Facilitating Workflow Interoperation Using Artifact-Centric Hubs

Richard Hull[1,⋆], Nanjangud C. Narendra[2], and Anil Nigam[3]

[1] IBM T.J. Watson Research Center, USA
hull@us.ibm.com
[2] IBM India Research Lab, Bangalore, India
narendra@in.ibm.com
[3] IBM T.J. Watson Research Center, USA
anigam@us.ibm.com

Abstract. Enabling interoperation between workflows, and between web services, continues to be a fundamental challenge. This paper proposes a new approach to interoperation based on hubs that are designed using "business artifacts", a data-centric paradigm for workflow and business process specification. The artifact-centric interoperation hubs are focused primarily on *facilitating communication and business-level synchronization* between relatively autonomous stakeholders (and stakeholder organizations). Interoperation hubs provide a centralized, computerized rendezvous point, where stakeholders can read or write data of common interest and check the current status of an aggregate process, and from which they can receive notifications about events of interest. The paper describes the approach, including an extended example, access restrictions that can be placed on stakeholders, some preliminary theoretical results, and a discussion of work towards a prototype system that supports interoperation hubs.

Keywords: Business Artifact, Interoperation, Service, Workflow.

1 Introduction

Enabling interoperation between workflows, and between web services, continues to pose a fundamental challenge. Two traditional approaches to this challenge are orchestration and choreography [16]. Orchestration tackles interoperation by essentially creating an new application with a centralized set of goals to be achieved. The orchestrator is typically designed to fit with the various workflows or services that are to interoperate, thus limiting opportunities for re-use of the orchestration. Also, orchestrators become the primary controllers of the interoperation, and as a result reduce the autonomy of the different stakeholders (individuals and organizations) in acheiving their portions of the aggregate goal. On the other hand, choreography embraces the autonomy of the stakeholders, and attempts to enforce the achievement of aggregate goals by restricting how messages can be passed between the stakeholder workflows or services. A weakness of choreography, however, is that there is no single conceptual point or "rendezvous" where stakeholders can go to find current status and information about the aggregate process. This paper proposes a new approach to workflow and web service interoperation, that largely preserves the autonomy of participating stakeholders, and provides

[⋆] This author partially supported by NSF grants IIS-0415195, CNS-0613998, and IIS-0812578.

a single conceptual point where stakeholders can obtain current status and information about a process, and can receive notifications about status changes of interest. We call our approach *interoperation hubs*.

The interoperation hubs proposed here focus primarily on *facilitating communication and business-level synchronization* between relatively autonomous stakeholders. The conceptual model used by the hubs is based on "business artifacts" [14,5,11], a data-centric paradigm for workflow and business process specification. An interoperation hub can be viewed as a stylized "whiteboard" for holding information relevant to the stakeholder community as they participate in a consensus-based aggregate process. The whiteboard is structured to ensure that certain constraints are satisfied about information access, information update, and task sequencing. The hub is primarily passive and re-active, allowing the stakeholder workflows to post new information of interest to the stakeholder community. The hub is pro-active in only one regard: stakeholders can subscribe for notification when certain steps of the aggregate process have occurred.

The interoperation hubs used here are fundamentally different from conventional orchestrators. The difference stems from the fact that *business artifacts*, or simply "artifacts", unlike BPEL, provide a holistic view of process and data. Artifacts are business-relevant objects that are created, evolve, and (typically) archived as they pass through the workflow. An artifact type includes both an *information model* (or "data schema"), that can hold data about the business objects during their lifetime in the workflow, and a *lifecycle model* (or "lifecycle schema"), that describes the possible ways and timings that tasks (a.k.a. services) can be invoked to manipulate these objects. A prototypical example of a business artifact type is "air courier package delivery," whose information model would include slots for data such as ID of the package, sender, recipient, arrival times at different points along the way, time delivered, and billing information, and whose lifecycle model would include the different ways that the package could be delivered and paid for.

In the context of individual workflows, experiences reported [6,5,7] by the business artifacts team at IBM Research show that an artifact-based perspective helps in improving stakeholder understanding of the workflows, and often leads to new insights. These experiences suggest that an artifact-based interoperation hub will be of significant business value to the many people in the stakeholder organizations. More specifically, it suggests that the business managers, business architects, and IT infrastructure specialists will be able to adapt their workflows, including both manual and automated portions, to take advantage of the interoperation hub, to draw upon the information that can be stored there, to write appropriate information there, and to help guide how the artifacts move through their lifecycles.

In addition to presenting an extended example to illustrate our approach, the paper includes formal definitions for three kinds of access restrictions. "Windows" provide a mechanism to restrict which artifacts a stakeholder can see; "views" provide a mechanism to restrict what parts of an artifact a stakeholder can see; and a varation of "Create-Read-Update-Delete (CRUD)" specifications is used to restrict the ways that stakeholders can read and modify artifacts. The paper also studies the question of *persistent visibility* of artifacts. An interoperation hub supports persistent visibility if, for each stakeholder p and artifact a, if a becomes visible (based on the window restrictions) to p at some point, then a remains visible to p for the remainder of its evolution through the workflow. In the general case testing this property is undecidable, but we show that it is decidable for a natural class of window specifications.

The approach of interoperation hubs can be used to facilitate interoperation between different enterprises or organizations. Indeed, companies such as PayPal or Salesforce.com can be viewed as providing massively scaled interoperation hubs, that facilitate interoperation between largely autonomous stakeholders. Conference submission management sites such as ConfTool or EasyChair also provide application-specific interoperation hubs. The framework and theoretical development of this paper provides a formal basis for analyzing application-specific interoperation hubs, such as those just mentioned.

Section 2 introduces an example of an interoperation hub along with its business artifacts. This example is used throughout the paper to illustrate different concepts related to interoperation hubs. Section 3 presents a more formal description of the framework for artifact-centric interoperation hubs. Section 4 incorporates various notions related to access control, and presents some preliminary theoretical results about the framework. Section 5 describes work towards a prototype implementation of interoperation hubs. Section 6 describes related work, and Section 7 offers brief conclusions.

2 Representative Example

This section presents an illustration of an artifact-based interoperation hub, which will be used through the rest of the paper. It is based on employee hiring an an enterprise.

2.1 Example Overview

Fig. 1 shows the six primary kinds of stakeholders and stakeholder organizations whose interoperation around hiring will be supported, viz., *Candidates, Human Resources Organization, Hiring Organizations, Evaluators, Travel Provider, Reimbursement*. There could be several hiring organizations, each with its own worklows for managing the recruitment process. We assume that the enterprise has a single HR organization responsible for recruitment purposes.

