Volume versus Participant Variance

A different aspect of data-intensive workflow applications is the number and variation of workflow participants. Some workflow applications are enacted in stable environments with a defined number of humans, systems, or services that participate in the execution of each workflow instance. Other workflow applications may allow an unforeseen number of participants to interact with it (workflow for the crowd). And yet other workflow applications may interact with a defined number of participants, but the capabilities of these participants may vary with each workflow instance.

An example of high participation volume is a process where a complex problem is broken into smaller units which are assigned to individual agents to solve. Amazon.com's mechanical turk service is an example of such a *crowdsourcing* application. In this example a large task (such as the analysis of a large number of images) is broken into small, identical subtasks that are assigned to individual actors. The number of actors in the system is not know ahead of time and can be influenced through the use of bidding mechanisms and the creation of task-dependent incentives. If a workflow is performed by a large number of casual users the design of user interfaces has to consider in particular how an untrained user can learn the task at hand, whereas a workflow task that is regularly performed by a select group of specialists can be tailored to the specific abilities of the specialist, with less regard to common accessibility.

An example of high participation variability is the military process of Close Air Support. This process describes how ground troops may request the assistance of airborne assets in the fulfillment of their mission [3]. An instance of this process may involve fixed wing or rotary wing aircraft, which may have different operating capabilities and communication devices. In addition, these assets may be prepared to assist (pre-planned scenario) or may be diverted from another mission (ad-hoc scenario). Despite these differences, the overall structure of the Joint Close Air Support process remains the same, but its execution needs to be tailored to the specific communication capabilities and requirements of the participating actors. The military has solved this issue by standardizing the content of the messages exchanged between process participants, rather than standardizing the medium through which these messages are communicated.

5 Context Information

Workflows may be instantiated in different environments. These environments may affect the reliability of the services or actors that the workflow enactment service depends upon. Taking the aforementioned Joint Close Air Support example, this process can be invoked in a daytime environment with clear visibility, reliable communication links between participants, and a technical infrastructure that allows the exchange and confirmation of broadband information such as video streams recorded by aircrafts. In another setting the process can be invoked at night, in a mountainous terrain, where image processing equipment is unavailable, communication bandwidth is limited, and the accuracy of information is much less certain.

If a workflow is executed in the same or similar context its design and development can be performed in a closed environment, and it can be optimized to perform under these anticipated circumstances. This is typically the case when the enactment environment is entirely under the control of the organization that performs the workflow, as are back-office processes and certain transactional processes where the provider can dictate data formats and interaction patterns to the requester, e.g. insurance claim scenarios.

In cases where a workflow is executed under different circumstances, and where these circumstances have a direct impact on the routing, decision logic, or performance of individual tasks, the workflow designer has fewer options to optimize the performance of the process a priori. In these cases the workflow design needs to provide event handling capabilities to react to changes in the environment and mechanisms that allow for the flexible routing, performance, and assignment of tasks (see e.g., [9]).

6 The Role of Semantics

The formal representation of the data context of workflow applications in a semantic format such as *RDF-S* or *OWL* can be beneficial in case of high content and/or context variability. The semantic annotation of content and context information allows for the following:

If a process modeling grammar such as BPMN is encoded in a semantic markup format, then a process model can be automatically compared to the grammar in order to identify modeling mistakes. Since the presence of modeling mistakes has been documented even in commercial reference models [5], such an evaluation would assist workflow developers in minimizing the risk of failed workflow instantiations and executions.

If a process model is encoded in semantic markup format, then a process instance can be evaluated for compliance against the process model. This might be useful if the process instance is not derived directly from the model (as is the case in many production-type workflow systems), but is rather a dynamically evolving execution path that is constrained by a declarative process modeling formalism, such as GPSG [4].

If the payload of a process is described in a semantic markup format, then the process designer may be able to specify the process logic by referencing the semantic classes of information that the workflow is designed to process, rather than the actual data format that needs to be ingested and transformed. This would allow for a separation of the processing concerns (what the workflow is designed to achieve) from the execution concerns (how the transformation has to take place).

If the audit trail information of the process is described in a semantic markup format, the designers and users of process analytics may be able to evaluate workflow instances that did not process the same data formats, yet were enacted on information with similar semantics.

