# AXART - Enabling Collaborative Work with AXML Artifacts*

Serge Abiteboul[#], Pierre Bourhis[#], Bogdan Marinoiu[◇], Alban Galland[#]
[#]*INRIA Saclay – Île-de-France and University Paris Sud*
*4, rue Jacques Monod, Orsay 91893, France*
firstname.lastname@inria.fr

[◇] *SAP Labs France*
*157, rue Anatole France, Levallois 92300, France*
firstname.lastname@sap.com

## ABSTRACT

The workflow models have been essentially operation-centric for many years, ignoring almost completely the data aspects. Recently, a new paradigm of data-centric workflows, called *business artifacts*, has been introduced by Nigam and Caswell. We follow this approach and propose a model where artifacts are XML documents that evolve in time due to interactions with their environment, i.e. human users or Web services. This paper proposes the AXART system as a distributed platform for collaborative work that harnesses the power of our model. We will illustrate AXART with an example taken from the movie industry. Indeed, applying for a role in a film is a typical collaborative process that involves various participants, inside and outside the film company. The demonstration scenario considers both standard workflow process and dynamic workflow modifications, based on two extension mechanisms: workflow specialization and workflow exception. The workflows, modeled using artifacts, are supported by the AXART system by combining techniques specific to active documents, like view maintenance, with security techniques to manage access rights.

## 1. INTRODUCTION

Many applications are nowadays a combination of workflows and databases. The workflow models have been essentially operation-centric for many years, ignoring almost completely the data aspects. Recently, a new paradigm of data-centric workflows, called *business artifacts*, has been introduced by Nigam and Caswell [7]. We follow this approach and propose a model where artifacts are XML documents that evolve due to interactions with their environment, i.e. human users or Web services. Another important characteristic of current applications is distribution. Companies define procedures involving their employees and their partners. Parts of these procedures may take place at their partners. The access to the data and applications is governed by access rights based on security roles given by the company. This paper proposes the AXART system as a platform for collaborative work in a distributed environment that harnesses the power of our model. The name of the system is derived from ActiveXML (short AXML), since we have developed it on top of the AXML distributed system [1].The general idea of building business artifacts on top of AXML documents has been introduced in [2].

The AXART system has several interesting characteristics: the simplicity of the interaction with the user, the dynamic modification of the workflow, and the use of artifacts in a distributed environment. The dynamic modification of the workflow and the management of the access rights are new with respect to previous works. We illustrate these with a real-life example: the application to a movie role in the movie industry, as inspired by [8]. This procedure is representative for widely spread cases where the workflow of a procedure owned by a company or a public institution is shared with its partners. We claim that AXART could have a very effective impact on the way such distributed procedures can be handled by the information systems. The example shows how a standard workflow is modeled with AXART and how users can easily interact with the artifacts using forms and modifying the states of the artifacts, effectively running the workflow. The artifact changes are monitored by the AXML system and notifications are sent to potentially concerned users, so that they can take appropriate actions. The example also illustrates the dynamic modification of the workflow by two AXART mechanisms, namely *workflow specialization* and *workflow exception*, that allow users with proper rights to modify the workflows during their runs. This underlines the flexibility of the workflows built with AXART.

The changes on the data and the workflow of an artifact are governed by the security rights that a user has for that artifact. In an open environment, a user may obtain a (potentially partial) copy of the artifact she needs to work on, that is a document, and process it remotely. An artifact contains the set of rules that define its valid evolution with respect to the workflow. When the artifact returns to the secure system, the system can check that the user modified the artifact in a way that respects the procedure and her access rights.

The main idea of business artifacts is to represent the workflow as data rules involving queries on the documents. The system maintains dynamically the views needed to verify the evolution of the workflow. It uses for that monitoring techniques like view maintenance on (active) documents in presence of positive updates. AXART combines these techniques with security techniques for managing access rights. This allows the management of a large collection of artifacts with data access control.

The rest of the paper is organized as follows. In the next section,

*This work is partially supported by ERC Advanced Grant Webdam on Foundation of Web data management. It started while *Bogdan Marinoiu* was a Ph.D. candidate and working at INRIA.