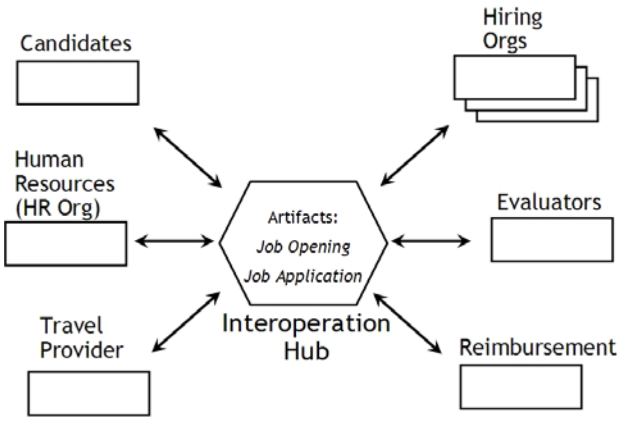


Fig. 1. Six types of stakeholder organizations using the Hiring Interoperation Hub

Participants can interact with the hub in several ways. For instance, a candidate may choose to interact directly with the designated *Travel Provider*, perhaps through their web-site, and create her itinerary. The hub can record the authorization for travel (perhaps from HR or perhaps from the Hiring Organization). The Travel Provider can the access the hub for the travel authorization, and place a link to the itinerary. This enables the Hiring Organization and the Evaluators to access the itinerary when preparing for interviews. Finally, the Reimbursement organization can access the airline and hotel invoices when processing the travel reimbursement request from the candidate. These interactions illustrate how an interoperation hub can facilitate information transfer between participant organizations, while giving them considerable autonomy and latitude with regards to how and when they provide the information or accomplish tasks.

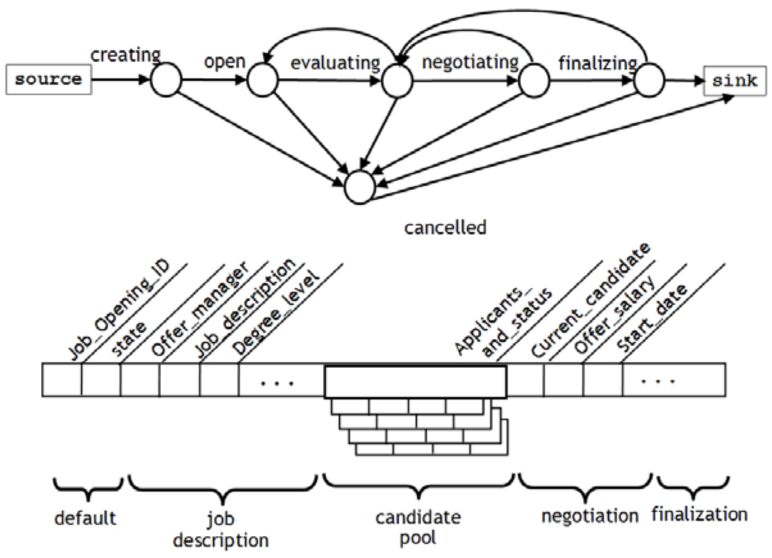


Fig. 2. `Job_Opening` artifact type used in the Hiring interoperation hub

Our focus here is on the two primary artifact types of the Hiring interoperation hub - `Job_Opening` and `Job_Application` (Figs. 2 and 3, respectively). Each artifact type contains two primary components - the *information model* (or "data schema"), that uses a variant of the nested relation model, and the *lifecycle model* (or "lifecycle schema"), that uses a variant of finite state machines.

The `Job_Opening` artifact type was designed on the premise that the enterprise would negotiate with just one candidate at a time; a richer information model could be used if simultaneous negotiations with multiple candidates are to be supported. Fig. 2 also depicts the six states that a `Job_Opening` artifact can be in, including the inter-state transitions. When the artifact moves to `open` state, summary information about the candidates that apply can be stored into the candidate pool portion of the information model. (However, the bulk of the candidate information will be stored in the corresponding `Job_Application` artifact instance.) In the `evaluating` state, one or more of

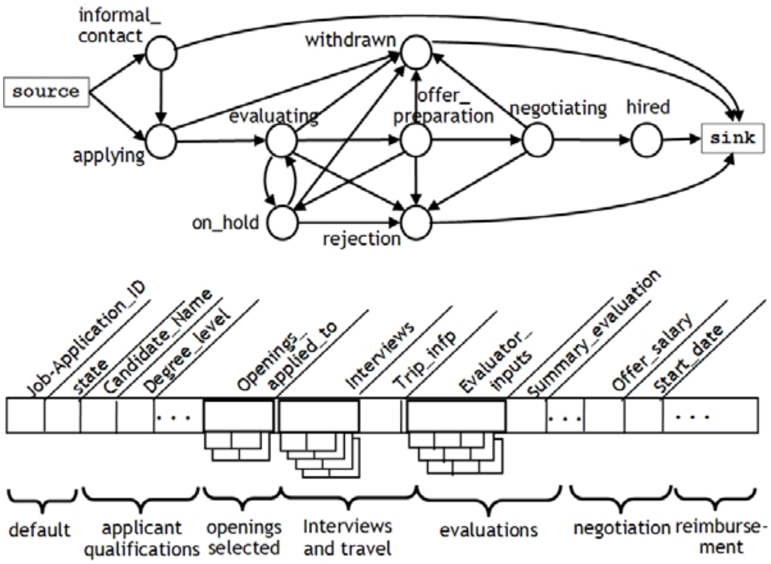


Fig. 3. `Job_Application` artifact type used in the Hiring interoperation hub

the applicants might start through the evaluation process. In the `negotiating` state, one candidate has been shortlisted for selection, and undergoes salary negotiations with the HR organization. At any point of time, HR or the Hiring Organization can cancel the job opening by moving the artifact instance to `cancelled` state.

The information model for `Job_Opening` consists of the following attribute clusters - an ID, job details (job description, offering manager, etc.), candidate pool (summary of each applicant being considered), negotiation details (current candidate, offered salary, start date), finalization (salary, start date, hiring manager, etc.). Similarly, the lifecycle specification of `Job_Opening` is specified as the states - `creating, open, evaluating,negotiating, finalizing, cancelled` - and transitions between them. Each stakeholder is responsible for adding appropriate information to the `Job_Opening` in some state and might also move it to the next state.

The `Job_Application` artifact type is intended to track job candidates from when the enterprise first thinks of someone as a potential recruit, through the formal application process, to the point of hiring, rejection, or the candidate withdrawing her application. As noted previously, the interoperation hub should be viewed as an electronic repository of selected information relevant to the stakeholders involved with a given applicant; it is not required that all information exchanged between the stakeholders be formally recorded into the hub. For example, the candidate and the travel provider might interact directly, and then have the travel provider record the planned itinerary into the interoperation hub. This illustrates another aspect of the flexibility in communication and business-level synchronization enabled with using interoperation hubs.

