FlexFlow: Workflow for Interactive Internet Applications

Rakesh Mohan, Mitchell A. Cohen, Josef Schiefer
{rakeshm, macohen, josef.schiefer}@us.ibm.com

IBM T.J. Watson Research Center
PO Box 704
Yorktown Heights, NY 10598

Abstract

In this paper, we present a state-machine based workflow system, named FlexFlow, which formally describes business processes with state charts. The FlexFlow system uses these descriptions to directly control the execution of the applications. We give a description of the FlexFlow process model and the underlying FlexFlow engine, and explain how FlexFlow can be used in commercial platforms for B2B e-commerce.

Keywords: Interactive applications, e-business systems, workflow management, business process management, e-commerce platforms.

1. Introduction

The implementation of an e-commerce platform at a company often requires a customization of processes, such as an order process or a Request for Quotes, to the existing environment of that company. Workflow technology is prevalent for the modeling, analysis and execution of business processes [2][6].

In most current e-commerce systems, the steps of a business process, or the actions a system takes in response to user actions in such a process, are not made explicit, but are buried in a software code for both the dynamic pages and the application server. This makes the modification of implemented business processes extremely difficult and fragile. For example, to change the ordering of the process steps requires substantial rewriting of the software for the application and the web pages for the user interface. For e-commerce platforms made to be used by different companies, this presents a big problem as most companies' business processes differ from those of other companies to a small or large extent. Thus, deploying such e-commerce platforms at each different company incurs a large overhead in terms of time and money required to rewrite the business processes [1][7]. Often, this overhead actually forces companies to adjust their business processes to conform to an e-commerce system instead of modifying the system to match their preferred processes.

In this paper, we show how to employ the formal method of state charts [3][4][5] for the specification of processes for e-commerce platforms. By using state charts as our specification method, we are able to model business processes which can be automatically executed by a workflow engine. Our contribution is the introduction of process state diagrams, which use the state starts notation for modeling business processes. Furthermore, we introduce the FlexFlow system, which supports the formal specification of process state diagrams, including the simulation and execution of processes modeled with these diagrams.

