Generating Database Schemas from Business Artifact Models

Maroun Abi Assaf and Youakim Badr
University of Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
E-mail: {maroun.abi-assaf, youakim.badr}@insa-lyon.fr

Hicham El Khoury and Kablan Barbar
Lebanese University, Faculty of Sciences, Fanar Campus, Jdeidet, Lebanon
E-mail: {hkhoury, kbarbar}@ul.edu.lb

Received: 27 October 2017; Accepted: 17 November 2017; Published: 08 February 2018

Abstract—Business Artifacts, as an alternative approach to Business Process Modeling, combines both process and data aspects of a Business into the same model. Many works in the literature have focused on defining Artifact-centric processes and graphical modeling notations. But, to the best of our knowledge, no prior work has directly tackled the problem of generating Database Schemas from Business Artifact Models. In this paper, we propose an algorithm that generates Database Schemas from Business Artifact Models (BAMs). The proposed algorithm not only takes into consideration the different data attribute types of Artifacts’ Information Models, but also supports different Artifacts relationships. We also validate our work with a prototype implementation of a Business Artifact Models Modeler and a Database Schema Generator.

Index Terms—Business Artifact, Modeling Notations, Business Process Models, Database schema.

I. INTRODUCTION

As an alternative to traditional activity-centric Business Process Models, artifact-centric models seek to unify both process and data aspects of a business into the same model [1]. This unification of process and data is achieved by modeling business processes as a set of self-evolving and interacting semantic entities referred to as Business Artifacts [2]. The goal of the artifact-centric approach is to provide a holistic and intuitive framework that can be used by business people on a daily basis in order to manage, analyze, and transform business processes with limited or no IT expertise.

As described in [3], an artifact-centric business process is composed from three main components:

1) Business Artifacts that include an Information Model and a Lifecycle. The Information Model is a collection of attribute/value pairs used to store information about the Business Artifacts or relevant objects in the business process. The Lifecycle is a finite-state machine that describes the possible evolutions of a Business Artifact from initial to final states.

2) Services are the basic units of work that operates on Business Artifacts and update their information models. And,

3) Business Rules are declarative Event-Condition-Action Rules (ECA Rules) that are used to invoke Services or to trigger Lifecycle’s state transitions.

In order to model artifact-centric business processes on a conceptual level, the Business Artifact Modeling Notation (BAMN) is proposed in [4]. BAMN provides six graphical constructs: Task, Repository, Flow Connector, Data Attribute List, Condition, and Event. Using the six modeling constructs, users can construct conceptual models referred to as Business Artifact Models (BAMs) that capture the three components of an Artifact-centric business process, and at the same time omit implementation and technical details that are not relevant to end users.

On the other hand, developing information systems that implements BAMs is a serious challenge due to the conceptual nature of BAMs. Moreover, three types of data attributes including Simple, Complex, and Reference types can exist in the information model of Business Artifacts as described in [5]. This diversity in Business Artifact components and relationships leads to a complicated database design when implementing BAMs.

In this paper, we address the problem of generating Database Schemas that support BAMs by devising a suitable algorithm that automatically collects needed information from BAMs and generates Database Schemas. The database tables and relationships are generated according to the different relationships that can exist between Business Artifacts and the different types of data attributes in the Information Models of Business Artifacts.

The remainder of the paper is organized as follow: Section II introduces the Business Artifact Modeling Notation (BAMN) and Business Artifact Models (BAM). Section III describes the Database Schema generation algorithm in addition to an example scenario about a candidate admission process in a university master program. Section IV describes our prototype.
implementation of a BAM Modeler and a Database Schema Generator. Section V is a description of related works. Finally, Section VI concludes the paper.

II. BUSINESS ARTIFACT MODELING NOTATION

In Table 1, we summarize the Business Artifact Modeling Notation (BAMN) constructs and their graphical representations for modeling Business Artifact Models (BAMs). The notation’s main focus is to capture artifact functional properties in terms of data, Events, and Tasks.

The graphical notation is based on six modeling constructs: Task, Repository, Flow connector (read-only and read/write), Data Attribute List, Condition, and Event. Using these constructs, an artifact-centric process can be represented at a conceptual level by modeling interacting artifact lifecycles.

1) Tasks correspond to Services in Artifact Systems and represent units of work to be performed in order to manipulate artifacts and evolve them in their lifecycle thus achieving goals.

