Proc. Workshop on Cross-Organizational Workflow Management and Co-ordination, 1999

The Vortex Approach to Integration and Coordination of Workflows

(A position paper)

Richard Hull
Bell Laboratories, Lucent Technologies
600 Mountain Ave
Murray Hill, NJ 07974
hull@research.bell-labs.com

Jianwen Su*
Department of Computer Science
University of California
Santa Barbara, CA 93106
su@cs.ucsb.edu

February 10, 1999

1 Introduction

While many organizations use workflow management systems, few tools or techniques have been developed to support the interaction and coordination of workflows residing in different organizations. We distinguish two related aspects of such coordination, namely transactional (concerned with ensuring atomicity and durability of specified groups of actions) and semantic (concerned with the flow of information between, and intended meaning of, groups of actions performed by different workflows). This position paper identifies several issues that arise in connection with the semantic aspect of workflow coordination. The paper goes on to illustrate how a new declarative workflow paradigm called Vortex (see [HLS+99]) can be used to overcome these issues.

It is crucial to use a clearly defined framework and language for specifying workflow interaction. The framework must address the heterogeneity between workflow systems and schemas, and between data representations. The language must support data translations and restructurings, coordination of the input and output of existing workflows, and semantic enrichments and extensions of existing workflows. Furthermore, it is essential that the framework and language facilitate formal analysis of workflow interactions, such as whether data flow dependencies between data in different workflows can lead to deadlocks.

Vortex is a new programming paradigm for specific application domains, including in particular, workflow. The focus of Vortex programs is on information gathering, decision making, and launching and monitoring of external tasks. The Vortex language is declarative: programmers specify the conditions under which decisions and tasks should be performed, but

*Part of work was done while visiting the Bell Labs. and part of work was supported by NSF grants IRI-9411330 and IRI-9700370.

the flow of control is not specified explicitly. As a result, Vortex programs are more succinct and easier to analyze formally than equivalent procedural programs, and are easier for humans to understand and modify. Because the core semantics of Vortex is declarative, analysis is simpler than with procedural workflow languages. Section 2 highlights some key aspects of Vortex relevant to workflow coordination.

We propose in Subsection 3.1 an architecture for workflow coordination that is based on key elements of the Vortex paradigm. In Subections 3.2 and 3.3 we illustrate how this architecture can support a variety of crucial features for workflow coordination, including data translation, semantic enrichment of workflows, workflow integration, and mechanisms for querying workflow status and history.

2 Overview of Vortex

There are four fundamental aspects of Vortex that make it well-suited for workflow coordination: (a) the attribute-centric perspective, (b) enabling conditions, (c) snapshots, and (d) decision modules. This section presents very brief descriptions of these. The reader is referred to [HLS+99] for a more detailed introduction to Vortex.

Individual Vortex programs are centered around the decision making and processing needed to react to a single event, e.g., a new claim input to an insurance claims workflow, or a customer contact through a web-based storefront. Programmers specify what tasks might potentially be performed for an incoming event, and specify logical conditions under which these tasks should be performed. A typical Vortex application will involve several Vortex programs, each for dealing with a different class of events.

The Vortex language is *attribute-centric*, in the sense that programs in the language focus on how values should be assigned to the attributes of an input object. In particular, each workflow instance (i.e., enactment) begins with values only for the *source attributes*. As execution continues values for additional attributes are obtained, perhaps by computation or synthesis based on previously obtained attribute values, by information retrieval, or by interaction with humans or other workflows. Not all attributes need take values. Attributes can have atomic or complex type (based on tuple, list, etc.). External actions (e.g., issue checks) may be launched as a side-effect of attribute evaluation.

A Vortex program includes *enabling conditions* to determine whether an attribute will be evaluated for a particular workflow instance. These are reminiscent of the enabling rules in Meteor [KS95] with a crucial difference: the enabling rules of Meteor explicitly mention events, and can be fired only if the mentioned event(s) occur and the remainder of the condition is true at that time. Thus, an analysis of tasks and enabling rules in Meteor requires an understanding of the relative timing of tasks executions. In contrast, the assignment of attribute values in Vortex is monotonic (i.e., once assigned an attribute value cannot change), and the enabling conditions refer only to attribute values and states (i.e., whether an attribute has been enabled or disabled). This makes it possible for Vortex to have a simple declarative semantics that ignores issues around order of execution except for those implied by data flow constraints between tasks.

