RC22327 (W0202-020) February 7, 2002
Computer Science

IBM Research Report

A State Machine Based Approach for a Process Driven Development of Web-Applications

Rakesh Mohan, Mitchell A. Cohen, Josef Schiefer
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

A State Machine Based Approach for a Process Driven Development of Web-Applications

Rakesh Mohan, Mitchell A. Cohen, Josef Schiefer

IBM Watson Research Center
PO Box 704, Yorktown Heights, NY 10598
{rakeshm, macohen,josef.schiefer}@us.ibm.com

Abstract. Traditional workflow systems are not suited for highly interactive online systems. We present a state machine based workflow system, named FlexFlow, which formally describes Internet applications using statecharts. The FlexFlow engine uses these descriptions to directly control the execution of web applications. FlexFlow helps in generating controls for user interactions on web pages. Different versions of an application can be generated by visually editing its FlexFlow description, with minimal incremental effort in rewriting application code or related web pages. FlexFlow provides an efficient way to customize online systems and supports different versions of business processes in the same e-business system for different sets of industries, organizations, users, or devices. We demonstrate FlexFlow's use for rapid prototyping of business processes and describe how we have used FlexFlow in commercial platforms for B2B e-commerce.

1 Introduction

In business systems, abstraction of the process logic from the embedded task logic enables the business processes to be modified independent of application code. 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 [4], [13].

Business process management is critical in a three- or multi-tier environment of e-business systems. Business rules and process information is extracted from the business logic tier and is presented in a workflow-based environment, which manages the execution of the business processes. Consequently, this approach greatly simplifies the application logic at each step. Business rules become explicit, visible, and rapidly changeable. System changes are stimulated and can be easily communicated between the development team and the business, and between the business and its partners, i.e., the customers and suppliers.

Business processes vary with a company's business model, and its industry sector. In e-commerce systems, trading mechanisms, such as auctions and negotiations are varied to suit particular business partners, product categories or market conditions. Business processes are customized to the role of the user and the terms and conditions of a contract with the user's organization. For example, the registration process for an administrator may be different from that of a buyer, and whether payment precedes or follows order confirmation may depend on the terms of a contract. E-commerce platforms thus need to provide both an easy way to modify business processes and to maintain variations of business processes. The separation of process and task logic allows both the easy customization of business process and reuse of the task logic in the variants of a business process.

In most current e-commerce systems, the steps of a business process, or the actions a system takes in response to user requests, are not made explicit, but are buried in 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, deployment of such e-commerce platforms incurs a large overhead in terms of time and money required to rewrite the business processes. 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 present a state machine based approach for managing web-based business processes that is more suited for the interactive nature of online systems than traditional workflow systems. We introduce a system which facilitates communication about system change with a descriptive model in which "as-is" and "as-to-be" models represent business processes. Since the e-business environment is so dynamic, change often overtakes models before delivering any significant results. Business people, rather than information technology experts, must be able to develop and extend the business process model. Hence, tools are required that facilitate business experts in communicating their vision and insights via a descriptive model.

Additionally, we show how to employ the formal method of statecharts [6], [7] for the specification of processes for e-commerce platforms. By using statecharts 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 statechart 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. FlexFlow is suited for interactive applications and is lightweight. It uses state machines to (a) describe the actions that can be taken by a particular user at particular points in a process based on the role of the user, (b) to enforce the validity of user requests, (c) to track the execution of actions within an instance of the business process, (d) to provide the user interface with a list of actions available to a user working on an instance of the business process, (e) to provide coordination between state machines, and (f) to allow different organizations to have varied business processes.

First, we discuss existing related work. Then, we give an overview of the FlexFlow system, introduce the FlexFlow process model, and explain how defining business processes with FlexFlow can drive e-commerce development. We wrap up with our real world experiences using FlexFlow and where we can go with it next.

2 Related Works

Business Process (Re-)Engineering [5] is an important driving force for workflow management. It aims to make business processes more efficient and quickly adjustable to the ever-changing needs of customers. In contrast to specifications of business processes, workflow specifications serve as a basis for the largely automated execution of processes. Workflow specifications are often derived from business process specifications by refining the business process specification into a more detailed and more concrete form. Automated and computer-assisted execution means that a workflow management system (WfMS) [4], [9], [12] controls the processing of activities, which have to be performed in the workflow. Some activities may have a manual or intellectual part, to be performed by a human. But the workflow management system is in charge of determining the (partial) invocation order of these activities. In contrast to business process specifications, this requires a formal specification of control flow and data flow.