As per Fig. 3, a candidate can either apply formally (`applying` state) or come into informal contact with the Hiring Organization (`informal_contact` state). The rest of the state machine is self-explanatory.

2.2 Design Considerations

The design of the artifact types in an interoperation hub may involve both top-down and bottom-up thinking and analysis. For the top-down, the designer thinks in terms of the artifact types needed to support the common process, whereas for the bottom-up, the designer considers the artifact types that are explicit or implicit in the stakeholder organizations. We first describe the design of artifact types in general, and then consider the top-down and bottom-up approaches for interoperation hubs in particular.

The basic guideline for designing an artifact type is that it should provide a tangible means for tracking a specific business goal. The business goal provides clues as to the information that ought to be maintained by the business artifact, i.e., its information model. Next considering the operations that fill in this information incrementally, the goal provides the basis for the applicable states and transitions, i.e., the lifecycle model. This approach can used, in a top-down manner, either as a starting point or during refinement in the design of artifact types of an interoperation hub.

We now illustrate how the bottom-up approach can help in the design of hub artifacts. For each stakeholder, we briefly review their goals and the artifacts that can track these goals. It should be noted that the stakeholder workflows do not need themselves to be artifact-centric; the determination of artifacts that capture their goals and behaviors is a useful thought exercise that contributes to the design of the hub. *Candidates* strive to obtain suitable employment at the best terms; they will think primarily in terms of `Job_Application` artifact type of the Hiring hub. *HR* employees are focused on ensuring that the hiring policies are followed; towards this end their artifacts will also be based on `Job_Opening` and `Job_Application`. *Hiring Organizations* are primarily interested in recruiting highly qualified applicants who will be productive contributors in their organization; their artifacts would be Prospects (or "Leads"), which in some cases lead to the creation of `Job_Application`. *Evaluators* focus on providing effective input on the capabilities of Prospects/Candidates. The *Travel Provider* works to provide a friendly and effective service to help candidates finalize their travel arrangements, and as such is focused on managing the Itinerary. *Reimbursement* strives to reimburse applicants for travel expenses incurred, accurately and in a timely manner, and their primary artifact is an Expense Report.

Considering the goals that need to be tracked, the `Job_Application` artifact type addresses the needs of *Hiring Organization* and Candidates alike. It also incorporates information relevant to *Evaluators*, the *Travel Provider*, and *Reimbursement*. Additionally, the `Job_Opening` artifact can serve to track a union of the goals of *HR* and the *Hiring Organizations* with regards to Candidates and Prospects, respectively. A final observation is that both artifact types of the Hiring hub is of interest to more than one stakeholder; this makes the case for an interoperation hub that houses the aggregate process. As we will see later, each stakeholder will have differing abilities to view and modify the information in these artifacts.

3 A Framework for Artifact-Centric Interoperation Hubs

This section presents a succinct description of the framework for interoperation hubs, and illustrates the framework in terms of the Hiring example. Due to space limitations, only the most essential definitions are given in detail. For this formalism, we use the terms 'data schema' and 'lifecycle schema' rather than 'information model' and 'lifecycle model', to be more consistent with the database and workflow literature.

3.1 Nested Data and Artifact Types

The information model for the artifacts in interoperation hubs is based on nested data types, based on a nested relation model. These are built up using scalars and four types of identifier, namely `artifact_ID`, `participant_ID`, `stakeholder_org_ID`, and `state_name` (described below), and the record (formed with attribute names) and set constructs. We permit the use of an undefined value, denoted as $\perp$, for any type. We consider only nested data types that are in *Partitioned Normal Form* (PNF) [17,2], that is, so that for each set of (nested) tuples, the set of non-nested attributes forms a key for the overall set of tuples.

An *artifact data schema* is a nested record type of the form

$$D = \langle \textbf{ID} : \texttt{artifact\_ID},\ \textbf{state} : \texttt{state\_name},\ A_1 : T_1, \ldots, A_n : T_n \rangle$$

(where the T_j's range over nested data types). Intuitively, in an instance of an artifact data schema, the first field will hold a unique ID for the artifact being represented, and the second field will hold the state in the lifecycle that the artifact is currently in. We use the term *snapshot* to refer to instances of an artifact data schema; this is to reflect the intuition that artifacts persist as they evolve through a workflow, and pass through a sequence of "snapshots" over this time period. We informally use the term "*artifact instance*" to refer to the persisting object that underlies a sequence of artifact snapshots all having the same artifact ID.

An *artifact lifecyle schema* is a pair (S, E) where

1. S is a finite set of *states*, which includes the designated states **source** and **sink**.
2. E is a set of directed edges (ordered pairs over S), such that there are no in-edges into **source** and there are no out-edges from **sink**.

Intuitively, on a move from **source** to another state an artifact instance is created, and on a move from a state into **sink** an artifact instance is archived and effectively taken out of further evolution or participation in the workflow.

An *artifact type* is a triple $\mathcal{R} = (R, D, L)$ where R is the name of the type (a character string), D is an artifact data schema and L is an artifact lifecycle schema. Suppose that $L = (S, E)$. A *snapshot* of $\mathcal{R}$ is a snapshot of D such that the **ID** and **state** attributes are defined, and the **state** attribute is an element of $S - \{\textbf{source}\}$.

3.2 Artifact Schemas and Hub Schemas

An *artifact schema* is a collection $\mathcal{S} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ of artifact types that have pairwise distinct names $R_1, \ldots, R_n$. If *PID* is a set of participant IDs and *OID* a set of stakeholder organization IDs, then a *snapshot* of $\mathcal{S}$ over *PID*, *OID* is a function $I : \{R_1, \ldots, R_n\} \rightarrow$ sets of artifact snapshots that use participant (organization) IDs from *PID* (*OID*), where $I[R_j]$ is a set of artifact snapshots of type $\mathcal{R}_j$, $j \in [1..n]$; there are no pairs s_1, s_2 of distinct snapshots in $\cup_j I(R_j)$ with $s_1.\textbf{ID} = s_2.\textbf{ID}$; and for each artifact ID g occurring in any artifact snapshot of I, there is a snapshot s occurring in I with $s.\textbf{ID} = g$.