The use of semantic markups and ontologies for the design of process-aware information systems has seen an increased interest recently, as demonstrated e.g. by the EU-funded IP-SUPER project (*www.ip-super.org*) and remains a promising area of research to allow for workflow applications that can perform well in heterogeneous data environments.

Item	Data Volume		Content Variance	
	Low Volume	**High Volume**	**Low Variance**	**High Variance**
Workflow Data	Little processing and storage requirements for analytics information, however: limited insight	Increasing processing and storage requirements for analytics information, however: rich insight	Allows for the design of stable analytics views and reporting components	Requires adaptive transformation logic to feed analytics information, views must be configurable
Workflow-relevant Data	Control-flow rules may be specified as part of the process model	Control-flow rules should be handled by separate rules logic	Process debugging and automated decision making are possible	Manual decision making may be required if data types cannot be anticipated
Content Data	Lightweight, fast workflow applications	Increasing demands for storage and network bandwidth	Predictable data formats can be used to optimize data flow	Variable data formats may lead to case-management-based workflow solutions
Performer Data	Organization structures can be designed based on process logic	Workflow organization model may have to reflect real-world organization	User interface screens can be tailored to the specific abilities of performers	User interface screens have to be easy to learn by new performers
Context Data	Testing, simulation and deployment of the workflow application can be performed in a closed environment	Event-processing capabilities are required to react to context data changes	Workflow design can be optimized to a particular execution scenario	Workflow design and execution capabilities need to be flexible to accommodate context changes

Figure 2: Implications of Data Volume and Content Variance

7 Implications for Workflow Application Designers

Data volume and content variance can have a significant impact on the design of workflow applications. Workflow designers should be aware how big and how stable the different classes of data are that their application interacts with. While the volume of data typically affects network throughput and storage requirements, the variability of content information has a more pronounced impact on design decisions. We have provided a classification schema for the different classes of data typically encountered in the context of workflow applications, and discussed the implications of changes in data volume and content variance. Data-intensive workflows can be encountered in many different disciplines, but their management may be simplified by a common set of design principles based on the characteristics of data that makes the workflow data-intensive.

References

[1] D. Hollingsworth: The Workflow Reference Model. Document Number TC001003, Workflow Management Coalition, Winchester, UK, 1995.

[2] Workflow Management Coalition: Business Process Analytics Format - Draft Specification. Document Number TC-1015, Version 1.0, 2009, Cohasset, MA.

[3] U.S. Department of Dense: Joint Tactics, Techniques, and Procedures for Close Air Support (CAS). Joint Publication 3-09.3, 2 September 2005. Washington, DC, 2005.

[4] N. S. Glance, D. G. Pagani, and R. Pareschi, Generalized Process Structure Grammars (GPSG) for flexible representations of work, Boston (MA), 1996, ACM, pp. 180-189.

[5] J. Mendling, Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, Springer, Berlin et al. 2008.

[6] M. Zisman, Representation, Specification, and Automation of Office Procedures, University of Pennsylvania, Philadelphia (PA), 1977.

[7] E. D. Chapple, and L. R. Sayles, The Measure of Management. Designing Organizations for Human Effectiveness, Macmillan, New York (NY), 1961.

[8] F. Nordsieck, Grundlagen der Organisationslehre, C. E. Poeschel Verlag, Stuttgart, 1934.

[9] P. Dadam, M. Reichert et al., Towards truly flexible and adaptive process-aware information systems, Proc. UNISCON, 2008, pp. 72-83.

Integrating Data for Business Process Management

Hong-Linh Truong and Schahram Dustdar
Distributed Systems Group, Vienna University of Technology
Email:{truong,dustdar}@infosys.tuwien.ac.at

Abstract

To be able to utilize Web-scale resources for business processes and to adapt these processes to the dynamic change of environments, business process management suites/systems (BPMS) must be able to gather, integrate and manage various types of data in the lifecycle of business processes. We discuss the issue of integrating data for business process management. We provide an overview on the current state of how data is integrated into business process management and recommend new directions.