Workflow specifications based on script languages contain control flow and data flow constructs which are specifically tailored to workflow applications. Such script languages are popular in current WfMS products. They provide a compact representation making them easy to use. A drawback of most script languages is their lack of a formal foundation. Their semantics is mostly ’defined’ by the code of the script interpreter used.

Leymann argues in [11] that state transition nets are a good choice when a graphical visualization of workflow specifications has high priority. In state transition nets, activities are represented by nodes, and control flow is represented by edges. In fact, almost all WfMS products provide means for graphical specifications similar to state transition nets.

Considering only net-based methods with a formal foundation, we have to restrict ourselves more or less to statecharts [6] and Petri nets [3], [15]. Variants of Petri nets, especially predicate transition nets, are used in a number of research prototypes as well as in several WfMS products [2], [14]. Some workflow management systems use variants of Petri nets for the internal representation of the workflow engine, e.g., [16]. Statecharts [6], [7] have received little attention in workflow management, but they are well established in software engineering, especially for specifying reactive systems. In the MENTOR project [21], statecharts are used as a formal foundation for workflow specification.

Event-Condition-Action-Rules (ECA) rules are used in active database systems and have been adopted by a number of projects in the workflow area (e.g., [10]). ECA rules are used to specify the control flow between activities. Like for other methods that are not based on nets, the graphical visualization of sets of ECA rules is a non-trivial task. Large sets of ECA rules are hard to handle, and a step-wise refinement is not supported [17]. In terms of their formal foundation, ECA rules are typically mapped to other specification methods, especially variants of Petri nets or temporal logic.

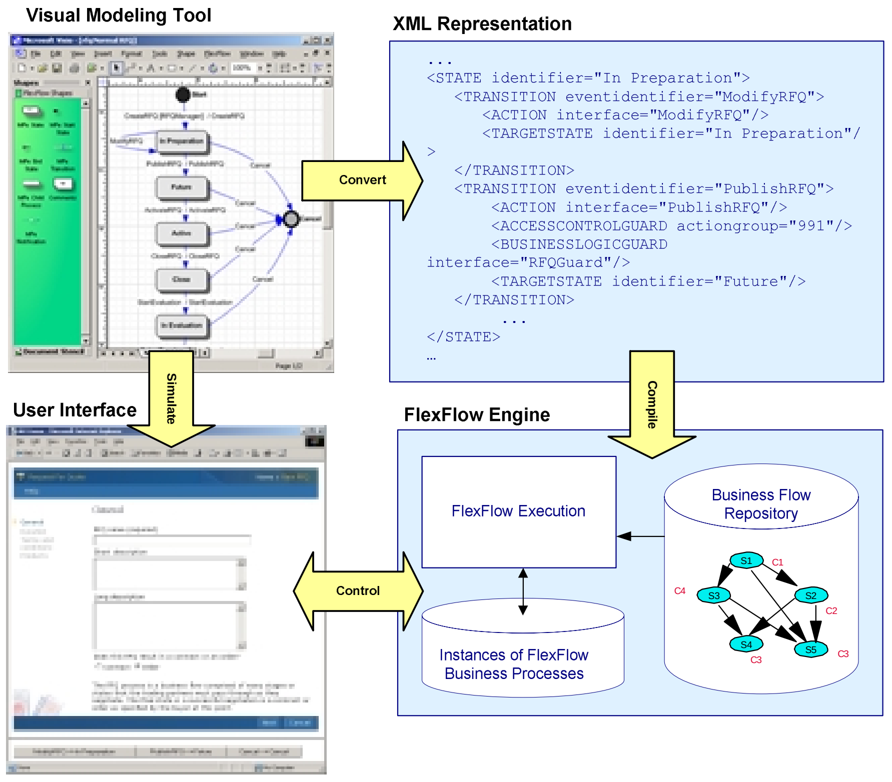
The pattern of user interaction with e-commerce business processes is very different from that of traditional workflow systems. Online business systems are highly interactive. Internet applications follow the request-response model. In online business systems, a user takes an action, such as clicking a submit button on a web page. This results in the form data on that page being sent to the system and the system acting on it and presenting another page to the user. For example, a user goes to a shopping web site, fills out the login page and clicks the submit button. This results in her user name and password being sent to the system, which authenticates the user and returns the catalog page. Then the system waits until the user selects products to fill the shopping basket. This interactive, conversational pattern of the system acting based on a user request and then waiting for the user to initiate the next step is not well modeled by existing workflow systems. This modeling difficulty is a major reason why online e-business applications do not use workflow systems.