2. FlexFlow – Overview

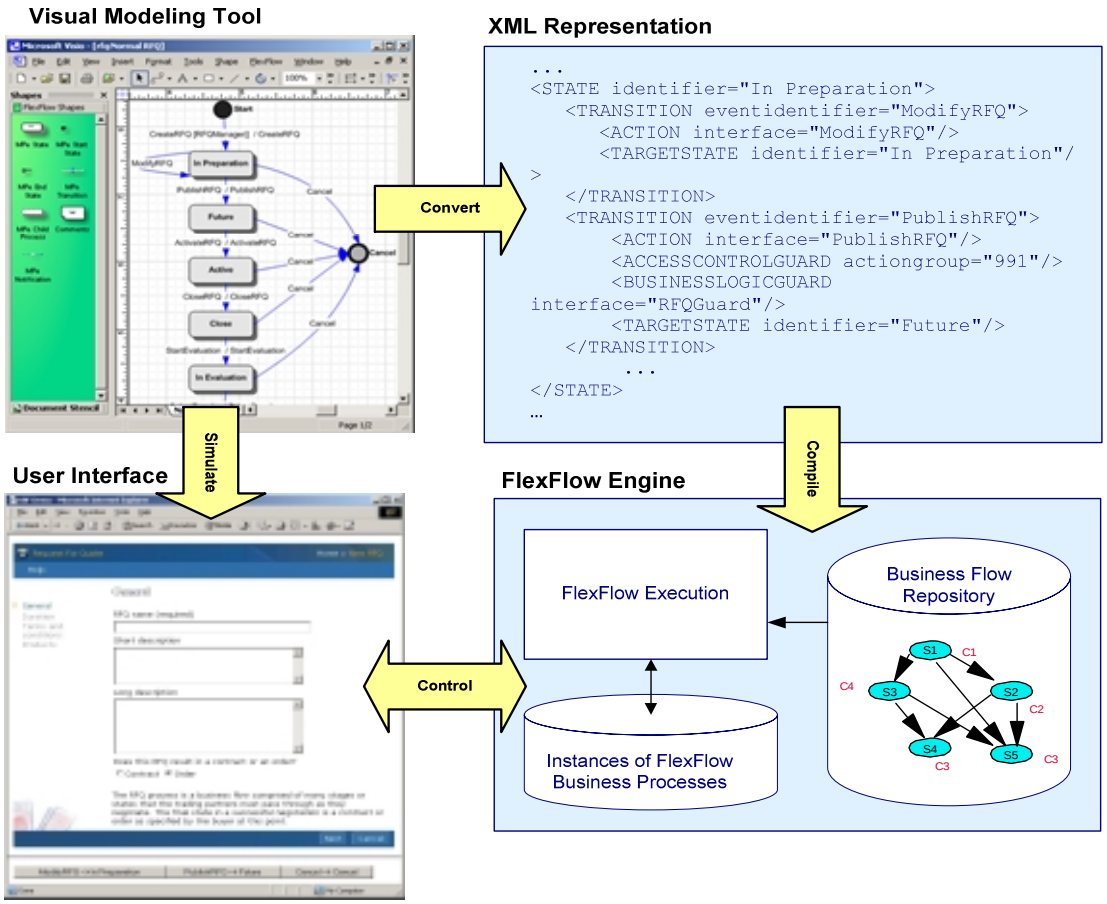

Figure 1: FlexFlow - Lifecycle of a Business Process

Figure 1 shows the lifecycle of business processes in the FlexFlow system. A visual modeling tool is used to design new and modify existing business processes. The visual modeling tool generates from the process state diagrams an XML representation. The XML is compiled and loaded into the FlexFlow system database. In this database, FlexFlow also tracks each instance of a business process running at a given time in the business system including the current state. The FlexFlow engine uses the database to control both the execution of the business process and the user interface.

3. The FlexFlow Process Model

FlexFlow models e-commerce business processes as Unified Modeling Language (UML) state diagrams [10], which are an adaptation of Harel's statecharts [3][5]. UML uses state diagrams to describe the behavior of objects, whereas, FlexFlow uses them to describe processes. We adopt the UML state diagram notation for the FlexFlow.

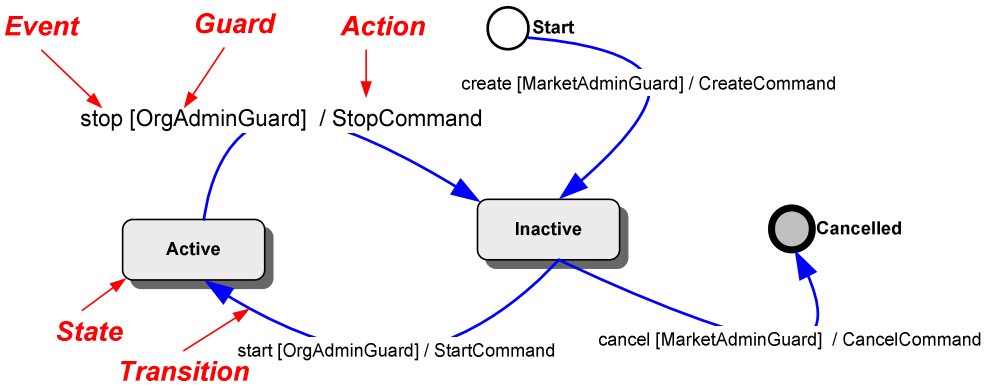

Figure 2: Sample FlexFlow State Diagram

UML state diagrams are directed graphs with nodes called *states* and the directed edges between them called *transitions* (see Figure 2). A transition represents a change of the process state, connecting a source state with a target state. A transition corresponds to an *action* taken in response to an *event*. The transitions have *guards* specifying under what conditions the transition can be traversed. Only one transition out of a state is taken in response to an event.

FlexFlow adds three key features beyond UML: 1) the concept of roles, 2) the coordination of interactions of multiple parties, and 3) the ability to allow different organizations to use different versions of the business process.

