Workflow Mediation using VorteXML

Vassilis Christophides
Institute of Computer Science, FORTH,
Vassilika Vouton, P.O.Box 1385, GR 711 10,
Heraklion, Greece
christop@ics.forth.gr

Richard Hull Akhil Kumar Jerome Simeon
Bell Laboratories, Lucent Technologies
600 Mountain Avenue,
Murray Hill, NJ, USA
{hull, akhil, simeon}@research.bell-labs.com

1 Introduction

The Internet and Web are revolutionizing the way companies interact with their suppliers, partners, and clients, by enabling substantial automation of the full spectrum of their business activities. In the 21st century economy, the primary form of automation will be B2B e-commerce, in which enterprises interact with each other through entirely automated means. As an example, consider an electronic market place in a vertical industry segment, in which suppliers and buyers tie into a common IT infrastructure to exchange goods and services. This forms a *supply chain* in which buyers investigate possible suppliers, (b) check the terms and conditions under which suppliers can do business, (c) interoperate with the suppliers' enterprise support systems, i.e., workflows, and (d) monitor ordering/purchasing for possible delays, unexpected events, react to such events, etc. This paper presents a new framework for specifying, enacting and supervising e-services on the Web, based on XML and rules-based support for *products/services description* and *workflow mediation* across organizations.

Traditionally, WorkFlow Management Systems (WFMS) have focused on homogeneous and centrally controlled environments for binding people and processes within the boundary of a single organization. In the context of B2B e-commerce, WFMSs need to support *collaboration* between various *autonomous* parties, some of which may even have conflicting business goals. More precisely, they must cope with heterogeneous enterprise support environments (e.g., through different WF systems), to model the interaction of independent partners by abstracting the internal details of their activities (e.g., through different WF schemas), and finally to facilitate flexible linking and monitoring of inter-enterprise processes (e.g., through different WF enactments). To address these challenges we are currently developing a workflow mediation middleware which relies on three basic technologies: (a) the XRL workflow specification language [13, 18] for representing in XML heterogeneous workflow schemas and enactments, (b) an XML query language [4, 5] for manipulating both complex product and service descriptions, and (c) the Vortex rule-based language [12, 6] that supports heuristic reasoning in order to take on-line business decisions during the workflow execution.

Recently, workflow interoperation has received considerable attention. Numerous research projects and prototypes have been proposed [16, 2] while basic interoperability between various vendor WFMSs has been a subject of standardization efforts by the Object Management Group (see Workflow Management Facility [9]), and the Workflow Management Coalition (see the Xf-XML binding of the WfMC Interface 4 [1]). Workflow integration in an e-business setting has also been addressed in projects such as CrossFlow [7], WISE [14], FlowJet [17], InterWorkflow [10] and MOCASSIN [8]. Compared to these projects, our work introduces an explicit

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

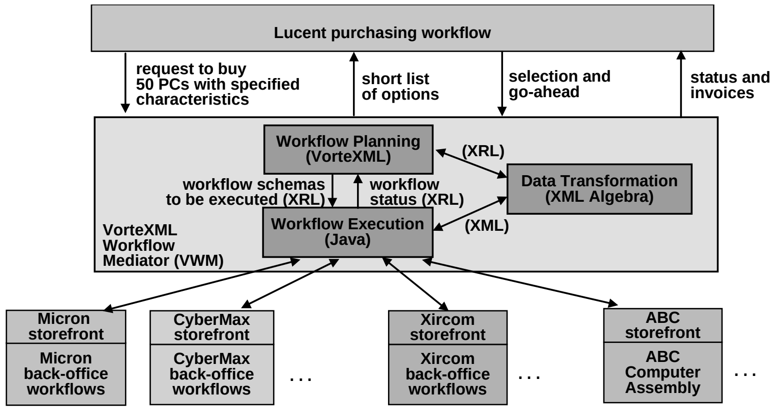

Figure 1: Example Workflow Mediator for Managing Complex Purchases

workflow mediator for enterprise processes, which goes beyond a simple mediation of the data that is passed between workflows. In particular, our mediation middleware (a) enables uniform manipulation of workflow flow control (e.g., enterprise processes) and business data (e.g., product catalogs) as they are both represented in XML; (b) allows us to inspect workflow enactments, as well as to construct dynamically new workflow schemas linking cross-enterprise processes; and (c) permits different kinds of reasoning and heuristics when constructing the new schemas, when dealing with delays, etc. Finally, there is significant industrial interest in the development of infrastructure to support e-service interoperability e.g., the Universal Description, Discovery and Integration (UDDI) standard; the XML-based Web Service Description Language (WSDL); e-speak; and BizTalk[1] (respectively) frameworks. These efforts complement our approach: they aim to be a lowest common denominator along all classes of Internet applications, whereas our work focuses more on more specific e-services based on workflow technology.