Another problem is the complexity, cost, and size of workflows systems cause a high cost of deployment and limit the responsiveness when servicing a large number of concurrent requests. Microflows [1] have been proposed to address this drawback of workflow systems. Microflows are small footprint workflow systems crafted for a particular class of applications. They provide minimal or no support for services provided by full workflow systems such as transaction management, guaranteed messaging and worklists. Microflows provide the benefits of abstracting process logic from task logic while at the same time improving the responsiveness and reducing the cost as compared to industrial strength workflow systems.

State machines are widely used for implementation of network protocols to describe the conversation between a sender and receiver. Business processes for e-commerce platforms also interact frequently with each other. Examples are negotiation scenarios between buyers and sellers, where the current state determines the next available actions to each. State machines have been also used to model negotiations [18]. They have been used for real-time systems; a system reacts or responds to events with a quick, nearly instantaneous response [19]. Thus, there is strong evidence to support that state machines would be useful for interactive, conversational and responsive online business systems.

3 Flexflow Overview

Fig. 1 shows the lifecycle of business processes in the FlexFlow system. A visual modeling tool is used to design business processes as process state diagrams. The visual modeling tool generates from the process state diagrams an XML representation, which is a full description of the business process. It contains all the information required by the FlexFlow engine to control the execution of the business process. This XML description is compiled and loaded into the FlexFlow system database tables. An additional table is used to store the current state of each instance of a business process running at a given time in the business system. The FlexFlow



```
...
<STATE identifier="In Preparation">
   <TRANSITION eventidentifier="ModifyRFQ">
      <ACTION interface="ModifyRFQ"/>
      <TARGETSTATE identifier="In Preparation"/
>
   </TRANSITION>
   <TRANSITION eventidentifier="PublishRFQ">
       <ACTION interface="PublishRFQ"/>
       <ACCESSCONTROLGUARD actiongroup="991"/>
       <BUSINESSLOGICGUARD
interface="RFQGuard"/>
       <TARGETSTATE identifier="Future"/>
   </TRANSITION>
        ...
</STATE>
…
```

Fig. 1: FlexFlow - Lifecycle of a Business Process

engine uses these tables to control the execution of business processes as well as the user interface.

The visual modeling tool is also used to modify definitions of business processes. Different versions of the business process are stored in the business flow storage. Business processes can be changed with limited or no change to the task logic or computer programs that are implementations of the business actions. Simply reconfiguring its corresponding state machine reconfigures a commerce function.

The FlexFlow system also includes a component for simulating business processes. Thereby, users and developers can explore process variations to reach a common vision of how the user might interact with the system to perform a task.

4 The Flexflow Process Model

FlexFlow models e-commerce business processes as Unified Modeling Language (UML) state diagrams [20], which are an adaptation of Harel's statecharts [6], [7].

UML uses state diagrams to describe the behavior of objects, whereas, FlexFlow uses statecharts to describe processes. We adopt the UML state diagram notation for the FlexFlow.

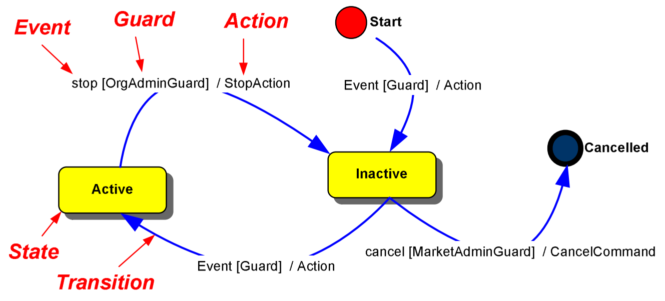


Fig. 2: FlexFlow State Diagrams

UML state diagrams are directed graphs with nodes called *states* and the directed edges between them called *transitions* (see Fig. 2). FlexFlow models interactive online business processes with these state diagrams. However, unlike UML, where state diagrams describe the behavior of objects, FlexFlow state diagrams describe processes. In addition to the functionality of Harel statecharts and UML state diagrams, FlexFlow adds three key features: 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. Business processes are versioned as different state diagrams. Versions can be selected based on membership at the organization level. Versions can also be selected based on other factors including the mode of interaction, such as device, browser, and messaging method. The UML notation used by FlexFlow consists of states, transitions, events, guards, and context.

Actions

Actions correspond to task logic being executed at the application server. For FlexFlow they are atomic units of business work. Actions can appear in states and transitions. An action can be used to interface to an external system, such as a workflow system handling its own set of functionality. An action can be a conglomeration, or sequence, of pre-defined internal commerce actions. All actions caused by the processing of an event are run within the same transaction.