2) Repositories denote storage locations into which artifacts can be stored awaiting for future processing if any. For every state of an artifact, an associated Repository is used to store all the artifact instances that are in that state of their lifecycle. Artifact instances can then be pushed into or pulled from particular Repositories as needed.

3) Flow Connectors connect Tasks and Repositories and allow Artifact instances to be transferred between them. Read/write Flow Connectors indicate that Artifacts are transferable between Tasks and Repositories where they are manipulated and evolved with respect to their lifecycles. Read-only Flow Connectors indicate that artifact content is required in read-only mode and no modification is performed, thus the artifact remains in the same Repository. Fig. 1(a) illustrates a Repository connected to a Task through a read/write Flow Connector.

4) Data Attribute Lists are associated to Repositories. They describe Information Models by annotating the conceptual model when Artifacts reach a certain state. Simultaneous definitions of the Information Model and the Lifecycle in the same conceptual model leads to building business processes incrementally without dealing with fine-grained details related to Artifact models. Additionally, the aggregation of Data Attribute Lists allows the generation of Information Models of interrelated artifacts and eventually of Database Schemas. Data attributes are written as attribute-type pairs. Fig. 1(b) depicts a Repository with an attached Data Attribute List.

5) Events are attached to Flow Connectors and specify external Events that are received. For example, a Create New Order Event is generated when clients make new orders.

6) Conditions are also attached to Flow Connectors and specify constraints that should be satisfied in order to

---

Table 1. Business Artifact Modeling Notation (BAMN)

<table>
<thead>
<tr>
<th>Modeling Construct</th>
<th>Graphical Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>[Task]</td>
<td>Units of work operating on business artifacts</td>
</tr>
<tr>
<td>Repository</td>
<td>[Repository]</td>
<td>Storage locations where artifacts await future processing if any</td>
</tr>
<tr>
<td>Read/write Flow Connector</td>
<td>[Artifact]</td>
<td>Transit business artifacts between tasks and repositories</td>
</tr>
<tr>
<td>Read-only Flow Connector</td>
<td>[Artifact]</td>
<td>Read business artifact content from a repository</td>
</tr>
<tr>
<td>Data Attribute List</td>
<td>[Attribute:]</td>
<td>Attribute-Type pairs that characterize a repository</td>
</tr>
<tr>
<td>Condition</td>
<td>[Condition]</td>
<td>Conditions associated to flow connectors</td>
</tr>
<tr>
<td>Event</td>
<td>[Event]</td>
<td>Event associated to flow connectors</td>
</tr>
</tbody>
</table>

---
activate a Flow Connector. Conditions express constraints over artifact attributes by using the defined(attribute)/notdefined(attribute) and scalar comparison (>, <, <=, >=, =, !=) predicates. The defined(attribute)/notdefined(attribute) predicates signify that attribute respectively has or does not have a value. Fig. 1(c) displays a Flow Connector with attached Event and Condition constructs.

III. Generating Database Schemas from Business Artifact Models

In order to generate Database schemas from Business Artifact Models, we define a formal representation for BAMs that serves as the input to the described algorithm. A BAM is defined as a tuple $G=(T, R, F, L, E, C)$ where: $T$ is the set of Tasks, $R$ is the set of Repositories, $F$ is the set of Flow Connectors, $L$ is the set of Data Attribute Lists, $E$ is the set of Events. And, $C$ is the set of Conditions. In the Database Schema generating algorithm, we make no use of Events and Conditions sets which are only relevant for process execution.

We then define an algorithm that takes the formal representation of BAM and generates the corresponding Database schema. This algorithm is based on Data Attribute types constituting the Information Model of Business Artifacts. In BAM, the Information Model is expressed through Data Attribute Lists attached to Repositories.

A. Data Attribute Types

We differentiate between three types of data attributes of artifact’s Information Model: Simple, Complex, and Reference.

1) Simple attributes can hold one value at a time and are used to record information related to Artifact instances, i.e. the instance identifier, the candidate first name, the candidate last name,… Simple attributes are written as Name:Value pairs in Data Attribute Lists and are attached to the Repository of the corresponding Business Artifact, i.e. ApplicationArtifactId:Integer, FirstName:String, LastName:String. In the generated Database Schemas, simple attributes correspond to tables’ columns.