Job_Opening_ID	**State**	**Applicants_and_Date**		...
J312	negotiating	A567	4/15/09	
		C123	3/10/09	
		B647	4/10/09	

(a) Partial snapshot of a `Job_Opening` artifact

Job_App_ID	**State**	**Name**	...
A567	evaluating	Alice	
C123	evaluating	Carl	
B647	offer_preparation	Bob	

(b) Partial snapshots of `Job_Application` artifacts

Fig. 4. Partial snapshot of the artifact schema from the Hiring example

Example 1. An example snapshot from the Hiring example is depicted in Fig. 4 (only some of the attributes are shown). Part (a) depicts a snapshot of a single `Job_Opening` artifact with ID of J312. This artifact instance is in the `negotiating` state, indicating that one candidate has been short-listed. Part (b) shows a portion of the snapshots of three artifact instances of type `Job_Application`, all of whom applied for opening J312. One of them, corresonding to Bob, is in the `offer_preparation` state. □

A *hub schema* is a pair $\mathcal{H} = (\mathcal{S}, \mathbf{Org})$ where $\mathcal{S}$ is an artifact schema and **Org** is a finite set of stakeholder organization types. In the Hiring example, **Org** has six types.

Let $\mathcal{H} = (\mathcal{S}, \mathbf{Org})$ be a hub schema. A *snapshot* of $\mathcal{H}$ is a 5-tuple

$$H = (I^{art}, OID, PID, I^{org}, I^{part})$$

where

1. I^{art} is a snapshot of $\mathcal{S}$ over OID, PID.
2. *OID* is a finite set of `organization IDs`.
3. *PID* is a finite set of `participant IDs`.
4. $I^{org} : OID \rightarrow$ **Org** is the organization to organization type mapping.
5. $I^{part} : PID (\rightarrow 2^O - \{\emptyset\})$ is the participant to role mapping.

Here, each stakeholder organization in *OID* is associated with exactly one stakeholder organization type in **Org**, and each participant in *PID* is a member of at least one, but possibly more than one, stakeholder organization in *OID*.

Example 2. Returning to our running example, two stakeholder organizations of the same type could be Software_Group and Research_Division, both acting as Hiring Organizations. As an example of a participant belonging to more than one stakeholder organization, an employee might be a member of Research_Division and involved with overseeing the recruiting for a staff researcher position, and also serve as an Evaluator for a candidate being considered by Software_Group. □

We now consider how the contents of an interoperation hub can evolve over time. Let $\mathcal{H} = (\mathcal{S}, \mathbf{Org})$ be a hub schema, and let H, H' be two snapshots of $\mathcal{H}$. Then H *transitions* to H', denoted $H \rightarrow_{\mathcal{H}} H'$, if one of the following holds

1. **(New artifact instance:)** H' is the result of adding a single, new artifact shapshot s to H, and the state of that snapshot is one state away from the **source** node of the state machine of the artifact schema of s.
2. **(Update to artifact instance:)** H' is the result of replacing a single snapshot s of H by a new snapshot s' having the same type, where $s.$**ID** $= s'.$**ID** and $s.$**state** $= s'.$**state**, and for at least one top-level attribute A, $s.A \neq s'.A$.

3. **(Change state of artifact instance:)** H' is the result of replacing a single snapshot s of H by a new snapshot s' having the same artifact type $\mathcal{A} = (D, L)$, where s.**ID** $= s'$.**ID**, (s.**state**, $s'.state$) is a transition in L, and for each other attribute A, $s'.A = s.A$.
4. **(Modify participants or stakeholder organizations:)** H' is the result of adding or dropping an element to OID or PID, or making a change to the function I^{org} that impacts a single organization, or making a change to the function I^{part} that impacts a single participant.

3.3 Adding a Condition Language

We shall use a logic-based expression language for nested data types, that can be used to express conditions and queries on snapshots of artifact and schema snapshots. The language is modeled after the calculus defined in [1], and we provide here only the most salient details. Variables are typed using the nested complex types. Terms include $\tau.A$ for record type term τ and attribute name A. Constructors are provided to create record- and set-typed terms. Atomic formulas include $R(\tau)$ for artifact schema name R, $\tau = \tau'$ for scalar or ID types (but not set types), $\tau \in \tau'$ where the type of τ' is set of the type of τ. It also includes atomic formulas of the form $\tau \in \tau'$ where τ has type `participant_ID` and τ' has type `organization_ID`. Query expressions are created in the manner of relational calculus queries.

4 Views and Access Rights

An important component of the interoperation hub vision is that typically, stakeholders will not be able to see entire artifacts, nor will they be able to make arbitrary updates to them. This section introduces the notion of *views* of artifact schemas and snapshots which restrict what participants from a given organization type can see, *windows* into the set of artifacts that a given participant can see, and also access rights for making updates against them based on a generalization of "CRUD" restrictions. These notions embody an important aspect of the utility of interoperation hubs in facilitating communication and business-level synchronization between organizations, because they provide mechanisms for ensuring that information and events that should be kept private are indeed being kept private.

The section also develops a family of simple theoretical results, including a decidability result concerning whether an artifact, once visible to a participant, remains visible to the participant for the rest of its lifecycle.

4.1 Views

We begin with an example of the views presented to one kind of stakeholder.

Example 3. Figures 5 and 6 show the views of `Job_Opening` and `Job_Application`, respectively, that are made available to Candidates in the Hiring example. In these views, some of the attributes of the data schema are grayed out, because the view prevents a candidate from seeing those attributes. In terms of the lifecycle, some states are collapsed or "condensed" together. (These are shown as solid disks.) For example, in the view

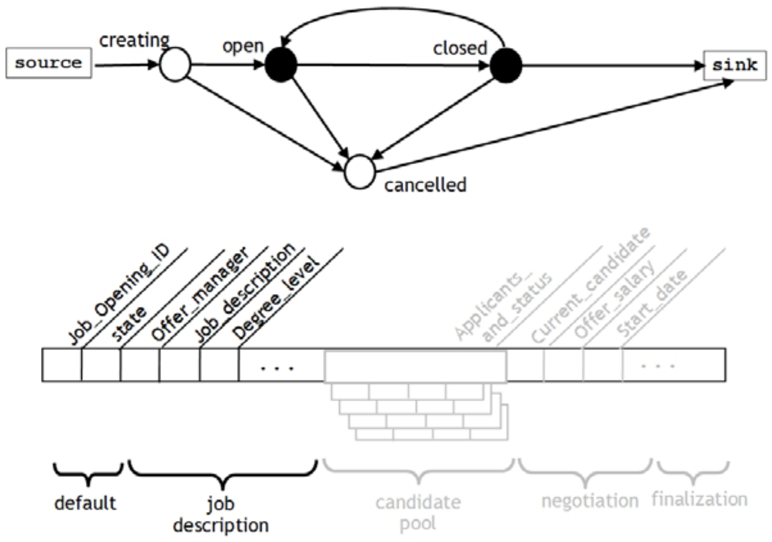


Fig. 5. The view of the `Job_Opening` artifact type that would be visible to Candidates