States

States correspond to stages in a business process. A state identifies a precise point within the process. In a given business process at a given state, the actions that can be taken by various parties are completely defined by the set of outgoing transitions. A state may have an *entry action*, an action that is executed upon entering the state, and an *exit action*, an action that is executed upon leaving the state. In FlexFlow, *entry actions* are allowed to trigger new events, which in turn get processed by FlexFlow.

Transitions

A transition represents a change of the process state. It connects two states, a source state it exits and a target state it enters. A transition corresponds to an *action* that is taken in response to an *event*. Additionally, the transitions may have *guards* on them. These guards are checked, and the transition is taken only if they are true. Only one transition out of a state is taken in response to an event. In UML state diagrams, the actions on the transitions are assumed to be instantaneous. In FlexFlow most of the processing activity happens on the transition actions. Given the interactive nature of the applications, these usually take a very short time, but are not instantaneous.

Events

An event is a named message needing to get processed. In Internet applications, an event is usually an HTTP client request generated by a user pressing a hyperlink, button, etc. on a web page. It can also be an incoming Simple Object Access Protocol (SOAP) request or a Java Message Service (JMS) message. It can also be an event generated by another process such as a scheduler or another FlexFlow process. It can even be an event generated in the same FlexFlow process by a transition or a state entry action.

Guards

A guard is a set of conditions that need to be true before the action can be taken. Conditions are Boolean computations on the *context* of the business process and/or the parameters of the event. In general, the guards can be rules. In our implementation, an access control condition is always present in the guard. Thus, the action on a transition is taken only if access is allowed. If no access control policy is explicitly specified, the default access control mechanism is used.

Context

Context is data associated with a business process. It consists of

– The session information that includes information about the user including roles and permissions, and

– The data submitted by the user such as form entries and the data stored in the form such as the identification of the process and the business object. For example, if a user submits a bid for an auction, the context would contain the username and roles as well as the amount of the bid. The event would include an identifier for the auction on which the bid is made. Also included in the context is more general information about the process such as auction start and end time, number of bids, etc.

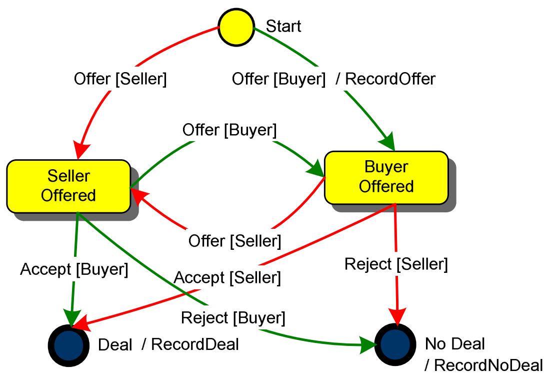


Fig. 3: A simple state diagram for bilateral negotiation

This context can be referenced in guards as well as read and updated in actions. Fig. 3 shows a FlexFlow model for a simple bilateral negotiation process between a buyer and a seller. The top right transition shows that on the event “Offer” the action “RecordOffer” is taken. The guard checks 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 simplicity. There is no action corresponding to the “Accept” or “Reject” events. On entry to the final state “Deal” a “RecordDeal” action is taken.

Fig. 4 shows two variations of the bilateral negotiation process shown in Fig 3. The process in Fig. 3 forces the buyer and seller to alternate their bids, i.e., once a participant makes an offer, she has to wait for the other party to make a counter offer. In Fig. 4 (a), the parties can improve their offers without waiting for a counter offer. In Fig. 4 (b), the parties can make a final offer which forces the other party to either accept of reject the offer but does not allow them to counter offer. As is obvious from the process diagrams, the three variants of the business process reuse the code for just three actions “RecordOffer”, “RecordDeal” and “RecordNoDeal”.

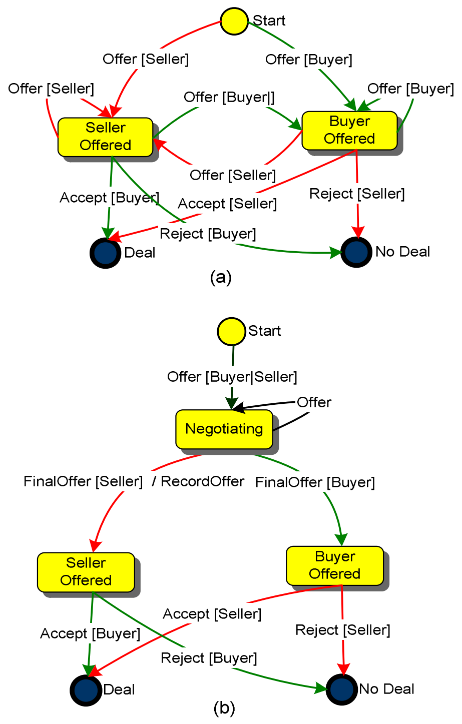


Fig. 4: Two variations of the bilateral negotiation process.

