Research and Implementation of Event Driven Multi Process Collaboration Interaction Platform

Xiang Li* and Shuai ZHAO

Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China.

*Corresponding author

Keywords: Event driven, Process modeling, Process validation, Message middleware.

Abstract. Along with the development of electronic commerce and the increase of business requirement, the interactive system becomes the typical form of the daily computing world, and the multiple processes of the system must cooperate with each other to make the whole system work well. In order to make the software system have a complete business processing function and meet the growing demand of the users, this paper proposes an event driven multi process collaboration interaction platform. This platform provides a process interaction design modeling tool with a friendly interface, which can design collaboration diagram to organize the interaction of multi process and support process modeling and process validation. And it can be remotely deployed to the workflow engine of the platform. This platform also includes the operation support of multi process interaction, and it can provide messages integration and filtering between the workflow engines through the jBPM workflow engine combined with message middleware, effectively supporting event interaction in multi workflow engine. Finally, an example is used to analyze and test event driven multi process collaboration interaction platform, and the availability of the platform is verified.

Introduction

In recent years, the demand for interactive integration of business process management systems has become increasingly prominent, and has gradually become a hot research and development of academia and industry [1]. Existing related products provide some support for business process modeling and operation, but there are still some areas worthy of improvement:

1) The commonly used BPMN process modeling tools do not provide the relevant functions to verify the correctness of the business process, but the verification of the business process is an important guarantee to realize the business goal correctly. Therefore, the existing modeling tools need to be modified. Provides formal verification of processes.

2) BPMN2.0 specification of the send and receive is an abstract definition, and does not specify the specific message sending and receiving how to achieve, and the commonly used process engine does not provide a message forwarding mechanism. Therefore, it is useful to provide an easy-to-use message integration method.

In this paper, a multi-process interaction platform based on event-driven is proposed in view of the above problems and the actual situation. The platform for the business process modeling, BPMN formal verification process, the process engine event routing and message filtering, process design and deployment of centralized implementation of such issues as theoretical and technical research.

Platform Design

Figure 1 shows the application scenario of the event-driven multi-process collaborative interaction platform.
From the scene can be divided into two parts of the use of the platform, respectively, for modeling support and runtime support. So the main function of the platform design from the modeling and runtime analysis of two aspects.

**Event-driven Multi-process Modeling**

Event-driven multi-process modeling includes the development of business process collaboration diagrams, event modeling and interaction. Need to provide comprehensive and easy to use BPMN business process interface, business process correctness verification, event definition interface, event routing management and other design tools support.

**Business Process Modeling Functions**

A business process collaboration view can be used to describe the modeling of interactions between different business processes. The modeling process only defines message-level interactions at the process level without focusing on the orchestration details of the activities within the process. Therefore, business process modeling needs to be verified after the process, the specific business process for formal expression, and the use of mathematical means to verify the process to ensure the correctness of the process of modeling and development.

**Event Modeling**

Event modeling consists of two parts: complex event type definition and event subject association. The format of the event interaction between business processes is specifically determined by the design. Event interaction data can be simple, such as a single string, you can also use the custom structure of the data. So the modeling part of the modeling tool needs to provide the function of complex event data definition. Event-driven multi-process collaborative interaction platform uses publish / subscribe system as message middleware, while publish / subscribe message middleware [2] is associated with specific topic. So the event modeling part of the need to provide specific information events related to the function of different themes.

**Event Interaction Modeling**

Event interaction modeling refers to associating different tasks (Tasks) in a collaboration view through a message flow, declaring the sender and the receiver of a message event. In order to visually manage the interaction with the display event, the modeling tool part needs to manage the event routing table to manage and display the specific routing information of the event message. Routing information can be used not only for display, but also to provide the role of specific message routing at runtime.

**Event-driven Multi-business Process Running Functions**

The integration of business process interaction at runtime includes the support of publish / subscribe message middleware for event interaction, and the management of event routing runtime. The
event-driven multi-process collaborative interaction platform adopts the publish / subscribe system as the message middleware and is embedded into the jBPM process execution engine [3]. The business processes located on different process execution engines can interact with each other to achieve event message interconnection. Since the publish / subscribe message message middleware is based on the message subject to filter the message distribution, so it is necessary to provide the message routing table for run-time management.