*Proceedings of the VLDB Endowment,* Vol. 3, No. 2

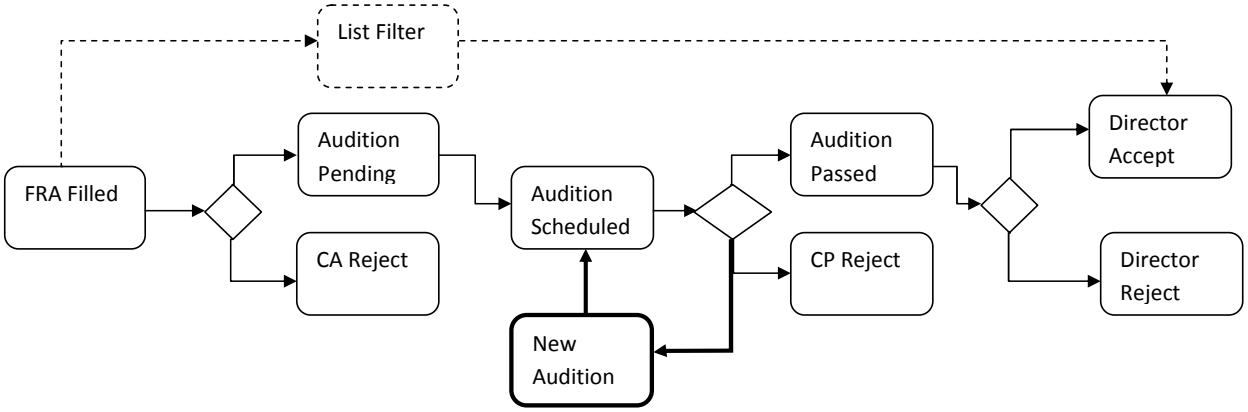


**Figure 1: The standard workflow (solid), its exception (dotted) and its specialization (bold)**

we present the setting and the demonstration scenarios. Section 3 describes the model, the architecture and the user interface of the AXART system. The last section is a conclusion.

## 2. THE DEMONSTRATION

*The Setting.* The real-life situation that we consider is the *Role Application Procedure* at Hollywood. It is a workflow owned by a Film Company, whose purpose is to cast actors for a movie. There are typically four types of participants involved in such a workflow: the *Impresario* acting for the interest of the Film Actor, the *Casting Panel*, the *Casting Assistant* and the *Film Director*. Each of them corresponds to a security role in our system.

A Casting Assistant sends a *Film Role Proposal* to a set of selected Impresarios. A Film Role Proposal is an artifact that contains information like the role name, the role description and an editable part, a sub artifact: the *Film Role Application* (FRA). This part will be detached from the artifact Film Role Proposal, filled in by the Impresario with personal data of the Film Actor and sent back to the Casting Assistant. The FRA is the artifact that we consider for modeling in our demonstration.

The Casting Assistant filters the applications e.g., eliminates the ones that are not filled in properly and schedules auditions for the Film Actors. At the due date, the audition takes place in front of the Casting Panel. It might result in the rejection of the application or in a "passed audition" verdict. This procedure is a standard version of the workflow, as defined initially for the class of FRAs.In addition, the Casting Panel may suggest new auditions and in this case details need to be provided by the Casting Panel for organizing them. This is a case of Workflow specialization.

A "passed audition" verdict means that the Actor is eligible for the role and that its application will be considered by the Director. Out of a pool of FRAs that passed auditions, the Director will choose one. Of course, the Director might have a short list of favorite actors and could make her choice at any moment if she spotted an application of a favorite actor, even without the actor passing an audition for the role. This is a case of Workflow exception.

Observe there are two kinds of users for this application. Some users are the employees of the Film Company that owns the workflow: Casting Panel, Casting Assistant and Director. They simply log in the system and all their actions are directly supervised by the system. The Impresario is a partner of the Film Company and fills in the FRA artifact remotely, i.e. on a system not owned by the Film Company. Some verifications about her actions need to be performed when she sends the artifact FRA back to the Film Company's system. This task will be done by the Casting Assistant that is fully automatic in our demonstration.

*The Demonstration Scenarios.* We start with the standard workflow (the regular boxes and arrows in Figure 1) that is predefined for an artifact of class FRA. The boxes represent stages in the workflow, while the diamonds stand for choice operators. In the case of the standard casting workflow we suppose that only an audition needs to be passed. At any moment, the artifact will be viewed by the workflow participants as a form with fields to fill in.