This paper presents research in progress. Many of the concepts presented have yet to be worked out in detail and tested in realistic contexts. We expect our framework to evolve as more details are filled in.

2 Workflow Mediation

The need to coordinate inter-organizational workflows and e-services arises in a broad variety of business contexts (e.g, B2B, B2C, etc.), which implies that different architectures and modes of processing will be needed. The focus of the current paper is on presenting a novel family of key building blocks that can be used in most if not all of these contexts. Extending the approach presented in [11], we describe here some specific ways that these building blocks can be integrated. To provide grounding for the discussion we focus on a representative form of workflow coordination. This is called workflow *mediation*, because of its analogy with database mediation, but in our context mediation concerns both enterprise data and processes.

A workflow mediator can be used to help insulate one organization from the intricacies of interacting with multiple other organizations that together provide some coherent family of e-services. Figure 1 shows a representative *VorteXML Workflow Mediator* (VWM) that might be used by an organization such as Lucent to substantially automate the selection and purchase of PCs (including outsourced assembly and shipping).

VWMs provide infrastructure to support basic forms of planning, scheduling and reactive execution [15]. As shown in Figure 1, a VWM has three main modules, for Planning, Execution, and Data Transformation. The *Planning module* builds workflow schemas based on goals (e.g., investigate possible PCs to be purchased; execute the purchase and assembly of selected PCs). Currently, it uses a form of hierarchical planning [3], in

[1] www.uddi.org; msdn.microsoft.com/xml/general/wsdl.asp; www.espeak.net; and www.biztalk.org

```
type Micron_PC = micron_PC [
       @computer_id [ Integer ],
       model        [ String ],
       processor    [
         model        [ String ],
         speed        [ String ]
       ],
       memory       [ String ],
       price        [ Integer ],
       ... ]
type Uniform_PC = uniform_PC[
       @pc_id       [ Integer ],
       @brand       [ String ],
       @model       [ String ],
       processor    [ String ],
       memory       [ Integer ],
       price        [ Integer ],
       ... ]
```

```
fun Transform_PC_Format(p:Micron_PC |
                          Cybermax_PC | ... ) : Uniform_PC =
  match p
    case q:Micron_PC do
      Uniform_PC [
          @PC_id       [ q/@computer_id/data() ],
          @brand       [ "Micron" ],
          @model       [ q/model/data() ],
          processor    [ q/processor/model/data() ],
          memory       [ cast (q/memory/data()) : Integer ],
          price        [ q/price/data() ],
          ... ]
     ...    (* similar mappings for Cybermax_PC and others *)
```

Figure 2: Illustration of type specification and queries in XML Algebra

which workflow schema templates of differing granularity are selected from a library and then expanded by filling in the slots of those templates appropriately (see Subsection 2.2). This construction of workflow schemas from other workflow schemas can be done in a hierarchical manner and is also one form of *schema splicing* (see Subsection 2.3). The outputs of the Planning process are workflow schemas expressed in a dialect of XML, in the spirit of XRL [13, 18] and recent commercial e-services middleware (e.g., Excelon, Microsoft, HP and BEA Systems). We use a generalization[2] of XRL based on an extended Petri Net model that includes explicit flowchart and parallelism constructs. The Planning module is specified using VorteXML, (see Subsection 2.4), a high-level, declarative language that simplifies the specification and maintenance of the mediator.

The *Data Transformation module* is essentially an XML query processor, which is used by the other two modules. Currently this module uses the XML Algebra of [4], a working draft of the W3C. The higher-level XML query/integration language will be used as it becomes available. Unlike previous XML query languages, the XML Algebra permits complex restructuring of XML data, and associative access analogous to relational joins. The *Execution module* executes and monitors the workflow schemas that are produced by the Planning module. The Execution module also monitors the progress of the workflow executions, and reports back to the Planning module if delays exceed specified thresholds or if substantial exceptions arise.