5 User Interaction with FlexFlow

We have observed that a common practice for designing web sites, such as e-commerce sites, is to first mock-up the flow of web pages for user interactions, and then to use this flow to drive the development of application logic. This practice works when the business process is simple and when only one party (the user) is interacting with the system. However, this design practice does not scale to complex business processes, especially where multiple parties are participating in the business process, such as two users in a bilateral negotiation or a buyer and multiple sellers in an RFQ, along with schedulers for timeouts etc. Another drawback of this design practice is that process logic gets embedded both in web pages and application code further complicating any modification of the business process.

Process Reflection

The FlexFlow process model has sufficient information for deriving user interactions from the state diagram. The process reflection mechanism of FlexFlow allows clients to discover or query process information at run time. This mechanism can be used to drive the user interface or the future user interaction. Thus, with FlexFlow, the design practice is to first design the process and then to automatically derive the flow of user interactions. As the user interaction information is added dynamically to the web pages at run time, the modifications of the business process get automatically reflected in the web pages.

Process reflection allows users to query a list of actions that are valid for a given user role at the current state of the business process. At each given state, the FlexFlow system knows the next possible set of actions a particular user can perform by using the guards on all the outgoing transitions. Thus, FlexFlow can provide relevant information for the rendering of the user output (i.e, FlexFlow can determine whether buttons should be enabled or disabled). If web designers use this reflection mechanism, web pages can be shared among different process versions. Using reflection also reduces the effort for modifying FlexFlow processes [8].

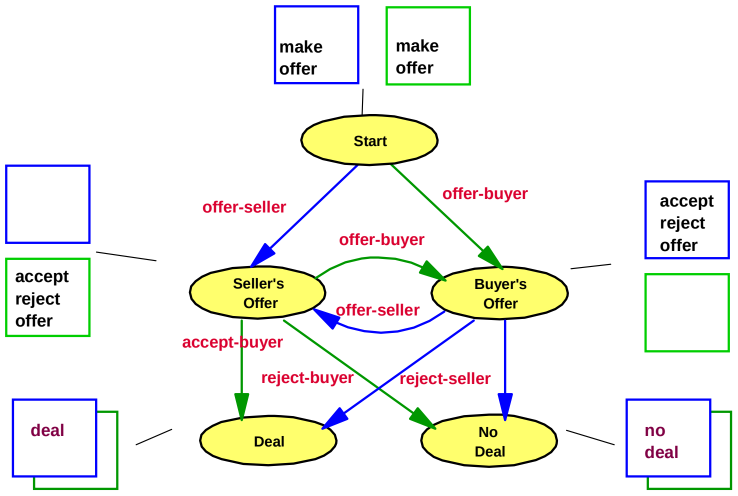


Fig. 5: Controls on web forms for user interactions are created using FlexFlow. The blue outlined page is for the seller, the green for the buyer. The text in black corresponds to buttons on the forms.

We illustrate this in Fig. 5 for a simple bilateral negotiation. There is web page for each state for each user where the user can take an action. The seller's pages are outlined in blue and the buyer's pages are outline in green. At the start state, either party can make an offer to start the negotiation so the both the buyer's and seller's page show a button (or other control) for making an offer. If the seller makes an offer, the process moves to the "Seller Offered" state and the page for the seller will show no buttons (corresponding to this instance of the bilateral negotiation) while the buyer's page will display the options to make a counter offer or to accept or reject the current offer. As the controls are generated dynamically via reflection on the process model, when the process is changed, for example as shown in Fig. 4, the controls on the web pages will show the correct set of actions without any rewriting.

6 Visual Modeling & Simulation

The FlexFlow engine uses an XML representation of the process definition. To allow business managers to easily create and change FlexFlow processes, we extended popular COTS (commercial off-the-shelf) modeling tools. Since statecharts are a part of the popular UML notation, a number of graphical tools are available. For managing FlexFlow processes, we have added extensions to both Microsoft Visio® and Rational Rose®. Therefore, business managers can use a familiar modeling environment, which provides the following key functionalities:

- Easy-to-use modeling interface for creating or modifying business processes by changing, adding, and/or removing states and transitions from the process state diagram.
- XML generation of the process definition based on the process state diagram.
- Import / Export of the XML process definition
- Management of different versions of process state machines.
- Simulation of the FlexFlow processes

The states and transitions of FlexFlow state diagrams have additional attributes like response views, additional guard properties and priorities. Business managers can import and export XML process definitions via a file or a web-service.