2) Complex attributes represent relations and are expressed as lists of simple attributes. Like relations, complex attributes can hold many tuples at a time. They are used to record information about various objects that are related to the Artifact, i.e., Documents, Interview Results. Complex attributes are written using the Name:{Att1:Type1, Att2:Type2,…} syntax in Data Attribute Lists and are attached to the Repository of the corresponding Business Artifact, i.e. Documents:[Type:Integer, Title:Integer, URL:Integer]. In the generated Database Schemas, complex attributes correspond to tables.

3) Reference attributes refer to other Artifacts that are directly related to the main Artifact in a Parent-to-Child relationship. For example, the Candidate Application Artifact dealing with candidate admission is the parent of a Candidate Interview Artifact dealing with interviewing the corresponding candidate. In BAM, reference attributes are deduced from Tasks creating child Artifacts. In other words, Tasks creating child Artifacts are Tasks that takes an Artifact as input and emit two Artifacts as output, i.e. The CreateCIA Task takes a Candidate Application Artifact instance as input, creates a Candidate Interview Artifact instance, and insert the child.
reference into the reference attribute of the Candidate Application Artifact instance. In the generated Database Schemas, reference attributes correspond to reference tables mapping two Artifact tables together.

```
Input: \(<A, T, R, F, D>\)
- \(A\): Set of Artifacts
- \(T\): Set of Tasks
- \(R\): Set of Repositories
- \(F\): Set of Flow connectors
- \(L\): Set of Data Attribute Lists

Output: Relations Schema \(S\)
for each Artifact \(a \in A\) do
    create relation \(S_a\) for \(a\)
    add primary key attribute to \(S_a\)
    add State attribute to \(S_a\)
    for each Data Attribute List \(l \in L\) do
        if \(l\) is associated to Artifact \(a\) then
            for each Data Attribute \(b \in l\) do
                if \(b\) is simple type then
                    add \(b\) to relation \(S_a\)
                else
                    if \(b\) is complex type then
                        create relation \(S_b\) for \(b\)
                        add primary key attribute to \(S_b\)
                        add foreign key referencing \(S_a\) to \(S_b\)
                        for each Data Attribute \(b' \in b\) do
                            add attribute \(b'\) to relation \(S_b\)
                            end for
                        end if
                    end if
                    end for
                end if
            end for
        end for
    end for
for each Artifact \(a_1 \in A\) do
    for each Artifact \(a_2 \in A\) do
        if \(a_1\) is parent of \(a_2\) then
            create reference relation \(S_{a_1,a_2}\)
            add foreign key referencing \(a_1\) to \(S_{a_1,a_2}\)
            add foreign key referencing \(a_2\) to \(S_{a_1,a_2}\)
            add \(S_{a_1,a_2}\) to \(S\)
        end if
    end for
end for
```

Fig. 3. Database Schema Generating Algorithm

B. Database Schema Generating Algorithm

Fig. 3 illustrates the Database Schema generating algorithm.

First, for every artifact in the input BAM, we create a corresponding relation. We then insert all the simple attributes in Data Attribute Lists corresponding to the artifact into the created relation. A primary key and Status attributes are also inserted into the created relation.

Second, for every complex attribute in Data Attribute Lists, we create a corresponding relation. We insert its constituting simple attributes into the created relation. Moreover, we insert a primary key attribute in addition to a foreign key attribute referencing the corresponding artifact relation.

Finally, for every parent-to-child artifact relationship, we create a reference relation constituted of two foreign key attributes referencing respectively the parent and child artifact relations.

C. Candidate Admission Scenario

In order to illustrate the proposed algorithm, we describe an example scenario about a business process for candidates’ admission into master programs in a university.

The candidate admission process is initiated by a candidate who submits his/her application. This step creates a new application form and records personnel information such as; first name, last name, date of birth, and master program.

After uploading supplemented materials (transcripts, letters of references…) the submitted application is marked as complete, otherwise, it is marked as incomplete and is rejected.

After that, the master program coordinator inspects all complete applications and checks if they are eligible. If an application is not eligible, the candidate is rejected; otherwise, the applicant is selected to be interviewed by an academic committee on a specified date and location.