A possible flow of activity for the VWM of Figure 1 is given below.

1. Create a workflow schema for investigating possible suppliers, assemblers, etc.; this workflow will incorporate information about the selection criteria for the PCs to be purchased.
2. Execute the previous workflow to determine a "short list" of options.
3. As a final part of the execution in step (2), specific PCs, vendors, assemblers, etc., are selected, either automatically or by asking a human at Lucent.
4. Create a new workflow for ordering the specific PCs selected in step (3), for shipping, payment, etc.
5. Execute the generated workflow.
6. During execution monitor for delays and/or exceptions.
7. If delays/exceptions occur, then go back to step (4) to modify the workflow.

In this process flow, steps `1` and `4` are performed by the Planning module. The Execution module performs steps `2`, `3`, and `6`, and involve execution of workflow schemas. Following subsections show how the coupling between key elements of the infrastructure are used together to support advanced mediation between e-services.

[2]In [18] XRL is used to specify only workflow enactments; XRL is generalized here to represent workflow schemas.

```
let PC_purchase1 : XRLTemplate =
  Route [
    Sequence [
dress ],
      Parallel_sync [
        Slot [ param [ buy_pc_template] ],
        Slot [ param [buy_modem_template] ]
      ],          (* end Parallel_sync *)
        Slot [ param [ assembly_template ] ]
    ]             (* end Sequence *)
  ]               (* end Route *)
```

```
let buy_from_Micron1 : XRLTemplate =
  Route [
    in_parameters [ pc_model : string , ship_to : Ad-

    out_parameters [ invoice : Invoice ],
      Task [ @name [ send_order_to_Micron ] ],
             @address [ "buy_Micron1.exe" ],
             @d_read [ "pc_model", "ship_to" ],
             @d_update [ "invoice", "order_form" ]
    ],         (* end Task *)
    ...        (* more Tasks *)
]              (* end Route *)
```

Figure 3: PC_purchase1, a generic template for purchasing PCs, and buy_from_Micron1, a base template that could fill the buy_pc_template slot

2.1: Data Transformation. Data transformation and integration is used at multiple stages during the operation of workflow mediators. In this subsection, we briefly present the strongly typed XML Algebra used for this purpose, with an example illustrating how heterogeneous product descriptions from different providers can be mapped into a uniform format. Examples involving the creation of XRL workflow schemas are given later on.

Figure 2 shows three XML algebra expressions. We use the syntax of the document [5] presenting XML Algebra, which is more succinct than the syntax used by XML DTDs and Schemas. The upper left shows the (hypothetical, simplified) schema for Micron PCs, the lower left shows an XML Schema that can be used to hold relevant information about these and other brands of PCs in a uniform format, and the right side shows an XML Algebra function that maps data about Micron and other PCs into the uniform format. In XML Algebra the schema specifications are in fact type specifications.

In the `Micron_PC` element, the first entry is an attribute (indicated by the `@`), and the others are child elements. The `processor` field is structured, having two child elements. The function `Transform_PC_Format` holds a query that takes as input an XML document `p` with type `Micron_PC` or `Cybermax_PC`, etc., and transforms it into an XML document of type `Uniform_PC`. The `match` construct provides for a different treatment for different input types. The keyword `data()` is used to access scalar data (strings, integers, booleans, etc.).

Note that the VorteXML mediator can take advantage of the decision engine to offer advanced transformation services, such as data cleaning, selection between redudant information sources, etc.

2.2: XRL workflow schemas and templates. This and the next subsection together provide an illustration of how the Planning module creates workflow schemas (cf. steps 1 and 4 in the mediator's processing), using the technique of schema splicing. This subsection illustrates the pieces of workflow schema that are used, and the next one illustrates how they are spliced together.

As noted above, the mediator uses (a generalization of) XRL to represent workflow schemas. The schemas can be completely specified, or might be *templates* which provide the high-level specification of a workflow but include `Slot` elements where selected other template can be inserted. Templates without slot elements are called *base templates*. Figure 3 shows on the left `PC_purchase1`, an extremely simplified template that might be used for purchasing PCs (cf. step 4 of the mediator's processing), and on the right `buy_from_Micron1`, a simplified base template that might be used to fill the `buy_pc_template` slot of `PC_purchase1`. In `PC_purchase1` a sequence of two activities will occur. The first activity involves the parallel execution of two inserted tasks (which will involve ordering items from a PC vendor and a modem vendor, respectively, and having them shipped to an appropriate location). The second activity consists in executing an inserted task, which asks an assembler to assemble the PC and modem, and ship to another location.