The attribute-centric paradigm and assumption of monotonicity make possible a natural mechanism for querying the status and history of workflow processing. This is based on the

notion of *snapshot* of a workflow instance. Suppose a workflow instance is being processed. At a given point in time the snapshot associated with this instance is a mapping from attributes to the current state of the attribute (not-yet-considered, enabled or disabled), and if the attribute is enabled either a value for the attribute or the "value" `uninitialized`. Once a workflow instance completes then its snapshot is archived. The set current and completed snapshots esssentially forms a relation with null values. Although snapshots appear to be a high-level abstract version of workflow audit data as specified in [WFMC98] for current and completed workflow instances, the snapshots can be queried using a relational query language such as SQL.

Finally we describe *decision modules*. These provide an eclectic mix of mechanisms for aggregating and synthesizing previously obtained attribute values. As a very simple illustration, suppose that multiple vendors are being considered in connection with a given purchase. Different factors might need to be weighed, e.g., the price quoted by different vendors, the availability date, previous history with the vendor, and etc. A Vortex decision module (or family of decision modules) might be used to associate a weight to each vendor and then pick the vendor with highest weight. As a simplified illustration, a rule such as "If vendor V gave lowest price quote, then contribute $[V, 10]$" can be used to contribute a weight of 10 in favor of V; and the total weight for V would be obtained by adding this with the other weights contributed for V. Vortex supports a broad family of simple and intricate semantics for combining the contributions of the rules in a decision module. Decision modules provide a simple mechanism for using a broad variety of heuristics when combining information. This appears to be useful in contexts that involve business considerations, e.g., customer care, automated resolution of exceptions, reconciling dirty data, and in particular, coordinating cross-organizational workflows.

3 Vortex-based Coordination Systems

This section proposes a conceptual architecture for workflow coordination that is based on the core ideas in Vortex, and then highlights some advantages of the approach. The discussion provided here a reflection of a preliminary report of the currently ongoing work on this topic.

3.1 Elements of the Architecture

A fundamental issue in developing workflow coordination systems lies in developing a "common" model that can provide a well-defined, unambiguous semantics for communication. Serious challenges arise in developing practical coordination, due in part to the specific semantics of workflow systems employed, heterogeneity in schemas of the databases involved, and semantic as well as structural differences in the data used. We provide here an exploratory discussion on key elements in a Vortex-based architecture that facilitates the development of coordination systems.

Our conceptual architecture (see Figure 1) is based on two key elements: (a) coordination interfaces that encapsulate participating workflows, and (b) a coordination system that supports controlled interaction between the workflows.

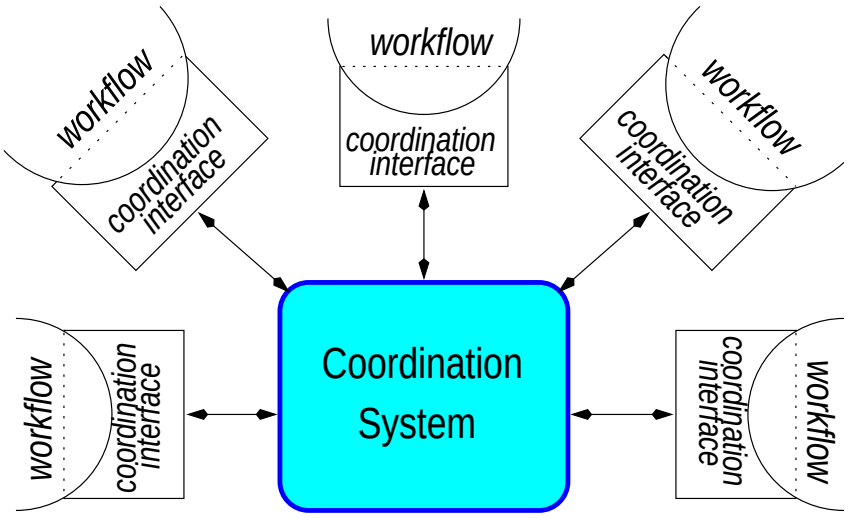

Figure 1: Conceptual Architecture

A *coordination system* implements a set of policies that govern the interaction between the participating workflows. The roles of a coordination system are to:

1. Translate of data produced by one workflow into one or more forms that are meaningful to other workflows.

2. Provide a "view" of one or several workflows, which may change or semantically enrich the external interfaces that the workflows provide.

3. Provide a mechanism for querying status and history of individual workflow instances.

4. Specify precisely (e.g., in a programming language) how attribute values should be collected/gathered.

5. Integrate/coordinate participating workflows so that the intended tasks are done in a proper manner.

This architecture is similar to the Federated Architecture for database interoperation [HM85]. A key difference is that the coordination system can support semantically rich manipulations of data, and can exercise discretion concerning how events are treated (see below).