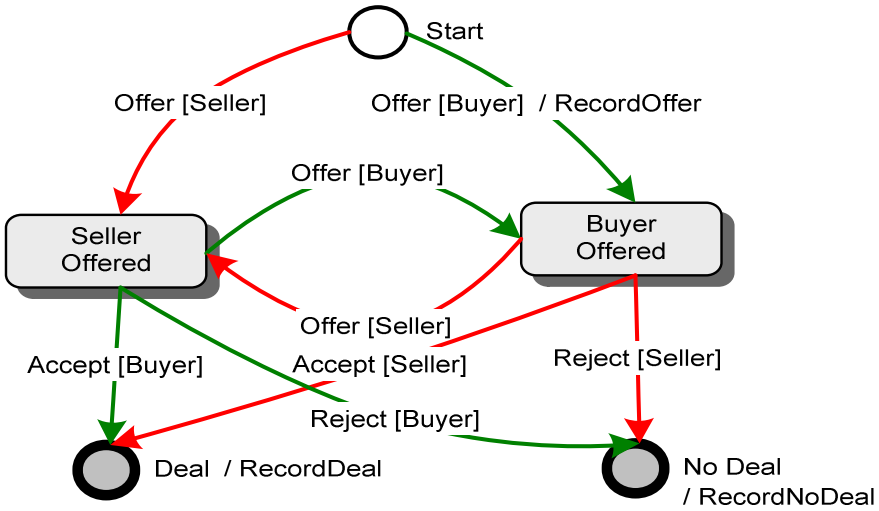

Figure 3: Simple FlexFlow state diagram for bilateral negotiation

For FlexFlow, events are incoming messages and actions correspond to task logic being executed at the application server.

Figure 3 shows a FlexFlow modeling of a simple negotiation between a buyer and a seller [8][9]. The top right transition shows that on the event "Offer", the action "RecordOffer" is taken. The engine checks the guard specifying that the user making the offer is the "Buyer". As the action for the other "Offer" transitions is also "RecordOffer" we do not show it here for the sake of simplicity. There is no action corresponding to the "Accept" or "Reject" events. On entry to the final state "Deal" a "RecordDeal" action is taken.

4. FlexFlow Process Execution

The FlexFlow system has an engine to manage the execution of business processes. When an event arrives, an event dispatcher figures out to which business process instances the event applies and invokes the engine for each. The engine processes the event based on the instance's current state and business process.

4.1. Event Creation

Events can be created in the following two ways (both shown in Figure 3):

- **Interaction Controllers** handle external interactions with different types of clients including web browsers, mobile devices, and message queues. For example, a buyer requests on a web form for an RFQ to be closed. This HTTP request is received by the interaction controller which converts it into an event.
- **Internal System Actions** can trigger events. For example, the "RFQ close" action might create a "Close Quote" event for the responses to that for that RFQ.

4.2. The FlexFlow Event Handler

All events arrive first at the FlexFlow event handler (see Figure 3). Events triggered by interaction controllers simply get routed to the FlexFlow engine. Events triggered by existing process instances are routed to all process instances listening for it. For example, all the quotes listening to their parent RFQ need to process events coming from the RFQ. To determine which instances are listeners, the router reads a Flow Instance Event Registry where quote process instances are registered to listen to the RFQ process to which they belong. The FlexFlow event handler will duplicate the event for all the listeners, routing each to the Flex Flow engine.

4.3. The FlexFlow Engine

The FlexFlow Engine receives targeted events from the event handler and executes the necessary actions. Figure 4 shows how the engine interacts with the other parts of the system for event processing.