During the interview, notes and decisions about the candidate are taken by the committee members. After interviews, decisions are made about whether the interviewed applicants are accepted or not. Finally, accepted candidates are included to the final definitive list of master program students.

Since Business Artifacts are goal oriented, in other words since the purpose of every Business Artifact is to perform a particular goal, we include two Business Artifacts in the candidate admission process:

1) A Candidate Application Artifact (CAA) that deals with processing candidate applications and tracks various decisions made about them, and

2) A Candidate Interview Artifact (CIA) that deals with interviewing candidates and collecting and evaluating interviews’ information.

Fig. 4 illustrates the corresponding conceptual BAM constructed using the BAMN described in Section II.
By applying the algorithm listed in Fig. 3, we obtain the following relations schema:

- **CAA** (CAA PK:Integer, ApplicationArtifactId:Integer, FirstName: String, LastName: String, DateOfBirth: String, MasterProgram: String, AllRequiredDocumentsPresented: Boolean, PassedInspection: Boolean, InterviewRequested: boolean, Accepted: Boolean, PublishedToFinalList: Boolean, State: String)

- **Documents** (Documents PK:Integer, CAA_FK:Integer, Type: String, Title: String, URL: String)

- **CIA** (CIA PK:Integer, InterviewArtifactId:Integer, SubjectOfInterview: String, Location: String, InterviewDate: String, State: String)

- **InterviewResults** (InterviewResults PK:Integer, CAA_FK:Integer, MemberName: String, MemberNote: String, Accepted: Boolean)

- **CAA-CIA** (CAA PK:Integer, CIA PK:Integer)

As expected, five relations are created. The first relation **CAA** corresponds to the Candidate Admission Artifact and contains the simple attributes attached to the CreatedCAA, Complete, Inspected, Rejected, WaitingInterview, Accepted, and Archived Repositories. Additionally, **CAA** contains a primary key **CAA_PK** and a State attribute.

The second relation **Documents** corresponds to the complex attribute Documents attached to the CreatedCAA Repository. In addition to its simple attributes, **Documents** contains a primary key **Documents_PK** attribute and foreign key **CAA_FK** attribute referencing the **CAA** relation.

The third relation **CIA** corresponds to the Candidate Interview Artifact and contains the simple attributes attached to the CreatedCIA and Ready Repositories. Additionally, **CIA** contains a primary key **CIA_PK** and Status attributes.

The fourth relation **InterviewResults** corresponds to the complex attribute InterviewResults attached to the Interviewed Repository. In addition to its simple attributes, **InterviewResults** contains a primary key **InterviewResults_PK** attribute and a foreign key **CIA_FK** attribute referencing the **CIA** relation.

Finally, the fifth relation **CAA-CIA** corresponds to the parent-to-child relationship between the **CAA** and **CIA** Artifacts and contains the foreign keys **CAA_FK** and **CIA_FK** of the **CAA** and **CIA** relations.

**IV. PROTOTYPING**

We have developed a prototype that implements BAMN and the Database Schema generating algorithm. The prototype is an HTML5 web application based on the open source JointJS Javascript Diagramming Library [6]. The prototype is made of two modules: **BAM Modeler**, and **Database Schema Generator**.

The **BAM Modeler** allows users to model BAMs using a graphical interface as illustrated in Fig. 5 depicting the Candidate Admission scenario. A **Toolbar** allows the user to select BAMN constructs and inserts them into the
Generating Database Schemas from Business Artifact Models

V. RELATED WORKS

To the best of our knowledge, no prior work has focused directly on the problem of generating Database Schemas from Business Artifact Models.

On the other hand, the concept of using artifacts as building blocks for modeling processes was first introduced in [2]. In [7], artifact-centric processes are implemented and executed using procedural finite-state machines. Work in [3], [8] implement and execute declarative Business Artifact systems based on Business Rules represented as ECA (Event-Condition-Action) Rules.

From a graphical perspective, [2] introduces three modeling constructs: Task, Repository and Flow Connector that can be used to model artifact Lifecycles. Based on the three modeling constructs, nine simple and intuitive modeling patterns that can be used in an artifact Lifecycle are introduced in [9]. These artifact-centric patterns eliminate the need to directly model complex Control-Flow patterns that exist in traditional activity-centric processes, yet they hold all the needed information in order to be translated into low-level Control-Flow patterns.