A coordination system is *Vortex-based* if it uses Vortex as the internal language, and complies with the attribute-centric perspective. In this context, a *Vortex-based coordination interface* is provided for each workflow, and specifies the following:

1. *Export* attributes that can be used by other workflow systems through the coordination system;

2. *Import* attributes, which will obtain values through the coordination system from attributes from other workflow systems or other sources.

3. Data flow dependencies among import and export attributes of the workflow.

4. A set of queries that provide access to snapshots of current and completed workflow instances.

A Vortex-based coordination interface provides a specification of a workflow for the purpose of coordination. Its roles are reminiscent of that of abstract structures for handling datasets actually stored in many different data structures in computation modeling systems [SSE+95]. The snapshots includes the states and values (if applicable) of attributes.

The coordination system defines the process of acquiring values for the import attributes of the workflow systems, including how the values are computed and when. A Vortex-based coordination system will include a family of Vortex programs, each for responding to a different class of events, e.g., requests from one workflow for input from another workflow, new attribute values being exported by a workflow, and time-driven actions such as checking the status of a workflow instance and possibly requesting an escalation.

Our architecture makes no assumptions concerning how the workflows are interacting [WFMC96]. It may be that some workflows act as producers and others as consumers. Indeed, a coordination system could simulate the Action Workflow model [MWF92]. Alternatively, it may be that all of the workflows collaborate as equals. Also, it may be that some workflows are used exclusively as resources of the coordination system, i.e., they are invoked only by the coordination system. The architecture is shown from a high level; a coordination system might also use external databases and other software components to perform its job.

3.2 Using Vortex to Support Value-added Wrapping

A crucial functionality in supporting workflow coordination is that of "wrapping", or more specifically, translating and restructuring outputs of one workflow for use as input by another. Such translation is supported by Vortex-based coordination systems. Further, it is possible to provide semantic enrichment to a workflow as part of the wrapping process.

In this subsection we briefly describe four wrapping features supported by Vortex. These are illustrated using a simplified application environment, based on a mortgage application workflow (*MortW*). We have chosen this example because it is simple and familiar, even though it involves the interaction of a human with a workflow rather than the interaction of two or more workflows.

For the example, we assume that *MortW* takes in values including `SS#`, `employer`, `income`, `assets`, `debt`, and `address_of_house`. Under certain conditions additional data may be gathered from the applicant, e.g., if self-employed. The workflow will also interact with other systems, e.g., to access data from a credit reporting agency, and possibly with other workflows, e.g., to obtain the appraised value of the house.

Translations and data mappings. The first wrapping feature focuses on translating values and data structures as the pass from one workflow to another. One example is translating currencies, e.g., if a mortgage applicant is holding assets in foreign countries but the mortgage

workflow assumes only US currency. The translation might be based on a simple table look-up or multiplier, or it might incorporate factors such as the stability of the countries where the foreign assets are held. The translation process might rely on both internal and external data sources, or on interaction with an external workflow.

Suppose that *MortW* relies on applicant input concerning existing debt and repayment schedules. An intricate translation could be developed, so that this data can be gleaned from a credit reporting agency report rather than from the applicant. The translation would include restructuring and interpretation of data in various fields of the report. In some cases heuristics might be used to convert data from the report into a form suitable for *MortW*. Enabling conditions and decision modules are useful when heuristics are used in the conversion process.

Semantic enrichment. A natural extension of the data translation feature just mentioned is semantic enrichment to a participating workflow. Suppose that the bank wishes to streamline processing for applications involving houses in certain neighborhoods. The coordination system could identify these applications and present to *MortW* credit data from a credit reporting agency for these applications (whereas other applicants would have to supply their own credit data). Another semantic enrichment would be to augment the integrity checking occurring within *MortW*. For example, additional checks or restrictions might be applied to applicants whose credit card debt has been high for more than 4 years, even though *MortW* looks only at the current credit card debt.

Flexibility in information gathering. As explained above, Vortex workflows focus on the collection of attribute values, and restrict the order of this collection only as needed to satisfy data input/output requirements. In some cases this can permit significant flexibility in the order of activities when an individual or workflow is interacting with another workflow. To illustrate, suppose that *MortW* initially asks the applicant for 30 values, and may come back for additional values depending on the individual situation. If *MortW* is a Vortex workflow, or if the coordination interface is able to hide irrelevant ordering relationships between data requests, then the applicant will have the option of providing input values singly, rather than as a complete group. In particular, this may allow *MortW* to start working sooner, and to reduce the overall response time for the workflow instance.