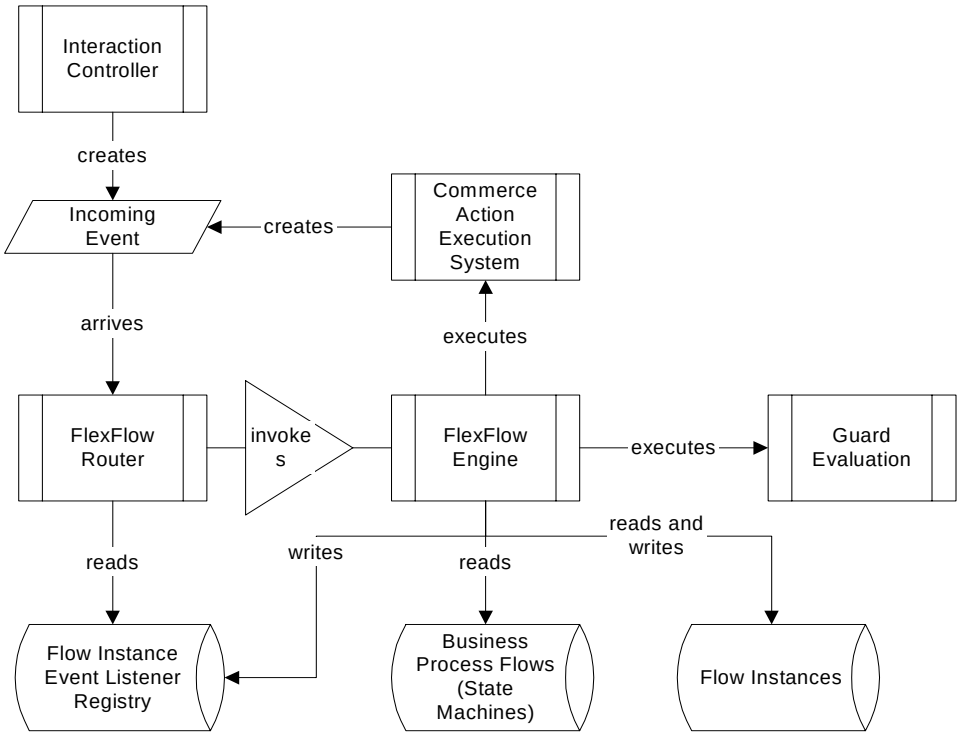

Figure 3: FlexFlow event handler and engine

When receiving an event, the FlexFlow Engine takes the following steps (as shown in Figure 4):

1. The incoming event is retrieved with its context including marshaling incoming parameters and deriving user and role information.
2. The engine either retrieves the existing targeted instance or creates an instance of the new specified process putting it in the start state. The engine registers the instance for events from those instances from which it needs to know outgoing messages such as a quote with its parent RFQ.
3. The engine looks for a transition that 1) exits the current state of the instance, 2) has an event matching the event being processed, and 3) has guards which can be satisfied; the engine calls the guard evaluation to check.
4. If no transitions were found in the previous step, then the engine returns control to the caller with a count of the number of transitions traversed and the list of the next available events. When appropriate, the caller will treat no transitions traversed as an error.
5. If the engine has come this far without returning, it has a transition that can be traversed. The first step is to execute the exit action on the current state if there is one.
6. The engine executes the action on the found transition.
7. The engine looks for an entry action on the transition's target state. If one exists, the engine executes it.
8. The engine updates the instance's current state to the transition's target state.
9. In order to process any automatic (null event) transitions exiting the new current state of the instance, the engine then sets the incoming event to null and returns to step 3.