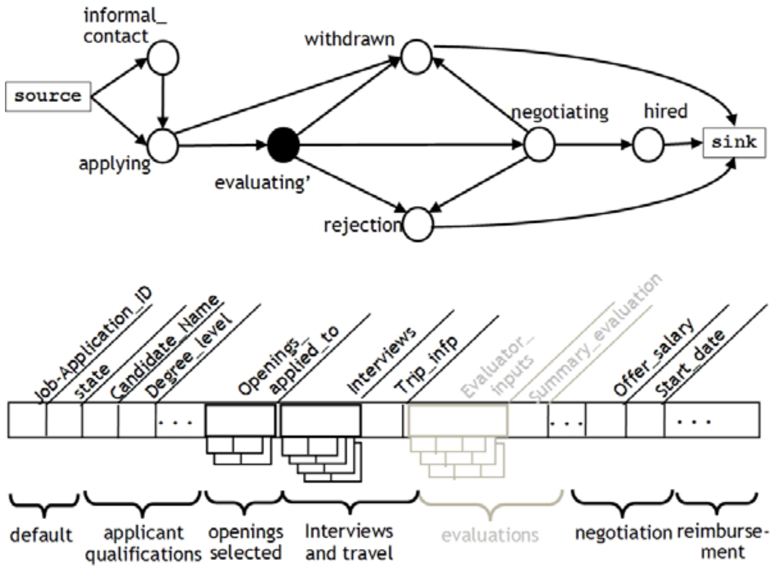


Fig. 6. The view of the `Job_Application` artifact type that would be visible to Candidates

of `Job_Application`, the enterprise will typically want to hide from the candidate whether it is in the `evaluating`, `on_hold`, or `preparing_offer` state. Similarly, as shown in the view of `Job_Opening` several states are collapsed into the two states `open` and `closed`. (The specific mapping of states to `open` and `closed` would depend on the business policy to be followed.) □

With regards to artifact types, the notion of view includes two components: a restriction on the attributes that can be seen, and a restriction on the set of states that can be seen. The first case will be achieved using projection, and the second by using node condensation.

Let $\mathcal{R} = (R, D, L)$ be an artifact type, where $D = \langle$**ID** : `artifact_ID`, **state** : `state_name`, $A_1 : T_1, \ldots, A_n : T_n \rangle$. A *projection mapping* over D is an expression of the form π_J where J is a subset of $A_1, \ldots, A_n$. Projection mappings operate at both the data schema level and the snapshot level in the natural manner. A *condensation mapping* over a lifecycle schema $L = (S, E)$ is an expression γ_f where f is a surjective function $f : S \rightarrow S'$, where S' is a set of state names, such that

1. **source** , **sink** $\in S'$.
2. $\gamma($**source**$) =$ **source** and $\gamma($**sink**$) =$ **sink**

(Note that multiple states of S might map into **source** or **sink**.) A condensation mapping works on the meta-data of a lifecycle schema; specifically, in the above case $\gamma_f(L) = (S', E')$ where $E' = \{(\gamma(\sigma_1), \gamma(\sigma_2) \mid (\sigma_1, \sigma_2) \in E\}$.

A *view* on $\mathcal{R} = (R, D, L)$ is an ordered pair $\nu = (\pi_J, \gamma_f)$ where π_J is a projection mapping over D and γ_f is a condensation mapping over L. The result of applying ν to a snapshot s of $\mathcal{R}$ is defined in the natural manner. For an artifact schema $\mathcal{S} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$, with artifact type names $R_1, \ldots, R_n$, a *view mapping* of $\mathcal{S}$ is a function ν with domain $\{R_1, \ldots, R_n\}$ such that $\nu[R_j]$ is a view on $\mathcal{R}_j$ for $j \in [1..n]$. We use $\nu(\mathcal{S})$ to denote the schema $\{\nu[R_1](\mathcal{R}_1), \ldots, \nu[R_n](\mathcal{R}_n)\}$, and for a snapshot I of $\mathcal{S}$, $\nu[I]$ is a snapshot of $\nu(\mathcal{S})$ defined in the natural manner. Finally suppose now that $\mathcal{H} = (\mathcal{S}, \textbf{Org})$ is a hub schema. A *view mapping* for $\mathcal{H}$ is a function ν with domain **Org**, such that for each stakeholder organization type O in **Org**, $\nu[O]$ is a view mapping of $\mathcal{S}$.

Example 4. The projection mapping of `Job_Opening` artifact for an Evaluator would include all of the attributes depicted in Fig. 5, except for `Offer_salary`. The condensation mapping, however, would include all states in the `Job_Opening` artifact. □

If ν is a view mapping, and there is a participant p who is a member of two or more organizations of different types, then $\nu[R](p)$ can be defined using a union on the projections and a variant of the cross-product construction for the condensations. The details are omitted due to space limitations.

Given a view mapping and an organization type O, we ask: when does it make sense for participants in an organization of type O to be able to request a transition in a lifecycle in their view? Let $\mathcal{H} = (\mathcal{S}, \textbf{Org})$ by an interoperation hub, ν be a view mapping, and $\mathcal{R} = (R, D, (S, E))$ be an artifact type in $\mathcal{S}$. An edge $e = (\sigma_1, \sigma_2) \in E$ is *eligible* in $\mathcal{H}$ if for some $O \in \textbf{Org}$ with $\nu[O](R) = (\pi_J, \gamma_f)$ we have: for each state $\sigma'_1 \in f^{-1}(f(\sigma_1))$ there is exactly one state $\sigma'_2 \in f^{-1}(f(\sigma_2))$ such that $(\sigma'_1, \sigma'_2) \in E$ Intuitively, this means that if a participant p in an organization of type O requests a transition in p's view from $f^{-1}(\sigma_1)$ to $f^{-1}(\sigma_2)$, then there is no ambiguity with regards to which transition in the base state machine (S, E) should be taken.

Although not done here due to space limitations, it is straightforward to characterize, given an interoperation hub $\mathcal{H}$ and view mapping ν, the full set of transitions $\rightarrow_{\mathcal{H},\nu}$ between snapshots of $\mathcal{H}$ that can be achieved by participants working through their views.

4.2 Windows

The notion of "window mapping" is used to restrict which artifact instances a participant can see. Recall the condition language from Section 3. Suppose that $\mathcal{S} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ is an artifact schema with artifact type names $R_1, \ldots, R_n$. For $j \in [1..n]$, a query $Q_{\varphi(x,y)}$ is a *window mapping* for $\mathcal{R}_j$ using x for participant IDs and y for artifact IDs if

1. x has type `participant_ID`,
2. y has type `artifact_ID`, and
3. φ has the form $\exists z(R_j(z) \wedge z.\mathbf{ID} = y \wedge \psi(x, z))$ for some formula ψ.