Furthermore, it may arise that some requested values are determined to be unneeded as the result of working with already provided values. As a simple example, the applicant may have provided information on income, assets and credit card debt, but not on automobile debt. If the credit card debt is high enough to yield a denial, then the workflow need not wait for the automobile debt before issuing its response.

Informative workflow status and historical reports. The focus of Vortex on attribute values also offers the possibility of highly informative status reports, that focus on the current "snapshot" of the workflow instance. This snapshot will show what data has been gathered, what decisions have been made, and what additional data and processing must be performed. Ordering relationships not implied by data flow can largely be ignored in such reports. More generally, the set of current and completed snapshots can be represented as a relation (a set of tuples) that permits the status checking using SQL queries.

3.3 Workflow interoperation

The use of a Vortex-based coordination system offers tremendous flexibility with regards to managing the interoperation between workflows. As indicated above, the workflows might work in a consumer-producer model, or collaborate as equals. Furthermore, the Vortex programs in the coordination system can support highly differentiated responses to incoming events. For example, the status of a workflow can be checked at different times, and the response to a workflow that is delayed could vary based on numerous factors.

We now mention some additional benefits of using a Vortex-based coordination system.

Analysis of data flow. One issue in integration is to avoid situations that two or more workflows wait for the completion of evaluating attributes from each other in a deadlock situation. With a Vortex-based coordination system, tools can be developed for static analysis using graph- and logic-oriented techniques (e.g., see [HLS+98]).

Integration. A Vortex-based coordination system offers several functionalities useful in connection with supporting a unified interface against two or more workflows. To illustrate, we suppose that the bank using *MortW* also offers homeowner's insurance using a workflow called *InsW*, with input values included `SS#`, `employer`, `value_of_house`, and `value_of_belongings`.

Suppose that an individual is applying for both a mortgage (against *MortW*) and a homeowner's insurance policy (against *InsW*) for this home. The simplest functionality supports the sharing of input values, e.g., the applicant can provide shared attributes such as `SS#`, `employer`, and `address_of_house` just once. These input values can be translated if necessary.

Another functionality is to use values computed by one workflow as input to another. For example, in normal operation *InsW* may need the appraised value of the house. When being run in conjunction with *MortW*, the appraised value can be obtained from *MortW*. This kind of interaction is especially convenient in the context of Vortex workflows, which are flexible with regards to the order in which input values are provided.

More sophisticated integration can be achieved by incorporating semantically enriched wrapping. Suppose that *InsW* has an integrity check on the relationship between the `value_of_house` and the `value_of_belongings` to be insured. If the two workflows are run together, then a more sophisticated check can be made, that incorporates information about `assets`, `debt`, etc. This check can be created as part of the coordination system, and would not require changes to the underlying workflows.

References

[HLS+98] R. Hull, F. Llirbat, J. Su, G. Dong, B. Kumar, G. Zhou. "Adaptive Execution of Workflow: Analysis and Optimization." Bell Labs Technical Report, November, 1998.

[HLS+99] R. Hull, F. Llirbat, E. Simon, J. Su, G. Dong, B. Kumar, G. Zhou. "Declarative Workflows that Support Easy Modification and Dynamic Browsing." In *Proc. Int. Joint Conf. on Work Activities Coordination and Collaboration, 1999.*

[HM85] D. Heimbigner and D. McLeod. "A Federated Architecture for Information Management." *ACM Transactions on Office and Information Systems*, 3(3), 1985.

[KS95] N. Krishnakumar and A. Sheth. "Managing Heterogeneous Multi-Systems Tasks to Support Enterprise-Wide Operations." *Distributed and Parallel Databases*, 3(2), 1995.

[MWF92] R. Medina-Mora, H. Wong, P. Flores. "The ActionWorkflow as the Enterprise Integration Technology." In *Proc. of ACM Conf. on Computer Supported Collaborative Work*, 1992.

[SSE+95] T. R. Smith, J. Su, A. El Abbadi, D. Agrawal, G. Alonso, and A. Saran. "Computational Modeling Systems." *Information Systems*, 20(2):127-153, 1995.

[WFMC96] *The Workflow Management Coalition Workflow Standard – Interoperability Abstract Specification.* Document Number WFMC-TC-1012, 1996. `http://www.aiim.org/wfmc/standards/docs/if4-a.pdf`.

[WFMC98] *The Workflow Management Coalition Audit Data Specification.* Document Number WFMC-TC-1015, 1998 `http://www.aiim.org/wfmc/standards/docs/if5v11b.PDF`.