1 Introduction

For an organization, its business processes are the key to its success. Therefore, it is of paramount importance for the organization to have a powerful business process management (BPM) approach that enables it to rapidly create new, capable business processes and to improve and adapt existing business processes to the changing environment in which the processes are executed. The term "BPM" is used loosely here and we mean that BPM includes all activities that help the organization to achieve such capable and adaptable business processes. Over the last a few years, we have observed that, instead of relying only on a single organization's resources (software services and humans) to perform process activities, business processes in an organization have increasingly relied on Web-scale resources by utilizing and assembling multiple-organizational or individual resources. This paradigm shift has been supported by emerging technologies, such as the SOA and Software-as-a-Service (SaaS) model and the integration of humans into business processes (e.g., BPEL4People/WS-HumanTask[1]). Together with more complex business requirements, this paradigm shift makes business processes more complex and difficult to manage and understand. This change has a profound impact on technologies used in BPM, affecting all phases of the lifecycle of business processes, including process design, modeling, execution, monitoring, and optimization.

To be able to utilize the Web-scale resources for business processes and to adapt these processes to the dynamic change of environments, BPM suites/systems (BPMS) must be able to gather, integrate, and manage various types of data in order to efficiently manage processes. It means that many types of data should be integrated and associated through the lifecycle of business processes. To date, those types of data that characterize the main five themes, named process strategy, process architecture, process ownership, process measurement, and process improvement [11], of a business process are voluminous, complex and difficult to collect and manage. In particular, by utilizing SaaS and user-generated services and by allowing mass customization, BPMS

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] `http://xml.coverpages.org/bpel4people.html`

should be able to closely interact with SOC management frameworks and utilize data provided by these frameworks. However, it is very challenging to integrate BPMS and SOC management frameworks together. This paper examines which types of data are needed, why they are important, and how they are currently integrated and utilized for business processes (Section 2). We also suggest to develop a unified model and techniques for linking and managing integrated data for BPM during the evolution of business processes (Section 3).

2 Data Associated with Business Processes

In order to examine types of data that should be integrated for BPM, let us consider data required in the lifecycle of a business process that has been studied in literature. Table 2 provides further description of these types.

Type	**Phases**	**Description**	**Example**
Strategy	Design, Modeling	including data about business strategy and IT strategy, expected SLA and KPIs for the process	"by 2010, more than 50% activities should be performed by third-party Web services", "utilize only shipping services offering the shipping time less than 3 days"
Capability	All	including capabilities associated with a service, a human or a human's offerings, as well as concerns describing when and how the service or human can be used.	"providing company credits", "the average response time to credit requests is 2 days", "a service provided by a media freelancer in Vienna", "an expert in BPEL design"
Contract	All	including software service contracts, contracts for business artifacts/data, business compliance rules	"pay-per-use with service credit", "the law enforcement is the European court", "the data is free, but owned by the provider"
Patterns and processes	All	including common patterns, discovered process models and patterns, existing processes relevant to the process.	"delegation pattern", "one-to-many service interaction pattern", "3 similar processes in the repository"
Business rules	Execution	including rules specifying business constraints and compliance policies used to execute the process	"a failed service is replaced by only similar services inside Europe", "when the order process delays 2 days, send a notification"
Performance	Monitoring, Optimization, Design	including IT performance and business performance metrics. Furthermore, historical performance and monitoring data for offline optimization and refinement, QoS metrics of services and humans	"the availability in the last 10 days is 70%", "today's number of failures is 3" , "the average response time in the last 100 calls is 4 days" , "the number of successful completed activities in the last 10 days is 70"

Table 2: Types of possible data relevant to a business process

Through the lifecycle of a business process, various types of data are needed for different purposes. All these types of information are important for deciding the techniques used in, for example, structure and behavior design, runtime execution and monitoring, and off-line and runtime optimization. Our first observation is that a majority of BPMS support only certain types of data associated with capability and performance. They allow the user to design services and humans in business processes, and monitor and optimize the processes, e.g., [8]. Our second observation is that there is a track record on advanced process patterns and model analysis, such as [16, 1, 15]. Furthermore, many performance data is collected, such as [17]. However, many tools and BPMS cover only a small part of these types of data. There is a lack of frameworks and techniques to support BPMS to harness multi-organizational and individual resources for business processes. We believe that the following points should be addressed in BPMS:

- integrating and managing data about resources capability and availability in the Web-scale
- ensuring contract compliance for business processes consisting of services and humans in the Web-scale

- managing and integrating reusable patterns and processes, and performance data

In our view, integrating these types of data into and providing them as an internal feature of BPMS are extremely important as they help to solve challenges raised by the gap between the business level and IT level as well as to adapt business processes to changing environments. In the following, we discuss these main points.