Different versions of a business process can be maintained based on membership at the organization level. Versions can also be selected based on the mode of interaction, such as device, browser, and messaging. Fig. 6 shows the default version of a RFQ process. By specifying new flows, the modeling tool allows administrators to manage several variations of a RFQ process. This way, business managers can model and maintain several RFQ processes (for instance a "Normal RFQ" process, and a "Fast RFQ" process, which is a more compact version of the normal RFQ process).

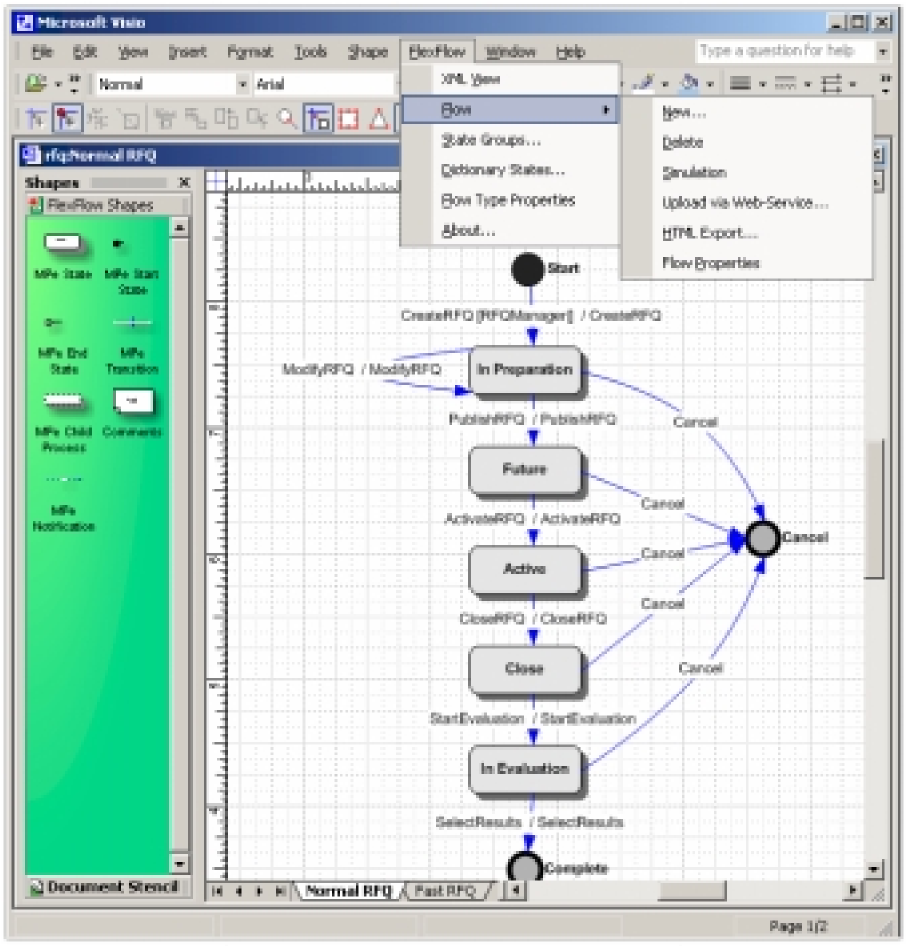


Fig. 6: Visio Modeling Tool for FlexFlow

Process Simulation

In a typical web application, users can navigate to a limited number of web pages based on the actions they take. The number of possible navigation paths can be very large in a complex graphical user interface, but the number is finite and the options usually are known. User interfaces also must stay in sync with the underlying business process. Therefore, process state diagrams reflect the navigation paths of the user at a high level of abstraction.

Process state diagrams can be used to explore hypothetical process models and user interface concepts based on the understanding of the requirements. Users and developers can study a process state diagram to reach a common vision of how the user might interact with the system to perform a task. The business process, business rules and the user experience can be incrementally and iteratively optimized by simulating the business process with user scenarios. These simulations can occur without implementing the business logic. This way, conflicts between the business process and the user interface can be easily discovered.