When a window mapping $Q_{\varphi(x,y)}$ for $\mathcal{R}_j$ is applied to a snapshot I of $\mathcal{S}$, the result is $Q_{\varphi(x,y)}(I) = \{(p, g) \mid I \models \varphi[x/p, y/g]\}$. Note that in each element of the answer, the second coordinate will be the ID of an artifact snapshot in $I[R_j]$. Analogous to view mappings, a window mapping ω for an interoperation hub $\mathcal{H} = (\mathcal{S}, \mathbf{Org})$ is a function that maps a pair O, R (for $O \in \mathbf{Org}$ and R the name of some $\mathcal{R}$ in $\mathcal{S}$) to a window mapping $\omega[O, R]$ for $\mathcal{R}$.

Example 5. In the running example, the window query for Hiring Organizations might permit them to see only `Job_Applications` that are targeted at `Job_Openings` sponsored by that Hiring Organization. For the snapshot of Fig. 4, Software_Group would see all three candidates, but Research would see none of them. $\square$

In many cases, a window mapping will focus on whether a certain pattern of values is found in the current snapshot. For example, an evaluator might be permitted to "see" all `Job_Application` artifact instances for which he is named in the *evaluators* attribute. Happily, the class of such window mappings, which have no negation and only existential quantifiers, correspond closely to the conjunctive queries with union in the relational model, about which many properties are known.

To illustrate, we briefly study the question of whether one window mapping ω is less restrictive than another one ω'. Let $\mathcal{H} = (\mathcal{S}, \mathbf{Org})$ be an interoperation hub. We say that ω *dominates* ω', denoted $\omega' \preceq \omega$, if for each instance I of $\mathcal{H}$ and each artifact type $\mathcal{R} = (R, D, L)$ of $\mathcal{H}$, $\omega'[O, R](I) \subseteq \omega[O, R](I)$. Using the fact that our nested types are in Partitioned Normal Form, the correspondance with conjunctive queries with union, and results from [18] we obtain the following.

Proposition 6. Let $\mathcal{H}$ be an interoperation hub, and assume that ω and ω' are window mappings based on queries that have no negation and only existential quantifiers. Then it is decidable whether $\omega' \preceq \omega$, and this decision problem is NP-complete.

4.3 Access Rights and CRUDAE

Windows and views give a first-tier, rather coarse-grained mechanism for specifying the access rights of participants to the contents of an interoperation hub. Following in the spirit of [19,8,10], we now introduce a finer-grained mechanism, that is based on providing “Create-Read-Update-Delete-Append (CRUDA)” and “Execute” permissions to a participant p, depending on what type of stakeholder organization(s) p is in, and what state an artifact instance is in. (More precisely, this is based on the state in p’s view of the artifact instance.)

Suppose that $\mathcal{R} = (R, D, L)$ is an artifact schema, where $D = \langle$**ID** : `artifact_ID`, **state** : `state_name`, $A_1 : T_1, \ldots, A_n : T_n \rangle$ and $L = (S, E)$. A *simple CRUDAE specification* for $\mathcal{R}$ is a mapping α with domain $\{A_1, \ldots, A_n\} \cup \{\text{'E'}\}$ and where

- $\alpha : \{A_1, \ldots, A_n\} \rightarrow 2^{\{\text{'C'},\text{'R'},\text{'U'},\text{'D'}, \text{'A'}\}}$
- $\alpha(\text{'E'}) \subseteq E$ (i.e., the set of edges in L)

Intuitively, if ’C’ $\in \alpha(A_j)$, this indicates that under α, a participant can “create” a value for A_j(e.g.,, provide a value to a previously undefined attribute) and similarly for cases of ’R’ $\in \alpha(A_j)$, ’U’ $\in \alpha(A_j)$, ’D’ $\in \alpha(A_j)$ and ’A’ $\in \alpha(A_j)$; and α(E) indicates the set of edges that the participant can request a transition along.

In general, we associate a simple CRUDAE specification to each state of an artifact lifecycle, reflecting the intuition that access rights typically change based on state. A *CRUDAE specification* for $\mathcal{R} = (R, D, (S, E))$ is a mapping β with domain S, such that $\beta[\sigma]$ is a simple CRUDAE specification for $\mathcal{R}$ for each state $\sigma \in S$. Intuitively, for state $\sigma \in S$, $\beta[\sigma]$ is intended to indicate the access rights that a participant will have when the artifact instance is in state σ. Suppose that $\sigma \in S$, and consider $\beta[\sigma]$(E). It is natural to assume that each edge $e \in \beta[\sigma]$(E) has σ as source.

Suppose now that $\mathcal{H} = (\mathcal{S}, \textbf{Org})$ is a hub schema, where $\mathcal{S} = (\mathcal{R}_1, \ldots, \mathcal{R}_n)$ with artifact type names $(R_1, \ldots, R_n)$. Suppose further that ν is a view mapping for $\mathcal{H}$. A *CRUDAE specification* for the pair $(\mathcal{H}, \nu)$ is a mapping δ with domain $\textbf{Org} \times \{R_1, \ldots, R_n\}$, such that

1. $\delta[O, R_j]$ is a CRUDA specification for $\nu[O](\mathcal{R}_j)$, for each $O \in \textbf{Org}$ and $j \in [1..n]$.
2. $\delta[O, R_j](E) \subseteq \{e \mid e$ is an eligible edge in the lifecycle of $\nu[O](\mathcal{R}_j)$

To understand this intuitively, think of a stakeholder organization type $O \in \textbf{Org}$. Recall that a participant p in an organization o of type O cannot “see” all of $\mathcal{S}$, but rather can “see” only $\nu[O]$. Furthermore, $\delta[O, R_j]$ will indicate, for each state in the lifecycle of $\nu[O](\mathcal{R}_j)$, which attributes of $\nu[O](\mathcal{R}_j)$ can be created, read, updated, or deleted by p, and also which transitions in the lifecycle of $\nu[O](\mathcal{R}_j)$ can be invoked by p.

Example 7. We recall that a candidate can view at most one `Job_Application` artifact instance, namely,, the one that the candidate created. For this instance, the candidate has Update permission only for attributes such as Degree_level, vita, experience, and only Read permission for the other attributes depicted in Fig. 6. The candidate will have execute permission to bring about a move his `Job_Application` artifact from `evaluating'` to `withdrawn` state, by withdrawing his/her application from consideration for the job opening. □

Finally, an *extended hub schema* is a tuple $\mathcal{H} = (\mathcal{S}, \mathbf{Org}, \omega, \nu, \delta)$ where

1. $(\mathcal{S}, \mathbf{Org})$ is a hub schema.
2. ω is a window mapping for $(\mathcal{S}, \mathbf{Org})$
3. ν is a view mapping for $(\mathcal{S}, \mathbf{Org})$
4. δ is a CRUDAE specification for $(\mathcal{S}, \mathbf{Org})$

The notion of transitions between snapshots of a hub schema can be genealized to extended hub schemas in the natural manner, taking into account the restrictions on participants, based on the view they can "see", the artifacts accessible through the window mapping, and the CRUDAE mapping. For an extended hub schema $\mathcal{H}$, this relation is denoted by $\rightarrow_H$.

