CONFLuEnCE: CONtinuous workFLow ExeCution Engine [1]

Panayiotis Neophytou, Panos K. Chrysanthis, Alexandros Labrinidis
Department of Computer Science, University of Pittsburgh
Pittsburgh, PA, USA
{panickos, panos, labrinid}@cs.pitt.edu

ABSTRACT

Traditional workflow enactment systems view a workflow as a one-time interaction with various data sources, executing a series of steps once, whenever the workflow results are requested. The fundamental underlying assumption has been that data sources are passive and all interactions are structured along the request/reply (query) model. Hence, traditional Workflow Management Systems cannot effectively support business or scientific reactive applications that require the processing of continuous data streams.

In this demo, we will present our prototype which transforms workflow execution from the traditional step-wise workflow execution model to a continuous execution model, in order to handle data streams published and delivered asynchronously from multiple sources. We will demonstrate a supply chain management scenario which takes advantage of our continuous execution model to enable on-line interaction between different user roles as well as streaming data coming from various sources.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Scientific databases*; H.4.1 [**INFORMATION SYSTEMS APPLICATIONS**]: Office Automation—*Workflow management*

General Terms

Algorithms, Design, Languages

1. INTRODUCTION

Many enterprises use workflows to automate their operations and integrate their information systems and human resources. Workflows have also been used to facilitate outsourcing or collaboration beyond the boundaries of a single enterprise, for example, in establishing Virtual Enterprises [1]. Recently, workflows have been used in the context of scientific exploration and discovery to automate repetitive, complex and distributed scientific computations that often require the collaboration of multiple scientists [4, 6].

A common class of applications in both business and scientific domains is monitoring and reactive applications that involve the processing of continuous streams of data (updates) [2]. Examples include financial analysis applications that monitor streams of stock data to support decision making in brokerage firms and sky monitoring applications that collect and analyze telescope images and metadata in real time for the detection of astronomical objects and events. The use of Continuous Queries [2] (CQs) is one of the current approaches in monitoring data streams. The drawbacks of CQs are: (1) they are stateless, (2) have a static configuration and (3) are unable to facilitate user interaction. This makes CQs unsuitable as a complete solution for enabling reactive applications.

Most recent workflow enactment/management systems orchestrate the interactions among activities within a workflow using web services [9]. Several business process modeling languages have been designed to capture the logic of a composite web service, in the form of a workflow, including WSCI, BPML, BPSS, XPDL and WS-BPEL 2.0. However, these interactions are usually one-shot interactions between the sender and the receiver of the request and it is clear that the existing workflow management systems and languages are not suited for reactive applications.

In our previous work in [5] we have examined the capability of current workflow models and workflow management systems to support business and scientific reactive applications. We based our examination on the Workflow Pattern framework [8]. This framework proposed a set of 20 common workflow patterns. An additional set of 6 communication patterns were proposed in [7]. This framework was used to evaluate the capabilities of some of the languages mentioned above ([10]), showing that these languages could not support nearly half of the 20 workflow patterns, and also two of the communication patterns. These two communication patterns are Publish/Subscribe and Broadcast which, interestingly, are essential for enabling reactive applications.

The lack of support of these two communication patterns in existing workflow models are a direct result of the assumption that data sources in workflows are passive (e.g., stored in databases or data files) whereas data consumers (users, tasks) are both active and passive. These missing communication patterns assume that some data sources are active, supporting continuous data streams.

In order to address the lack of support for continuous data streams in existing workflow models, we proposed a shift towards the idea of "continuous" workflows. The main difference between traditional and continuous workflows is that the latter are continuously (i.e., always) active and continuously reacting on internal streams of events and external streams of updates from multiple sources, at the same time and in any part of the workflow network. We have shown in [5] how the workflow and communications patterns are cast into continuous workflows (CWfs) and also proposed four new CWfs patterns.