Since then, many works have focused on defining syntactical and graphical languages and environments for modeling Artifact processes.

In [10], Cohn et al. introduce the Siena, an artifact-centric Business Process Modeling tool that models Business Artifact lifecycles as procedural finite-state machines. Siena provides the capability to generate XML-based Business Artifacts and deploy them into the Siena Runtime Container.

In [11], Artifact Conceptual Flow or ArtiFlow are introduced to a model finite-state machine like artifact processes. On the other hand, artifact processes are declaratively modeled using the Guard-Stage-Milestone (GSM) notations in [12], [13]. Works in [14] have introduced the Business Entities and Business Entity Definition Language (BEDL), an XML-based language, for modeling Business Artifact processes. In [15], artifact processes are defined using Active XML (AXML).

In [16], [17], algorithms for transforming artifact-centric business process models into activity-centric business process models and vice versa are presented.

The Business Artifact Modeling Notation (BAMN) described in this paper is introduced in [4]. BAMN extends the notation introduced in [2] with Data Attribute List, Condition, and Event constructs in order to allow the generation of declarative Business Artifact systems based on Business Rules.

In comparison to other modeling notations, BAMN allows the modeling of both Information Model and Lifecycle in the same model and the generation of declarative Artifact Systems that are based on declarative Business Rules.

VI. CONCLUSION

In this paper, we have presented an algorithm for generating Database Schemas from Business Artifact Models. The Business Artifact Models are modeled using the Business Artifact Modeling Notation (BAMN) which includes six modeling constructs; Task, Repository, Flow Connector (read/write and read-only), Data Attribute List, Event, and Condition.

The proposed algorithm generates Database Schemas based on the three data attribute types of artifacts’ Information Model: simple, complex, and reference.

Copyright © 2018 MECS

I.J. Information Technology and Computer Science, 2018, 2, 10-17
Additionally, the proposed algorithm supports parent-to-child relationships between artifacts.

In order to validate our contributions, we have developed a prototype that implements BAMN and the proposed generation algorithm based on two modules: BAM Modeler, and Database Schema Generator.

Future works seek to generate complete implementations of Business Artifact Models. Specifically, we plan to generate Workflow specifications from Business Artifact Models and execute Artifact-centric processes based on the generated Workflow specifications.

REFERENCES


Authors’ Profiles

Maroun Abi Assaf is a Ph.D student in Computer Science at the University of Lyon. He obtained his bachelor and master degrees in Computer Science from the Lebanese University. His current research interests include query languages, modeling notations, artifact-centric modeling, smart processes, and IoT.

Youakim Badr, Ph.D., joined the faculty of the National Institute of Applied Sciences, France (formally INSA-Lyon) as Associate Professor of Computer Science in 2004. Over the course of his research, he has worked extensively in the area of service computing and information security. His research interests lie in designing and implementing secured IT-enabled services in a socio-technical context. He currently focuses on security challenges in the Internet of Things (IoT), including research topics such as Blockchain-based identity and access controls, security-by-design, model-driven security strategies, on-the-fly IT security services configuration, and federated identity management. Dr. Badr is actively involved in a series of international conferences and also serves as a reviewer for various conferences and journals. Dr. Badr held short-term visiting scholar positions at the University of Sydney, the University of Namur in Belgium and Zayed University in the UAE and spent his sabbatical leave in 2010 at Cornell University and Pennsylvania State University in the United States.

Hicham El Khoury is an associate Professor at the Department of Applied Mathematics and Computer Science, Faculty of Sciences, Lebanese University. He got his master degree in Modelling from Lebanese University, and doctor degree in Computer Science and Telecommunications from Paul Sabatier – Toulouse III. His research interests include network security management information, formal methods, Ontologies and Semantic Web, and Educational Sciences.
Kablan Barbar holds Ph.D in Computer Sciences from the University of Bordeaux I. He is a full professor at the Faculty of Sciences of the Lebanese University and former director of the Lebanese University’s law center. His current research interests include attributed grammars, compiling of markup languages and automatic generation of web applications.

**How to cite this paper:** Maroun Abi Assaf, Youakim Badr, Hicham El Khoury, Kablan Barbar, "Generating Database Schemas from Business Artifact Models", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.2, pp.10-17, 2018. DOI: 10.5815/ijitcs.2018.02.02