The base template `buy_from_Micron1` may be used to purchase a PC from Micron. This template schema includes a task to send the `pc_model` and `ship_to` address to Micron, and receive an `invoice` in return. The `address` attribute names the program required to perform this task. This program might invoke

```
let mymapping : Mapping =
  map [ param [ "pc_model" ],
        entry_name [ "Millennia MAX XP" ] ],
  map [ param [ "buy_pc_template" ],
        entry_name [ "buy_from_Micron1" ] ],
  ...
```

Figure 4: Illustration of a mapping, that specifies how slots of `PC_purchase1` should be filled

a wrapper or other functionalities that are resident in the VWM, or provided by external systems. The entries `in_parameters` and `out_parameters` permit parameters (typed XML data) to be passed in and out of the template. Inside the task, attributes `d_read` and `d_update` indicate what XML data can be read or updated (respectively) by the task. In addition to parameters, the first task uses an initially empty `order_form` document to build up the actual order form that is sent to Micron. In practice, this schema should include actions to be taken if Micron doesn't respond in a timely fashion, or if the PC model is not available, and actions to ensure that the PC is eventually received in good condition Our forthcoming work will provide more details on the schema of XRL templates, timing requirements, and handling of simple exceptions.

2.3: Schema splicing. "Schema splicing" refers to the creation of new workflow schemas from existing workflow schemas and templates. Here we illustrate schema splicing for the hierarchical planning in VWMs. Figure 4 illustrates a mapping (with type `Mapping`) that provides the correspondence between the parameters appearing in `PC_purchase1` and either scalar values (such as `"Millennia MAX XP"`) or selected elements (such as `buy_from_Micron1`) of a template library. We assume that the Planning module has selected `PC_purchase1` and has constructed `mymapping` to fill in the parameters of `PC_purchase1`. The policy guiding the construction of `mymapping` is expressed in VorteXML. It includes specifications of how to choose (based on high-level and/or detailed input from humans) the PC and modem models, the vendors, the assembler, etc., and also includes commands for combining these choices to create the list `mymapping`. An XML Algebra query can now be used to combine the template `PC_purchase1`, the mapping `mymapping`, and elements of a template library, to form a workflow schema that will obtain the desired PCs.

In the example just presented, a single layer of hierarchical planning was used, i.e., slots in the top-level template (`PC_purchase1`) were directly filled with base templates. In general, multiple layers can be used, such that slots of the top-level template are in turn filled with successive lower-level templates, and finally with base templates at the lowest level of the hierarchy.

2.4: Heuristic reasoning with VorteXML. A key activity of the Planning module is deciding between alternatives, e.g., choosing a top-level template from the library and then choosing templates that are to plug into the slots of that template. Additional decisions and prioritizations must be made during workflow execution, e.g., to select the brand of PC, modem, etc., and to react to exceptions. In both of these areas we use the VorteXML language, chosen because it combines straightforward flowchart constructs along with the novel *DecisionFlow* construct. DecisionFlows provide a high-level, declarative language for specifying families of rules that make intricate decisions. DecisionFlows support a limited form of rule chaining, and can incorporate both formal and heuristic forms of reasoning. VorteXML generalizes the Vortex language [12] to work with XML data[3].

Vortex DecisionFlow is "attribute-centric" in the sense that each decision (or rule family) is focused on assigning values to one or more target attributes (e.g., the selection of a specific PC model, workflow schema template, etc.) based on groups of input and intermediate attributes. The computation of intermediate and target attributes may be specified in a variety of ways, including the use of database queries, user-defined functions, or, most importantly, attribute rules and "combining policies". An attribute rule has the form `if <condition> then contribute <expression>`, where `<expression>` provides a value. The combining policy

[3]The current Vortex language works with relational data, i.e., with scalars, tuples, and lists of tuples

can be essentially any aggregation function taking as input the zero or more values contributed by rules with true condition and producing a final value for the attribute. Simple combining policies include "max of true rules" and "sum of true rules". More sophisticated combining policies can perform grouping and aggregation of XML data from multiple sources, e.g., into a sorted list, or to form a grounded workflow schema from templates.