2.1 Integrating Capability and Availability Data in the Web-scale

Two main types of resources for a business process are software services and humans. Data describing the capability and availability of services and humans is critical for all phases of a business process. It can substantially improve the design and adaptation of business processes. For example, the design of a business process could start from scratch if we are not aware of existing resources that can be assembled. Capability data is virtually required in all lifecycle phases, but most BPMS use the capability data only for the design phase, some for the optimization phase at runtime. While data about few services and humans might be enough for the design (to prove that the functionality is working), rich data about services and humans would increase the possibility to analyze what-if scenarios in the process modeling and to adapt processes to situations at runtime. Unfortunately, managing Web-scale resources for business processes is still at an early stage. Most BPMS just assume that the designer knows where the resources are. But this assumption is hard, if not impossible, to be hold when business processes are relied on Web-scale resources.

BPMS used by an organization face many challenges when integrating resources outside the organization, in particular, commodity software services provided as SaaS and humans acting as external services. Many BPMS have already supported the design and execution of business processes whose activities are performed by services, in particular, Web services, but do not offer mechanisms to search and find relevant services. Furthermore, this search will not be limited to functional aspect of services (e.g., account management or payment) but also other concerns, such as licensing, location, and trust. With respect to the role of humans in business processes, humans can be actors who design the process as well as who perform activities in the process. In the first aspect, managing a person's capabilities, skill and team as well as his/her participation in the business process design could potentially help to improve the design of business processes by quickly locating the right person for the right task. This aspect requires BPMS to be integrated with social and team networks which is lacking in most BPMS. In the second aspect, some techniques, such as BPEL4People, have enabled the integration of humans in the Web as a part of business processes. They are, however, very premature. For example, they allow to specify human and software activities but neglect the discovery of human resources. In most cases, the user has to enter the information about human services. Harnessing mass user-generated services as one way of outsourcing, e.g., empowered by freelancers, is currently not in the focus of BPMS. To overcome these problems, it is necessary to integrate BPMS with service discovery and registry capabilities and social networks of humans. From the management point of view, solutions based on cloud computing, such as Platform-as-a-Service (PaaS) for BPMS, could also address these issues. PaaS providers could be responsible for managing resource capabilities, while an organization just focuses on utilizing these resources (e.g., the Boomi platform[2] for software services).

2.2 Integrating Service- and Data-Related Contract Concerns

The design and execution of business processes have to ensure that resources used and artifacts manipulated and produced will comply to certain contracts. Currently, the evaluation of contract concerns associated services and artifacts in business processes is focused only on a small number of concerns, notably service-related QoS/SLA metrics, e.g., [5], and mostly at the design time. However, resources and artifacts are bound to many other concerns, such as quality of data, intellectual property rights, law enforcement, data distribution, data disposition, to name just a few. These concerns are typically associated with DaaS (data-as-a-service), such as StrikeIron[3]

[2] www.boomi.com

[3] http://www.strikeiron.com/strikeironservices.aspx

and Amazon Web services[4], which are utilized by organizations to retrieve and store business artifacts. These concerns are important for both service and data aspects compliance in business processes.

With current composition and data mapping techniques, BPMS allow the designer to functionally compose data sources and services in an easy manner. However, the designer lacks a mechanism to validate whether these data sources and services being complied with the above-mentioned expected concerns. Partially, it is due to the fact that services and data are not well described, but also this research topic is not in the focus of existing BPMS. To overcome these issues, we should enrich current QoS/SLA metrics and techniques with quality of data, service and data licensing, and data governance metrics. Furthermore, compliance evaluation techniques for these concerns should be integrated into all phases of BPM.