In the first demonstration scenario, the Impresario of the actor *John Travis* receives a Film Role Proposal from the Film Company. She fills the attached FRA with details about the actor and sends it back to the Film Company. At the reception, an audition is automatically scheduled by a Casting Assistant. When the Casting Panel logs in, she will be able to choose one of the FRAs in the stage Audition Scheduled. When choosing the FRA of John Travis, she will need to fill in a form as the one in Figure 2. Supposing the decision of Casting Panel is positive, the FRA of John Travis will be considered by the Director at her next log in.

In the second scenario, we present the principle of Workflow exception. The *Director* may have a list of preferred actors for the main role. The Director logs in first. She will be able to edit a list of her preferences for the Role *Batman* in the film *A Batman Romance*. She will place the actor *John Travolta* as her first preference in the list and she will log out. This is a Workflow Exception. The dotted arrow and box in Figure 1 shows the exception when the Director chooses to approve the proposal of John Travolta at the first step. When the application of John Travolta matches the list of preferences, i.e. it enters the *List Filter* stage, a notification is sent to the Director that should log in and explicitly validate it.

The third scenario demonstrates how the workflow can be changed by Workflow specialization during the run of the application. The Casting Panel wants the actor *John Who* to pass another audition. The bold arrows and boxes show the extension of the workflow.

## 3. THE AXART SYSTEM

### 3.1 The Model

In our system, an artifact is an AXML document, which in turn is an XML document with embedded function (Web service) calls. These function calls are special nodes, that once activated, trigger calls to the corresponding Web services with XML data as parameters. The XML data obtained as answers are integrated as siblings to the XML node that corresponds to the function call. Thus, an

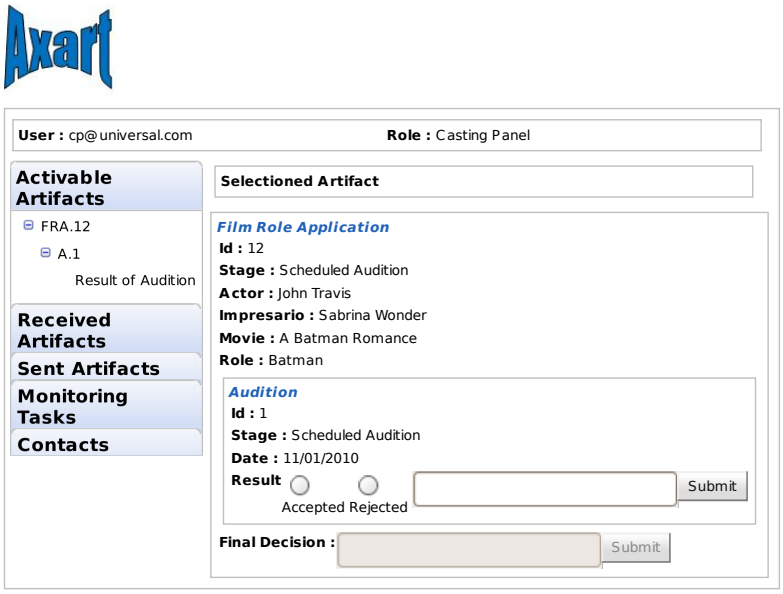


**Figure 2: Form snapshot for a stage (Audition Scheduled)**

XML document evolves in time. This is the *client* aspect of an artifact. An artifact also behaves as a server, by exposing an interface made of services. By calling an artifact service with a proper argument, a client receives answers in return. An artifact uses these calls to obtain input from users for instance.

AXART builds on the previous AXML concepts. But the concept of business artifacts as AXML documents is richer. The principles of artifacts supported by AXML are presented in [2]. In the original AXML system, the entire logic for activating function calls was available at the peer level. The peer was the manager of the document. In the new AXART system, the logic is at the level of the business artifact, i.e. at the document level. This logic specifies the *workflow* of the business artifact. An artifact may have at every moment several (sub)artifacts.