At a higher level, the use of attributes in DecisionFlows permits a simplified form of chaining between groups of non-recursive rules. The DecisionFlow paradigm also uses rules to control which attributes will be computed during a given execution. These *enabling rules* can help to avoid irrelevant computations, and to support a trade-off between how refined a decision is vs. how much resource is consumed in making that decision.

References

[1] W. M. Coalition. Workflow standard - interoperability wf-xml binding document number wfmc-tc-1023, April 1999. `http://www.aiim.org/wfmc/standards/docs/Wf-XML-1.0.pdf`.

[2] A. Dogac, L. Kalinichenko, T. Özsu, and A. Sheth, editors. *Workflow Management Systems and Interoperability*. Springer Verlag, 1998.

[3] K. Erol, J. Hendler, and D. Nau. Semantics for hierarchical task-network planning. Technical Report CS-TR-3239, Computer Science Department, University of Maryland, 1994.

[4] P. Fankhauser, M. Fernandez, A. Malhotra, M. Rys, J. Siméon, and P. Wadler. The XML query algebra. W3C Working Draft, Dec. 2000. `http://www.w3.org/TR/query-algebra/`.

[5] M. Fernandez, J. Siméon, and P. Wadler. A semi-monad for semi-structured data. In *Proc. of Intl. Conf. on Database Theory*, London, UK, Jan. 2001.

[6] X. Fu, T. Bultan, R. Hull, and J. Su. Verification of Vortex workflows. In *Proc. of Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2001. to appear.

[7] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig. Crossflow: Cross-organizational workflow management in dynamic virtual enterprises. *Int. Journal of Computer Systems Science & Engineering*, 15(5):277–290, September 2000.

[8] B. Gronemann, G. Joeris, S. Scheil, M. Steinfort, and H. Wache. Supporting cross-organizational engineering processes by distributed collaborative workflow management - The MOKASSIN approach. In *Proc. of 2nd Symposium on Concurrent Mulitdisciplinary Engineering (CME'99)*, Bremen, Germany, September 1999.

[9] O. M. Group. Workflow management facility, joint submission bom/98-06-07, revised submission, July 1998. ftp://-ftp.omg.org/pub/docs/bom/98-06-07.pdf.

[10] H. Hayami, M. Katsumata, M. Seko, K. Sagahashi, R. Shibuya, T. Ishimaru, M. Ohminami, and K. Okada. Interworkflow management system cooperating with commercial workflow management systems. *Special Issue on Knowledge and Information Sharing, Transactions of IPSJ*, 41(10), 2000.

[11] R. Hull, A. Kumar, and J. Siméon. Smart supply web: An application of web-based data and workflow mediation. In *Proc. of First Workshop on Technologies for E-Services*, September 2000. In cooperation with VLDB2000.

[12] R. Hull, F. Llirbat, E. Simon, J. Su, G. Dong, B. Kumar, and G. Zhou. Declarative workflows that support easy modification and dynamic browsing. In *Proc. of Intl. Joint Conf. on Work Activities Coordination and Collaboration (WACC)*, pages 69–78, February 1999.

[13] A. Kumar and L. Zhao. XRL: An extensible routing language for electronic applications. In *Intl. Conf. on Telecommunications and Electronic Commerce*, November, 1998.

[14] A. Lazcano, G. Alonso, H. Schuldt, and C. Schuler. The WISE approach to electronic commerce. *Int. Journal of Computer Systems Science & Engineering*, 15(5), September 2000.

[15] K. L. Myers and P. M. Berry. Workflow management systems: An AI perspective. Technical report, Artificial Intelligence Center, SRI International, December 23 1998.

[16] G. Riempp. *Wide Area Workflow Management: Creating Partnerships for the 21st Century*. Springer-Verlag, 1998.

[17] M. Shan. Flowjet: Internet-based e-service process management. In *Proc. of International Process Technology Workshop*, Villars de Lans, France, September 1999.

[18] W. van der Aalst and A. Kumar. XML based schema definition for support of inter-organizational workflow. Technical Report in preparation, CU Boulder, 2001.