2.3 Integrating Reusable Patterns and Processes, and Performance Data

Several research approaches have been carried out for understanding processes and patterns. But this kind of data is not well integrated into existing BPMS, if we consider how BPMS can utilize these data to recommend the process design, modeling and optimization in a (semi)automatic manner. Furthermore, currently BPMS lack a connection to existing business processes that might be reused. This is not only due to a small handful of research efforts on mining process repositories [7] but also due to the lack of shared process repositories[5]. To support the (automatic) search and reuse of patterns and process models, it is expected to have a service-based repository for sharing business processes, either in the Web-scale or the individual organization level. Existing work has demonstrated the usefulness of documented best practices, detected patterns, and mining results, such as patterns used for design and modeling [6] and mining results used for recommending processes [10]. When we are able to manage reusable patterns and processes, then these works can be combined with other techniques, such as similarity analysis of processes [2], to provide powerful mechanisms to the design of new processes.

With respect to performance data, currently most BPMS support only a few metrics of IT performance, such as failure, availability, and response time, collected from the monitoring of the execution of processes. Some support the optimization of the process at runtime based on these performance metrics. However, historical performance metrics are not well integrated into BPMS for supporting the design, modeling, and runtime adaptation. To date, many performance analysis works have been done but we lack a standard way to link and manage performance data throughout the lifecycle of business processes. We should consider mining, process analysis, and performance analysis results to be associated with different levels of abstraction of business processes, such as individual activities and workflow regions, to provide a unified view on the performance in order to support the process refinement and optimization at different levels. In addition, as the business performance is measured through KPIs, it is interesting to establish the correlation between IT performance and business performance metrics; this is not well researched and understood. The performance data is also strongly linked to patterns and process models, and process repositories, and thus they should be managed and provided together.

3 Unified Data Management for BPM

Given a requirement, a business process is designed, modeled, executed, monitored and optimized. Although various types of data related to the process alone might be provided by different tools, the current situation is that we lack a mechanism to link all kinds of data inherently in the lifecycle of business processes. From our analysis of integrating data for BPM, we propose two main points for a unified data management system that should be integrated into BPMS:

- a unified, scalable and flexible model for integrating diverse types of data required by BPM.

[4] `http://aws.amazon.com/`

[5] For example, the Process Wiki (`http://wiki.process.io`) is a place where we can find a few business processes, while in the myExperiment (`http://www.myexperiment.org`) hundreds of scientific workflows are shared.

- techniques for managing the integrated data for business processes during their evolution.

Stimulated by our work in Web services evolution management [13], we devise a conceptual model of integrated data for BPM in Figure 1. In this model, we have different types of data, described under different specifications and linked through a meta-model. The instance data belonging to each type will be linked as an external source, such as modeling process description, process execution description, performance data, documented best practices, detected patterns, service capability registry, and human capability registry, thus allowing different specifications and diverse types of data to be included. This meta-model can be built based on XML in which a type of data is represented by a concept describing the type of data, the schema location, and the source of instance data. The collected data is then managed over the time by incorporating temporal aspects into the acquisition, management, and retrieval of the data. Furthermore, social aspects, such as teams and social networks, can be associated with particular types of data which are understood, analyzed, created and manipulated by a team of people. Currently, we are focusing on integrating modeling models [12], performance data and detected patterns [14, 4], software and human service registry [13], and human-provided services [9] with a focus on self-adaptive design, execution and optimization of SOA-based business processes.