Business artifacts are partitioned in classes. A class of an artifact describes the static constraints that the document has to satisfy, the possible basic properties, the workflow, the rights to activate the functions, and the rights to change the workflow. To each class, one can associate possibly several *stages*, that are macro-states used to define the workflow of the artifact. An artifact is at any given time an AXML document (or part of it). It contains its data and the data of its sub artifacts, as well as information about the workflow stages, the artifact and its sub artifacts are in. To each stage of the workflow is associated a combination of basic properties (Boolean combination of tree patterns) described in this class or in the classes of the subartifacts. An artifact can be in a stage if its data satisfies the formula associated to it. Finally, the workflow of a class describes the possible transitions from a stage to another stage. Because we do not impose that an artifact should satisfy only one stage formula at a time, it is possible to change the stage without updating the content of the artifact. An artifact evolves when a user calls a function of the AXML document. Following the activation of a function, the user fills the fields of the artifact. The update is accepted if it leads to valid workflow stages for the artifact and for each of the artifacts this artifact belongs to. To each function is associated a pattern that describes the returned document (possibly a subartifact). The right to activate a function depends on the role of the user and on the stage of the artifact that contains the function. This is new with respect to previous works [2].

In a distributed environment, it might be the case that artifacts are processed outside of the system that originated them. An artifact contains the set of rules that define its valid evolution with respect to the workflow, so these rules can be enforced by the partner systems. However, there is no guarantee that the partner system will not violate them. In the example of the Film Role Procedure, the Impresario fills in the application for the role outside the Film Company's system. But she knows the rules that need to be followed for the application to get accepted. In our model, an artifact may leave and then return into the secure system. The system only gives the artifact to a user that is authenticated and has the proper access rights. When the artifact returns in the secure system, the system checks that the user modified the artifact according to the rights associated with her role, and may raise exceptions if it was not the case. Observe that all the participants, except for the Impresario, log in and work in the secure environment offered by the system of the Film Company. Checking the evolution of an artifact against security rights is new.

Our system allows two kinds of modifications of the workflow of an artifact: the *specializations* that constrain the workflow, and the *exceptions* that relax the workflow. These workflow extension mechanisms are also novelties proposed by our demonstration. A *specialization* is enforced by allowing some new functions and some new subdocuments (possibly an artifact) in the static constraints, and by adding some constraints to the stage of the workflow. This is for instance the case when several auditions need to be scheduled. The rights depend on the role of the user. The allowed changes are expressed by a type such that the new artifacts are included in this type. In the case of specialization, a user can further constrain a stage formula by adding a conjunction of basic properties. The specialization can reduce the rights of a user to call a function in a particular stage. Similarly, *exceptions* can be specified. In the case of an exception, a user can weaken a stage formula

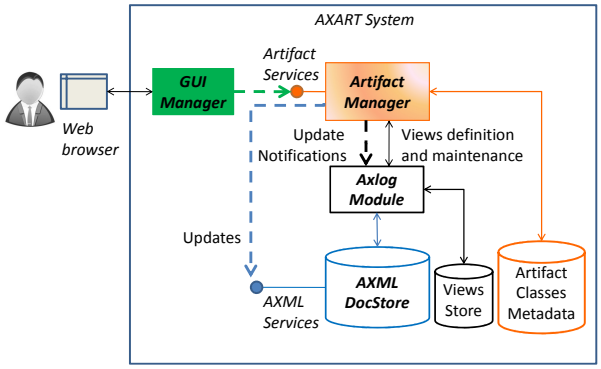


**Figure 3: Architecture of a peer of the AXART System**

by adding a disjunction of conjuncts of basic properties. Moreover, she can give to another user the right to call a function in a particular stage. For instance, in the main scenario, the Director accepts an application if the actor passed the auditions successfully. In the second scenario, the Director relaxes the constraints for John Travolta. The Director accepts an application if the actor passed the audition successfully, or is called John Travolta.

## 3.2 The Architecture

The main goal of the system is to manage the updates of artifacts. An update waits to be applied until the Artifact Manager checks that the artifact and the higher-level artifacts can pass to a next stage. The properties of stages are expressed by patterns that have to be checked. The Artifact Manager is helped by the Axlog Module that maintains incrementally the tree-patterns used to define the properties of the stages. Finally, the update is realized by calling a function of the AXML system that stores the artifacts.