4.4 Persistence of Visibility

We conclude the section by studying the question: Given an extended interoperation hub $\mathcal{H} = (\mathcal{S}, \mathbf{Org}, \omega, \nu, \delta)$, and a participant p and artifact instance a, is it possible that p can "see" a at some point but not "see" a at a later point.

More precisely, let $\boldsymbol{I} = I_0 \rightarrow_{\mathcal{H}} I_1 \rightarrow_{\mathcal{H}} \ldots \rightarrow_{\mathcal{H}} I_n$ be a sequence of snapshots of $\mathcal{H}$ satisyfing the $\rightarrow_{\mathcal{H}}$ relation as indicated, where I_0 is the empty snapshot, and in which there are no transitions involving changes to the particpants or the organizations. Suppose that for some participant ID p, artifact ID g, artifact type $\mathcal{R}$ with name R, and index j, we have $(p, g) \in \omega[O, R](I_j)$, and j is the first index with this property. Then g has *persistent visibility* for p in $\boldsymbol{I}$ if $(p, g) \in \omega[O, R](I_k)$ for each $k \in [j + 1, n]$. The sequence $\boldsymbol{I}$ has *persistent visibility* if each artifact ID occurring in $\boldsymbol{I}$ has persistent visibility for each participant occurring in $\boldsymbol{I}$. $\mathcal{H}$ has *persistent visibility* if each such sequence $\boldsymbol{I}$ has persistent visibility.

In some cases it may be natural to not have persistent visibility. For example, a candidate p may see a `Job_Opening` while it is still in the Open state (in the view provided to candidates). If p did not apply for this particular opening, and if the `Job_Opening` moves to the Closed state, then it may be appropriate to hide this job opening from the candidate. An alternative, that might be more convenient to users so that things don't unexpectedly disappear, would be to still show the artifact instance to the candidate, but grayed out.

In other contexts, it may be desirable to ensure that a given artifact type has persistent visibility for a given stakeholder organization type. The following result states that this is decidable if the window queries correspond to conjunctive queries.

Proposition 8. Let $\mathcal{H} = (\mathcal{S}, \mathbf{Org}, \omega, \nu, \delta)$ be an extended interoperation hub, and suppose that ω has no negation, no disjunction, and only existential quantifiers. Then it is decidable whether $\mathcal{H}$ has persistent visibility. This problem is in PSPACE in terms of the size of $\mathcal{H}$.

Although space limitations prevent inclusion of the proof, we note that the result is demonstrated by showing that it suffices to look at a small set of sequences of snapshots, which are constructed from a bounded active domain and have bounded length.

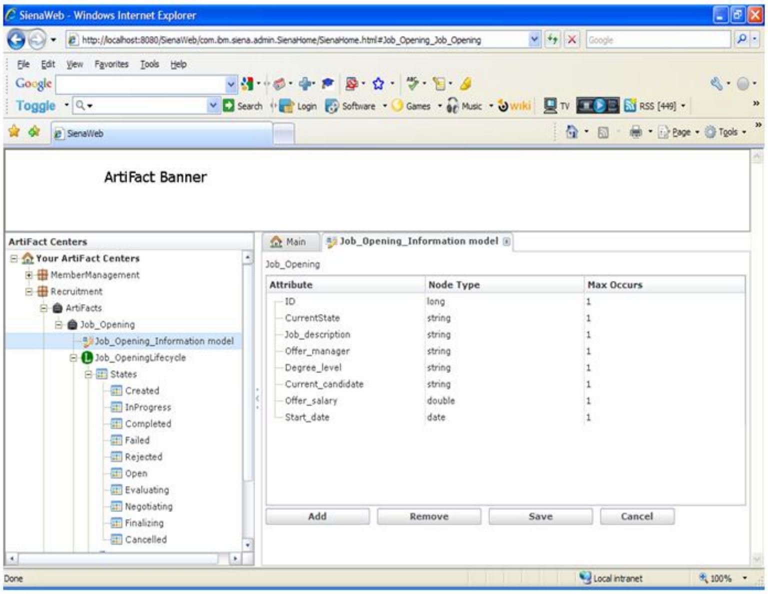
This result leaves several questions open, including finding a tight bound for the complexity of testing persistent visibility under the assumptions of the proposition, and finding the limits of decidability.

In the case of no negation and only existential quantifiers, a straightforward sufficient condition can be developed, that guarantees persistent visiblity. The basic idea is that if some value in a field A of some artifact instance b is used as a witness for p to see an artifact a, then we need to ensure that for the state that b is in, and any state that it can reach from there, the field A can be read (and appended if it is of set type), but not created, deleted or updated.

5 Towards a Prototype Implementation

It is natural to ask how difficult it would be to build a system that supports the creation and deployment of artifact-centric interoperation hubs. It appears that such a system can be constructed in a straightforward manner from an artifact-centric workflow engine. To verify this conjecture, we have been working to create a generic interoperation hub capability on top of the Siena prototype system [8,10]. (As an alternative, one could use the BELA tool developed at IBM Research [19], which operates on top of IBM's WebSphere product line.)

The Siena system includes a user interface for designing artifact-based workflow schemas (that use a state-machine based lifecycle model), a capability to represent the workflow schemas as an XML file along with some XSDs for holding the artifact information models, and an engine that directly executes against the XML file in response to incoming events and tasks being performed. As described in [8], Siena schemas can be specified in Microsoft Powerpoint. To permit easier access to Siena schemas by multiple designers, the Siena team at IBM Research is currently developing



Job_Opening

Attribute	Node Type	Max Occurs
ID	long	1
CurrentState	string	1
Job_description	string	1
Offer_manager	string	1
Degree_level	string	1
Current_candidate	string	1
Offer_salary	double	1
Start_date	date	1

Fig. 7. Screen shot of Siena's web-based GUI, used here to specify the Hiring artifact schema

a web-browser-based tool for specifying artifact schemas. A screen shot of this interface, showing part of the artifact schema for the Hiring example, is shown in Fig. 7.