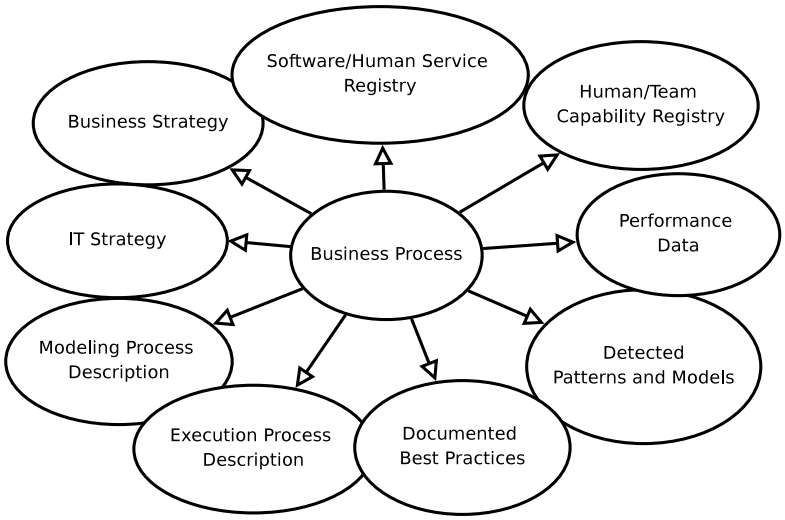


Figure 1: Unified model for integrating different types of data associated with business processes

4 Conclusion

In this paper we outline the current state of integrating data for business process management (BPM). As we have identified, since business processes increasingly rely on Web-scale resources, such as software services deployed under SaaS and human-provided services, there will be a need to integrate many types of data, to analyze and correlate these types of data, and to make them available in all phases of the lifecycle of business processes, in order to support the efficient design, adaptation and self-management of business processes.

References

[1] Alistair P. Barros, Marlon Dumas, and Arthur H. M. ter Hofstede. Service interaction patterns. In Wil M. P. van der Aalst, Boualem Benatallah, Fabio Casati, and Francisco Curbera, editors, *Business Process Management*, volume 3649, pages 302–318, 2005.

[2] Remco M. Dijkman. Diagnosing differences between business process models. In Dumas et al. [3], pages 261–277.

[3] Marlon Dumas, Manfred Reichert, and Ming-Chien Shan, editors. *Business Process Management, 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings*, volume 5240 of *Lecture Notes in Computer Science*. Springer, 2008.

[4] Schahram Dustdar, Thomas Hoffmann, and Wil M. P. van der Aalst. Mining of ad-hoc business processes with teamlog. *Data Knowl. Eng.*, 55(2):129–158, 2005.

[5] Genady Grabarnik, Heiko Ludwig, and Larisa Shwartz:. Management of service process qos in a service provider - service supplier environment. In *E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007. CEC/EEE 2007. The 9th IEEE International Conference on*, pages 543–550, July 2007.

[6] Thomas Gschwind, Jana Koehler, and Janette Wong. Applying patterns during business process modeling. In Dumas et al. [3], pages 4–19.

[7] Zhilei Ma1, Branimir Wetzstein1, Darko Anicic, Stijn Heymans, and Frank Leymann. Semantic business process repository. *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2007)*, 251:92–100.

[8] Carlos Pedrinaci, Dave Lambert, Branimir Wetzstein, Tammo van Lessen, Luchesar Cekov, and Marin Dimitrov. Sentinel: a semantic business process monitoring tool. In *OBI '08: Proceedings of the first international workshop on Ontology-supported business intelligence*, pages 1–12, New York, NY, USA, 2008. ACM.

[9] Daniel Schall, Hong Linh Truong, and Schahram Dustdar. Unifying human and software services in web-scale collaborations. *IEEE Internet Computing*, 12(3):62–68, 2008.

[10] Helen Schonenberg, Barbara Weber, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Supporting flexible processes through recommendations based on history. In Dumas et al. [3], pages 51–66.

[11] P. A. Smart, H. Maddern, , and R. S. Maull. Understanding business process management: Implications for theory and practice. *British Journal of Management*, 2008.

[12] Huy Tran, Uwe Zdun, and Schahram Dustdar. View-based reverse engineering approach for enhancing model interoperability and reusability in process-driven soas. In Hong Mei, editor, *ICSR*, volume 5030 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2008.

[13] Martin Treiber, Hong Linh Truong, and Schahram Dustdar. Semf - service evolution management framework. In *SEAA*, pages 329–336. IEEE, 2008.

[14] Hong Linh Truong and Schahram Dustdar. Online interaction analysis framework for ad-hoc collaborative processes in soa-based environments. *T. Petri Nets and Other Models of Concurrency*, 2:260–277, 2009.

[15] Wil van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[16] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[17] Ping Zhang and Nicoleta Serban. Discovery, visualization and performance analysis of enterprise workflow. *Comput. Stat. Data Anal.*, 51(5):2670–2687, 2007.

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398