As already mentioned, the AXART system is built on top of the AXML system, that is a manager of active documents. Like AXML system, the AXART system is distributed. Access control and monitoring queries are handled at the peer level. The architecture of an AXART peer is presented in Figure 3. In the demonstration, the system of the Film Company will be modeled as one peer, and the system used by the Impresario to fill in the form as another peer.

The AXML framework supports updates of artifacts (as active documents), by calling functions. The system maintains views and thus maintains the values of the stage formulas, using Axlog [3]. Thanks to a subscription service that we developed [3], the users may declare themselves to the AXART system as having particular roles and receive alerts about artifacts that concern them. For instance, the Director can register to the system. She is notified whenever a favorite actor applies for the role of Batman. Similarly, an Impresario can register to the system and be notified whenever an application of one of her actors has been approved.

The system has been extended with *artifact services* of several types. First, there are services that add *knowledge* to the peers, e.g., metadata about artifact classes, or that provide information about the *workflow stage* for a particular artifact. We have also added security services for checking the access rights when changes are performed. These are supported by the Artifact Manager.

## 3.3 The User Interface

The system proposes to the users a Web interface for managing the artifacts. After logging into the system, the user has access to the set of artifacts that she can modify, has sent or received. She has also a list of functions that she can activate according to her security rights. Finally, she can define or change monitoring queries.

When a user chooses to activate a function, a form as in Figure 2 is presented to her. The system may propose a set of possible valuations to the user. In Figure 2, the user can choose one of the *Accepted/Rejected* options. She can create or import artifacts from other systems, e.g. received by email.

In order to modify the workflow, the system proposes a high level language that supports the creation of an artifact of a known class. The user may redefine the constraints of stages using a set of basic properties suggested by the system. Defining new properties demands an expert, since tree-pattern queries are involved.

# 4. CONCLUSIONS

The system AXART is related to platforms that help users model Web applications, such as FORWARD [4] and webml [5]. It is also related to workflow design languages such as business artifacts [6] and to mashup systems such as Yahoo Pipe [9]. The first two systems help the user describe Web applications in terms of interactions. The business artifact model is used for complex applications, while mashup systems are used to perform automatic data integration. Our system focuses on collaborative and dynamic applications. In particular, the workflow can change during the life of an artifact.

The system AXART is based on Active XML technology [1] and efficient maintenance of views [3]. Recent work [2] introduced business artifacts on top of the Active XML. But there the workflow for an artifact was pre-defined, so only data could be updated. In AXART the workflow itself can be modified by a user, who is capable of adding new stages and integrating these into the previous workflow, i.e. the set of rules can be updated. Another novelty is the role-based access control that confines the way data and workflow are modified. The artifact manager implements the different security policies. Our approach to security in an open environment resides on the local enforcement of security rules. We also suppose that modifications performed outside are associated to the precise user that checked out for the artifact considered and returned it afterwards. More complex approaches for security in a distributed setting based on the XML data signature are currently under study.

# 5. REFERENCES

[1] S. Abiteboul, O. Benjelloun, and T. Milo. The Active XML project: an overview. *VLDB J.*, 17(5):1019–1040, 2008.

[2] S. Abiteboul, P. Bourhis, A. Galland, and B. Marinoiu. The Artifact AXML model. In *TIME*, 2009.

[3] S. Abiteboul, P. Bourhis, and B. Marinoiu. Efficient maintenance techniques for views over active documents. In *EDBT*, pages 1076–1087, 2009.

[4] G. Bhatia, Y. Fu, K. Kowalczykowski, K. W. Ong, K. K. Zhao, A. Deutsch, and Y. Papakonstantinou. Forward: Design specification techniques for do-it-yourself application platforms. In *WebDB*, 2009.

[5] S. Ceri, P. Fraternali, and A. Bongio. Web modeling language (webml): a modeling language for designing web sites. *Computer Networks*, 33(1-6):137–157, 2000.

[6] R. Hull. Artifact-centric business process models: Brief survey of research results and challenges. In *OTM Conferences (2)*, pages 1152–1163, 2008.

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

[8] Wikipedia. `http://en.wikipedia.org/wiki/Casting_%28performing_arts%29`.

[9] Yahoo Pipe. `pipes.yahoo.com`.