Siena provides REST and WSDL interfaces so that outside services can invoke the Siena capabilities, including changes to artifact values, and moving artifacts along their lifecycle. Siena provides the capabilities of sending notifications on a selective basis on entry into states, and of showing artifact instances to users, restricted according to a global, role-based specification of Read permissions, and it supports role-based access control based on CRUDE at the state level. (Restrictions on "append" capabilities are not currently supported.) Siena also provides numerous hooks for triggering of events and behaviors, along with guards on transitions and state entry. The main steps in creating a system for supporting inteoperation hubs on top of Siena include (i) enriching the current capability in Siena to recognize roles, so that participants, stakeholder organizations, and stakeholder organization types can all be specified and used during runtime; (ii) modifying the view of snapshots provided to participants to reflect the condensation of states in interoperation hub views; and (iii) incorporating the ability to specify access controls based on windows. Creating these extensions is a work in progress.

6 Related Work

Nigam and Caswell [15] present one of the earliest discussions of the artifact-centric model its application to business modelling. Here, we extend [15] to show how services and applications can interoperate using the artifact-centric approach. In [13], Nandi and Kumaran introduce the concept of Adaptive Business Objects (ABO) to integrate people, processes, information and applications. ABO presents an abstraction of a business entity with explicitly managed state and an associated programming model. In contrast, our interoperation hub model is at a higher abstraction level, understandable by those without IT expertise. Citation [19] describes how the artifact-centric technique has been incorporated into IBM's SOMA method for the design and deployment of business processes. In [12], Müller et al. present data-driven process structures in which Object Life Cycles (OLC) of different objects are linked through OLC dependencies. This approach enables adaptations of data-driven process structures at both design and run time.

Traditional approaches to service composition [9,4,3], use languages such as BPEL[1] to model low-level service interactions. Such implementations focus on the sequence of Web services to be invoked to reach a state goal, and do not explicitly specify how the underlying data is manipulated, or how that data constrains the operation. As a result, the approach is less intuitive than using business artifacts, especially in the case of large shared business processes. In the business process space, [20] models a business process as a collection of interacting process fragments called "proclets". Each proclet is autonomous enough to decide how to interact with the other proclets, and this provides flexibility in workflow execution. In that work, similar to choreography, the interoperation of proclets is not managed or facilitated by a centralized hub.

7 Conclusions

In this paper, we have illustrated how the artifact-centric approach can be used to create hubs that facilitate the interoperation of multiple automonous stakeholders who

[1] http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

have a common goal. Because the basic building block of the artifact-centric approach, namely "business artifacts", combine data and process specification into a single unit, it is straightforward to incorporate three natural forms of access control into the framework, namely, windows (that restrict which artifact instances a participant can see), views (that restrict which attributes and states of an artifact a participant can see), and CRUDAE (a variant of the classical CRUD notion, that restricts the kinds of reads and updates a partipant can perform, based on the current state of an artifact).

The formal framework developed in the paper was used to develop results concerning some of the global implications of placing these access restrictions on a hub, and a prototyping effort indicates that these hubs can be created through a straightforward extension of an artifact-centric workflow engine. This paper provides the starting point for a rich exploration into this new style of interoperation hub. Some theoretical questions of particular interest involve the interplay of, on the one hand, the views and windows exposed to participants and, on the other hand, the sets of achievable sequencs of hub snapshots and integrity constraints on them.

Acknowledgements. The authors thank the extended ArtiFact^{TM} team at IBM Research for many informative discussions about the artifact-centric approach and its application in business contexts. The authors are also very grateful to Fenno (Terry) Heath III, Florian Pinel, and Sridhar Maradugu for facilitating the use of the Siena prototype system [8,10].

References

1. Abiteboul, S., Beeri, C.: The power of languages for the manipulation of complex values. The VLDB Journal 4(4), 727–794 (1995)
2. Abiteboul, S., Bidoit, N.: Nonfirst normal form relations: An algebra allowing data restructuring. Journal of Computer and System Sciences 33, 361–393 (1986)
3. Agarwal, V., Chafle, G., Mittal, S., Srivastava, B.: Understanding Approaches for Web Service Composition and Execution. In: Proc. of Compute 2008, Bangalore, India (2008)
4. Barros, A.P., Dumas, M., Oaks, P.: Standards for web service choreography and orchestration: Status and perspectives. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 61–74. Springer, Heidelberg (2006)
5. Bhattacharya, K., Caswell, N.S., Kumaran, S., Nigam, A., Wu, F.Y.: Artifact-centered operational modeling: Lessons from customer engagements. IBM Systems Journal 46(4), 703–721 (2007)
6. Bhattacharya, K., et al.: A model-driven approach to industrializing discovery processes in pharmaceutical research. IBM Systems Journal 44(1), 145–162 (2005)
7. Chao, T., et al.: Artifact-based transformation of IBM Global Financing: A case study, 2009. To appear Intl. Conf. on Business Process Management (BPM) (September 2009)
8. Cohn, D., Dhoolia, P. (Terry) Heath III, F.F., Pinel, F., Vergo, J.: Siena: From powerpoint to web App in 5 minutes. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 722–723. Springer, Heidelberg (2008)
9. Decker, G., Zaha, J.M., Dumas, M.: Execution semantics for service choreographies. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 163–177. Springer, Heidelberg (2006)
10. (Terry) Heath III, F.F., Pinel, F.: Siena user's guide (2009) (in preparation)
11. Hull, R.: Artifact-centric business process models: Brief survey of research results and challenges. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 1152–1163. Springer, Heidelberg (2008)

12. Müller, D., Reichert, M., Herbst, J.: A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 48–63. Springer, Heidelberg (2008)
13. Nandi, P., Kumaran, S.: Adaptive Business Objects - A New Component Model for Business Integration. In: Proceedings of ICEIS 2005, Miami, FL, USA (2005)
14. Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification. IBM Systems Journal 42(3), 428–445 (2003)
15. Nigam, A., Caswell, N.S.: Business Artifacts: An Approach to Operational Specification. IBM Systems Journal 42(3) (2003)
16. Peltz, C.: Web services orchestration and choreography. IEEE Computer 36(10), 46–52 (2003)
17. Roth, M.A., Korth, H.F., Silberschatz, A.: Extended algebra and calculus for nested relational databases. ACM Trans. Database Syst. 13(4), 389–417 (1988)
18. Sabiv, Y., Yannakakis, M.: Equivalences among relational expressions with the union and difference operators. Journal of the ACM 27(4), 633–655 (1980)
19. Strosnider, J.K., Nandi, P., Kumarn, S., Ghosh, S., Arsanjani, A.: Model-driven synthesis of SOA solutions. IBM Systems Journal 47(3), 415–432 (2008)
20. van der Aalst, W.M.P., Barthelmess, P., Ellis, C.A., Wainer, J.: Proclets: A framework for lightweight interacting workflow processes. Int. J. Cooperative Inf. Syst. 10(4), 443–481 (2001)