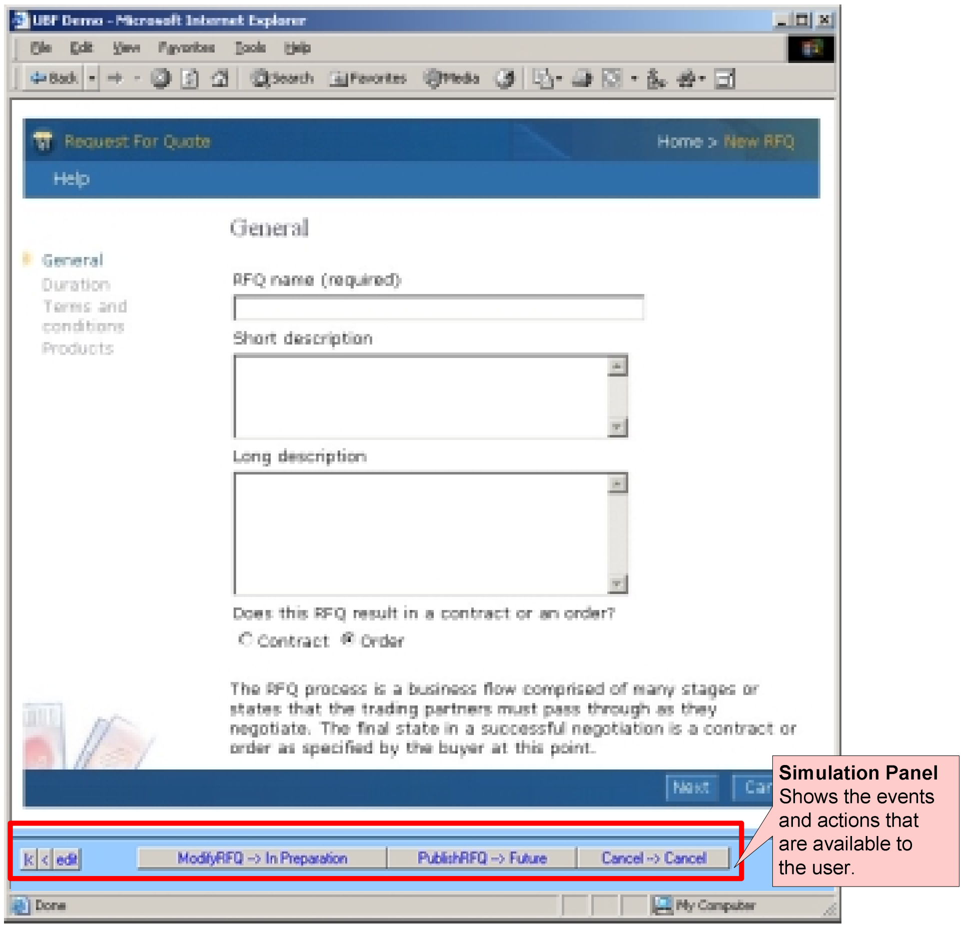
Process state diagrams capture the essence of the user-system interactions and task flow without getting one bogged down too soon in specifying the details of web pages or data elements. Users can trace through a process state diagram to find missing, incorrect, or superfluous transitions, and hence missing, incorrect, or superfluous requirements.

The FlexFlow modeling tool includes a simulation component that allows development of a horizontal prototype which displays the facades of user interface screens from the web application, possibly allowing some navigation between them. The tool does not show real data and contains little or no real functionality. The information that appears in response to a client's request is faked or static, and report contents are hard-coded. Nevertheless, the simulation component allows a process-oriented navigation through the web application. It allows users to change the status of the current process by selecting one of the available actions on the simulation panel. For the simulation, we can include web pages of existing web solutions or new web pages, which can be instantly created and modified. Fig. 7 shows the simulation of the RFQ process. The buttons in the simulation panel at the bottom of the screen show the available navigations paths based on the RFQ process state diagram.

Note, that not all page flows are represented by the control flows of a process. For instance, wizards, like in Fig., or other UI facilitators which have a predefined sequence of processing steps, are implemented solely in the presentation layer and have no impact the process itself. UI components of the presentation layer use the process reflection mechanism to determine functionality, which should be available to the user.

This type of simulation is often sufficient to give the users a feeling for the web application and lets them judge whether any functionality is missing, wrong, or unnecessary. The simulation prototypes represent the concepts to the developers of how the business process might be implemented. The user's evaluation of the prototype can point out alternative courses for a business process, new missing process steps, previously undetected exception conditions, or new ways to visualize information.

By modeling and executing business processes as state machines, FlexFlow enables its process to be modified with minimal changes to the underlying implementation of the business processes. A commerce process can be modified simply by reconfiguring its corresponding state diagram.



General

RFQ name (required)

Short description

Long description

Does this RFQ result in a contract or an order?