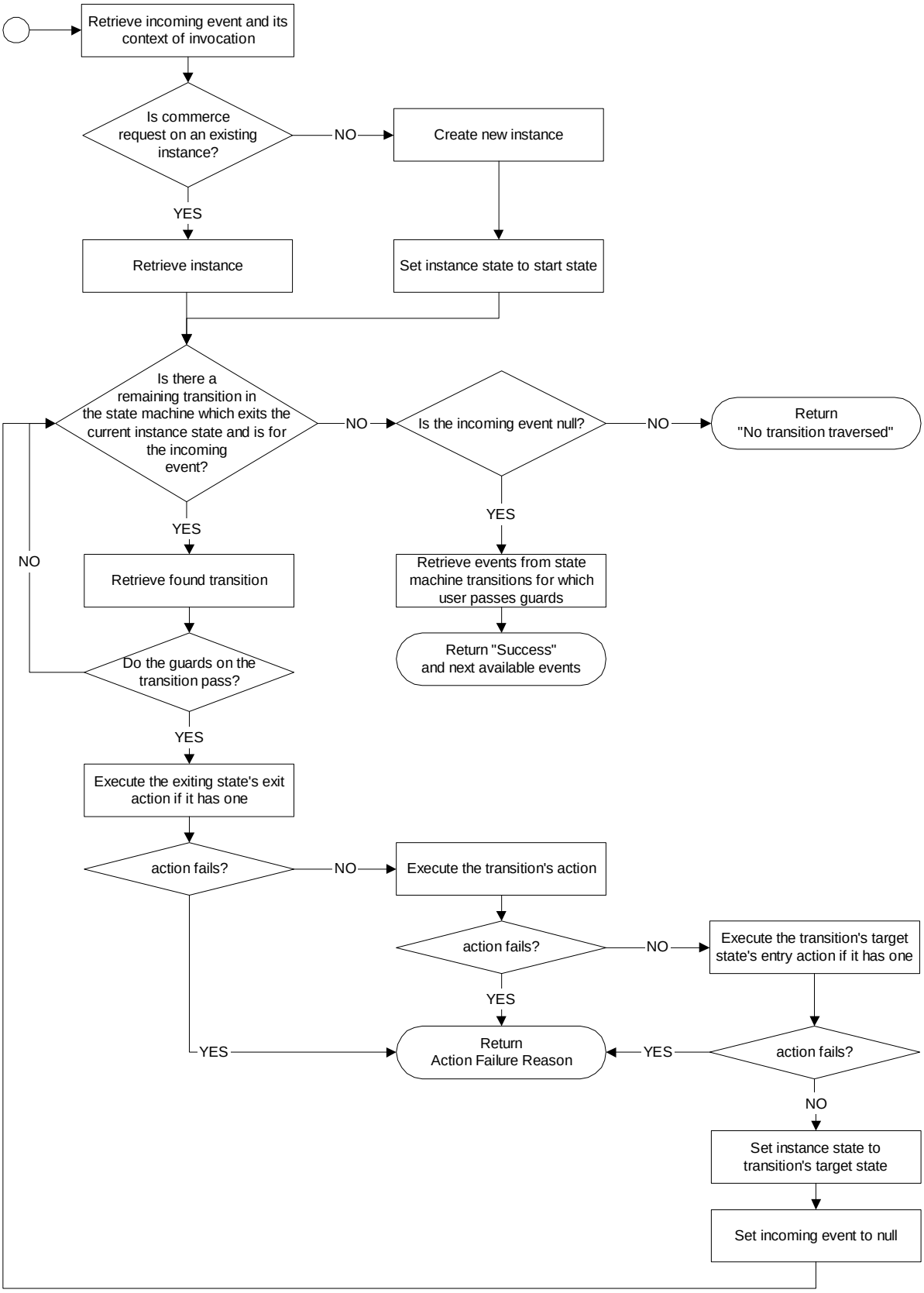

Figure 4: Flow chart showing engine execution of an incoming event

Note that if any of the actions (exit, transition, or entry) fail to complete successfully, the engine returns control to the caller with the reason for failure. The FlexFlow engine processes all actions from one event within a single transaction scope. In other words, the system is left with either the effect of all the actions executed, or none. If any action fails, the whole transaction is aborted and the effects of the previous actions whose execution was initially or subsequently caused by the incoming event are rolled back. Only when all the actions succeed is the transaction committed. This prevents the process instance from ending up in an "unnatural" state.

5. Conclusions

Web-applications are difficult to build with traditional workflow management systems. In this paper, we presented an approach for managing web-based business processes. We proposed a state machine based model for the specification of business processes and have shown the FlexFlow system which supports the modeling, simulation and execution of process state machines. We have deployed two generations of the FlexFlow system in commercial B2B e-commerce systems, first in IBM's WebSphere Commerce Suite MarketPlace Edition ® (WCS MPe), and then in IBM's WebSphere Commerce Business Edition ® (WCS BE). Additional problems we want to consider in the future include the management of hierarchical states as well as the concurrent execution of FlexFlow processes.

References

[1] Ahmed K. Elmagarmid and Weimin Du. Workflow Management: State of the Art vs. State of the Market. In Advances in Workflow Management Systems and Interoperability, pages 1-17, August 1997.

[2] Georgakopolaus, Diimitiros and Hornik, Mark, An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, Distributed and Parallel Databases, 3, 119-153, 1995.

[3] Harel D., Statecharts: A Visual Formalism for Complex Systems, Science of Computer Programming, Vol. 8, 1987.

[4] D. Harel et. al., "STATEMATE: A Development Environment for Complex Reactive Systems," IEEE Transactions on Software Engineering, April 1990.

[5] Harel, D., On Visual Formalisms, Communications of the ACM Vol.31 No.5, 1988

[6] Mohan, C., Recent Trends in Workflow Management Products, Standards and Research, NATO, 1997.

[7] Peter Muth, Jeanine Weissenfels and Gerhard Weikum, What Workflow Technology can do for Electronic Commerce, Current Trends in Database Technology, Idea Group Publishing, 1998

[8] Reuter, A., Schwenkreis, F., ConTracts - A Low-Level Mechanism for Building General-Purpose Workflow Management Systems, IEEE Computer Society, Bulletin of the Technical Committee on Data Engineering, 18(1):4-10, 1995

[9] J. Sprinkle, C.P. van Buskirk and G. Karsai, Modeling Agent Negotiation, Proceedings of the IEEE Systems, Man, and Cybernetics Conference, October 2000.

[10] Unified Modeling Language Specification, version 1.4, http://www.omg.org/technology/documents/formal/uml.htm, 2001