We have implemented our proposed CWf model as a prototype system, called CONFLuEnCE, which is short for CONtinuous workFLow ExeCution Engine, and was built on top of Kepler [4]. In

[1]This research was supported in part by NSF grant IIS-053453, NSF career award IIS-0746696 and NSF grant OIA-1028162.

SIGMOD'11, June 12–16, 2011, Athens, Greece.
ACM 978-1-4503-0661-4/11/06.

this demo we will show how our prototype enables reactive applications. CONFLuEnCE can facilitate both scientific and business reactive applications. We will demonstrate a business reactive application for a Supply Chain Management (SCM) System, which falls in a widely known domain that is easier for the audience to relate to. We will show how different roles of users, such as the clients, warehouse operators, and management personnel can interact with the workflow while it is running, both by providing inputs and outputs as well as by manipulating various parameters. SCM design patterns [3] have been reformulated, fitting naturally into the CWf domain and have been used in the design of our example. Additionally we will show an example of how CWfs can be used by astronomers to monitor sky observations in a collaborative manner.

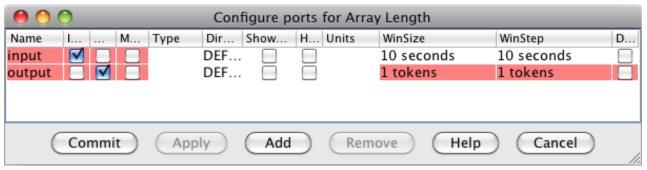
In the next sections we will provide some details on the CWf model and then describe how we implemented our prototype on top of the Kepler workflow system. Finally, we will describe the proposed demonstration scenarios.

2. CONTINUOUS WORFLOW MODEL

A "Continuous Workflow", is a workflow that supports enactment on multiple streams of data, by parallelizing the flow of data and its processing into various parts of the workflow. Continuous workflows can potentially run for an unlimited amount of time, constantly monitoring data streams. To achieve that, our Continuous Workflow Model that we have proposed in [5], introduced:

- Concurrent execution of sequential activities, in a pipelined way.
- Active queues on the inputs of activities which support windows and window functions to allow the definition of synchronization semantics among multiple data streams.
- Interactions between pipeline steps. That is, the ability to notify a downstream or upstream activity of an update and cancel its execution (i.e. invalidation of an earlier event, or future events)

A *window* is generally considered as a mechanism for setting flexible bounds on an unbounded stream in order to fetch a finite, yet ever-changing set of events, which may be regarded as a temporary bundle of events. We have introduced the notion of windows on the queues of events which are attached to the activity inputs. The windows are calculated by a window operator running on the queue. Windows are defined in terms of an upper bound, lower bound, extend, and mode of adjustment as time advances. The upper and lower bounds are the timestamps of the events at the beginning and the end of the window. The extend is the *size* of the window. This can be defined in three measurement units: (a) *Logical units*, which are time-based, and define the maximum time interval between the upper and lower bound timestamps. (b) *Physical units*, which are count-based, and define the number of events between the upper and lower bounds. (c) *Wave-based*, where the upper and lower bounds of a window are defined by the first and last events of a wave (as defined in [5]) currently being processed. If a wave has been processed then subsequent events of that wave are deleted. The mode of adjustment, also known as the *window step*, defines the period for updating the window. If a step is not defined, then the window is evaluated every time a new event comes into the queue. An activity is fired, when its preconditions are satisfied. The precondition may include the state of the current window at the queue. A flag called "delete_used_events" is also defined to denote if events that were used in the window that triggered the firing of an activity should be deleted from the queue. The signal to delete used events from queues comes as part of the post-conditions of an activity.


Configure ports for Array Length

Name	I...	...	M...	Type	Dir...	Show...	H...	Units	WinSize	WinStep	D...
input	✓	☐	☐		DEF...	☐	☐		10 seconds	10 seconds	☐
output	☐	✓	☐		DEF...	☐	☐		1 tokens	1 tokens	☐

Figure 1: Modified Kepler form for configuring actor ports. On this form the workflow designer can define in freeform text the size and step of the window associated with specific ports. Shaded cells denote non-editable parameters.