Contract [✓ Order]

The RFQ process is a business flow comprised of many stages or states that the trading partners must pass through as they negotiate. The final state in a successful negotiation is a contract or order as specified by the buyer at this point.

Simulation Panel
Shows the events and actions that are available to the user.

Fig. 7: Simulation of a RFQ Process

7 Conclusion & Future Work

Web applications are difficult to build with traditional workflow management systems. In this paper, we presented an approach for managing web-based business processes and introduced a state machine based model for the specification of these business processes. Since e-commerce environments are highly dynamic, we argued that a descriptive model in which business processes are represented “as-is” and “as-to-be” models is advantageous compared to workflow management systems, where separate models are used.

We have shown the FlexFlow system, which supports the modeling, simulation and execution of process state machine. Using the FlexFlow system, developers start the development of a web application with a business process model and afterwards, they can incrementally and iteratively implement all functionalities for this process. Furthermore, web applications remain with FlexFlow customizable and extendible.

Further problems we want to consider in the future include the management of hierarchical states as well as the concurrent execution of FlexFlow processes:

– FlexFlow state machines can be denoted as super-states, whereby each super-state corresponds to a state machine. We want to extend our model to allow states to be nested an arbitrary number of times. Nested states would also allow a notional simplicity for handling duplicate transitions and interrupts.
– Concurrent process state machines sometimes need to be synchronized with each other. Web applications with many business processes demand the ability to start business processes together, run them independently until a certain state and finally re-synchronize them. Forks and joins will allow us to specify more complex transitions to allow this kind of synchronization.
– Besides forks and joins, we want to include a sync vertex in our process model in order to synchronize concurrent regions in a process state machine. A sync vertex is different from a state in the sense that it is not mapped to a Boolean value (active, not active), but an integer. It is used in conjunction with forks and joins to insure that one region of a state machine leaves particular states before another region can enter particular states.

References

1. Dragos A. Molescu, Ralph E. Johnson, A Micro Workflow Framework for Compositional Object Oriented Software Development, OOPSLA, 1999
2. Ellis, C.A., Nutt, G.J., Modeling and Enactment of Workflow Systems, 14th International Conference on Application and Theory of Petri Nets, 1993
3. Genrich, H.J., Predicate/Transition Nets. In: Advances in Petri Nets, 1986, Springer, LNCS 254
4. 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
5. Hammer, M., Champy, J., Reengineering the Cooperation, A Manifesto for Business Revolution, New York, 1993
6. Harel D., Statecharts: A Visual Formalism for Complex Systems, Science of Computer Programming, Vol. 8, 1987
7. Harel, D., On Visual Formalisms, Communications of the ACM Vol.31 No.5, 1988
8. Ian Horrocks, Constructing the User Interface with Statecharts, Addison-Wesley, 1999
9. Jablonski, S., Bussler, C., Workflow-Management, Modelling Concepts, Architecture, and Implementation, International Thomson Computer Press, 1996
10. Kappel, G., Lang, P., Rausch-Schott, S., Retschitzegger, W.: Workflow Management Based on Objects, Rules, and Roles, IEEE Bulletin of the Technical Committee on Data Engineering, Vol. 18/1, March 1995, pp. 11-17
11. Leymann, F., Altenhuber, W., Managing Business Processes as an Information Resource, IBM Systems Journal Vol.33 No.2, 1994

12. Mohan, C.: State of the Art in Workflow Management Research and Products, SIGMOD, Montreal, Canada, 1996
13. Mohan, C., Recent Trends in Workflow Management Products, Standards and Research, NATO, 1997
14. Oberweis, A., Modeling and Execution of Workflows with Petri-nets, Teubner, 1996
15. Reisig, W., Petri Nets: An Introduction, Springer, 1985
16. 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
17. Simon, E., Kotz-Dittrich, A.: Promises and Realities of Active Database Systems, International Conference on Very Large Data Bases, Zurich, 1995
18. J. Sprinkle, C.P. van Buskirk and G. Karsai, Modeling Agent Negotiation, Proceedings of the IEEE Systems, Man, and Cybernetics Conference, October 2000
19. Tsai J.J.P., Yang, S., Bi, Y., Smith, R., Distirbuted Real-Time Systems, John Wiley and Sons Inc., 1996
20. Unified Modeling Language Specification, version 1.4, http://www.omg.org/technology/documents/formal/uml.htm, 2001
21. Weißenfels, J., Wodtke, D., Weikum, G., Kotz Dittrich, A., The MENTOR Architecture for Enterprise-wide Workflow Management, Workflow and Process Automation in Information Systems, 1996