3. ARCHITECTURE & IMPLEMENTATION

In order to implement our continuous workflow model, we have to implement all three requirements presented in the previous section. Instead of building a new system from scratch we evaluated a number of open sourced workflow systems such as [6, 4]. We chose Kepler [4] as the base for CONFLuEnCE because of our common aim to support scientific workflows and extensibility. Kepler is a free open source scientific workflow system, which was built on top of PtolemyII. The suitability of Kepler, for implementing our CWf model, comes from the underlying PtolemyII system: "the use of well defined models of computation that govern the interaction between components" [1]. Also, the fact that Kepler is being actively developed by nearly 20 different scientific projects, makes its code inherently extensible and it provides a large library of basic as well as specialized actors for easy reusability and composition of new applications. The library includes actors for database interfacing, data filtering, etc. and it is easily extensible to include domain specific actors such as annotations of astronomical objects.

A workflow in Kepler is viewed as a composition of independent components called *actors*. Communication between them happens through interfaces called *ports*, which are distinguished into input ports and output ports. The connection between two ports is called a *channel* and the receiving point of a channel has a *receiver*. Actors may also have parameters configuring and customizing their behavior. For example, a filtering actor may consume a stream of tokens from its input port and letting through only tokens satisfying a condition specified in one of the actor's parameters. The execution and communication model of the workflow is governed by the model of computation (MoC) defined by a *director* entity.

The first requirement of our CWf model is the concurrent execution of sequential activities, governed as mentioned earlier by the director entity. Currently, we achieve this by using existing execution models from Kepler (e.g., the Process Network model, where each actor is inside its own execution thread thus all actors run concurrently), adapted to specifically accommodate our model. We are also implementing more complex schedulers which optimize certain metrics (such as results response time), by making better resource allocation between actors, as compared to the operating system which is oblivious to these kind of metrics.

The second requirement is adding queues on the inputs of actors to buffer data. Although this is already implemented in certain MoCs in Kepler, window semantics on these queues do not exist in any MoC. We have implemented a new type of receiver which is associated with the directors which implement the window specifications. This new type of receiver defines windows by size and step. The unit of measurement of these two parameters can be token, time or wave. These parameters can be set in freeform text in the modified form which is provided by Kepler for configuring actor ports (Figure 1).

[1]http://ptolemy.eecs.berkeley.edu/objectives.htm

Finally the third requirement, cancelation of certain events, is achieved by connecting the actor which makes the cancellation decision with the actors which potentially carry the events to be canceled. This is done by forwarding special control tokens, containing the wave id of the events to be canceled, to the actors' receivers.

Another challenge for us was to keep our model backwards compatible with the existing library of actors, as much as possible. Since continuous workflows have the notion of timed events, we encapsulate each token within an event object which carries its timestamp (either the creation time or the time it entered the system), and its wave id. Since all current actors were implemented without being timestamp aware, they cannot output the timestamp of the events to the next receiving actor. To solve this problem the CWf enabled director maps a time-keeper object to each actor at initialization time.

4. DEMONSTRATION

In our demonstration we would like our audience to interact with the CONFLuEnCE system both as users of a continuous workflow, and as continuous workflow designers.

For the first type of interaction, we have implemented a Supply Chain Management Application on top of our CWf platform. The users of this system are split into four categories: (1) Clients, (2) Warehouse manager, (3) Company Manager and (4) Administrator. Roles 1-3 interact with the workflow through a web interface (through a mobile device or a laptop) and the administrator interacts with the workflow directly through the Kepler interface. We will deploy a wireless hotspot to let users participate with their web-enabled cell phones. A client submits orders with multiple items and receives a notification once her order has been shipped. A warehouse manager notifies the system when an item is out of stock and also receives order requests from the system. Note that an order may contain objects that are available in different warehouses. The company manager receives notifications when things go wrong more than once and in more than one way, e.g., when an item is reported out of stock more than once in some specified period, or when multiple orders have been delayed or canceled. The administrator's role is to change parameters, such as window sizes, or tune up settings in the scheduler to make the execution fit the application's requirements. The execution of this scenario should take less than 5 minutes to demonstrate all the features. In case of shortage of participants we will run simulated user roles for clients and members of our team will assume the remaining roles. A high level depiction of the workflow design is shown in Figure 2. The continuous workflow generally serves as an integration layer between the databases in the warehouses, the web server providing the user interface and ordering system, and other administrator interfaces.

In addition to the aforementioned example, we would also like to show how our system can be used in scientific discovery by astronomers, to monitor transient astronomical events, as well as monitoring the discovery and classification of new objects in the sky by other astronomers around the world. We will demonstrate how two workflows communicate with each other to keep users informed about transient events. The first workflow is detecting system-wide transient events. The second one, setup by an astronomer monitoring a specific area in the sky, is correlating these events with the observations made by other astronomers and rates the significance of each event. In this case, a messaging system is aggregating comments by astronomers around the world on their observations about the transient events.

To demonstrate the CWf designer interaction, the audience will be given a set of predefined actors suitable for data manipulation, user communication and data stream sources (such as the Twitter stream). They will then be guided through the process of putting the actors together in a manner of their choice to perform a complex task and produce some kind of results based on the input from the stream sources. The user can then run the workflow and watch in real time the kind of results produced. As an oversimplified example consider the case where the user would like to detect soccer player injuries around the world in real time (rather than transient events as in our previous application). These could be reported as discrete events via email messages, or visualized through the user interface, aggregated as a graph. To do this the user would use a *Twitter Source* actor with a query such as "#soccer injured". Then connect it to a *Group By* actor with a window size of one minute and step of one minute, which creates a wave for each soccer match discovered. Then a filter actor would measure the significance of the report (more people reporting on the event makes it more likely that the event is really happening), and then the final set of actors will notify the user some way (e.g., email, SMS, console etc).

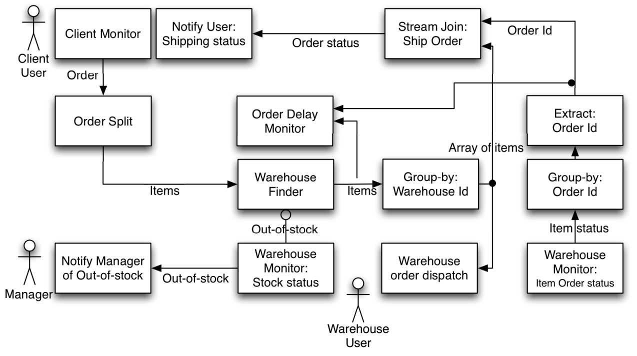


Figure 2: Demonstration Scenario: A continuous workflow for Supply Chain Management reactive application.

5. REFERENCES

[1] A. Berfield, P. K. Chrysanthis, I. Tsamardinos, M. E. Pollack, and S. Banerjee. A scheme for integrating e-services in establishing virtual enterprises. In *RIDE*, pages 134–142, 2002.

[2] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams: A new class of data management applications. In *VLDB*, 2002.

[3] R. Liu, A. Kumar, and W. M. P. van der Aalst. A formal modeling approach for supply chain event management. *Decision Support Systems*, 43(3):761–778, 2007.

[4] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[5] P. Neophytou, P. K. Chrysanthis, and A. Labrinidis. Towards continuous workflow enactment systems. In *CollaborateCom'08*, pages 162–178, 2008.

[6] T. M. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, R. M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[7] W. A. Ruh, F. X. Maginnis, and W. J. Brown. Enterprise application integration: A wiley tech brief, 2001.

[8] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[9] W3C. Web services glossary - http://www.w3.org/tr/ws-gloss/.

[10] P. Wohed, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede. Analysis of web services composition languages: The case of BPEL4WS. In *ER*, pages 200–215, 2003.