**Platform Implementation**

Based on the overall design of the multi-process collaborative interaction platform, this chapter discusses the implementation of several main modules in detail, including event modeling, process verification and message middleware integration.

**Event Modeling**

The interaction of business processes in different process execution engines in a multi-process interaction platform is based on message events. The event here is the event passed in the message middleware. In order to combine with the BPMN2.0 specification, use the message elements of BPMN2.0 specification for event modeling. The publish and subscribe system captures the modified Message in jBPM for event delivery and filtering.

**Complex Message Type Modeling**

In the BPMN2Modeler [4] tool, the value of the Message element can be associated with a simple java type. Developers can also manually create custom Java classes in the jBPM project and import custom classes into the process in the BPMN2Modeler tool by importing them. Since the events in the platform are only used as the bearer of the message, the custom Java type associated with the Message does not require very complex method logic, only member properties are required. So modeling tools can provide interface tools to model this custom JavaBean class for message-specific types of associations, reducing duplication of effort and improving efficiency.

**Message Extension**

Event-based multi-process collaborative interaction platform uses publish / subscribe message middleware as the underlying communication facility, and message event filtering in message middleware is achieved through subject filtering of message. But BPMN2.0 specification of the message does not have this attribute, so it’s necessary for the message model itself to do some expansion.

The core of BPMN2Modeler exposes the org.eclipse.bpmn2.modeler.runtime plug-in extension point, one of which is called modelExtension. Add a modelExtension extension point to the BPMN2Modeler plugin.xml file to extend the Message.

**Routing Table Extension**

The event routing table here refers to the information on the message flow in the collaboration view, including the sender, receiver, associated message, and message of the message, on the interface of the modeling tool through the attribute tab, theme.

Graphiti framework provides an easy way to expand, allowing developers to specify the model elements to add property tabs, to achieve the display of the properties of the model-related properties and changes. Add a property sheet to the specified model by setting the id, afterTab, class, type, label, and adding the settings of the routing table properties tab in the plugin.xml file in the BPMN2Modeler project.

**Process Validation**

The platform uses Petri net [5] to formalize the business process and verify the process, mainly to verify the security of the transformed Petri net. A Petri net can be expressed in a graphical way. According to Petri net definition, an identity Petri net is safe if and only if it is 1 bounded. The
security of the process is verified by applying and modifying the Karp-Miller algorithm to guarantee the security of the formalized Petri net flow. For Petri net, the K & M algorithm is as follows:

Algorithm 1. K&M

In: A Petri net \( N = (P, T, I, O, m_0) \).
Out: A labelled tree \( C = (X, x_0, B, \Lambda) \).
1. Let \( x_0 \) be a new node such that \( \Lambda(x_0) = m_0 \)
2. \( X := \{x_0\}; \text{Wait} := \{(x_0, t)|\Lambda(x_0) \xrightarrow{t}\}; B := \emptyset \)
3. WHILE Wait \( \neq \emptyset \) DO
4. Pop \((n', t)\) from Wait.m := Post(\(\Lambda(n'), t\))
5. IF \( \nexists y \in \text{Ancestor}_c(n') | \Lambda(y) = m \) THEN
6. Let \( n \) be a new node s.t. \( \Lambda(n) = \text{Acc} \left( \Lambda \left( \text{Ancestor}_c(n') \right), m \right) \)
7. \( X = X \cup \{n\}; B = B \cup \{(n', t, n)\}; \text{Wait} = \text{Wait} \cup \{(n, u)|\Lambda(n) \xrightarrow{u}\} \)
8. END IF
9. END WHILE
10. RETURN \( C = (X, x_0, B, \Lambda) \)

Message Middleware Integration

The subscriber or publisher of the message utilizes and encapsulates the interface provided by the publish / subscribe messaging middleware to consume or publish message events. Message publishing logic is the process of running the engine generated by the specific message events issued by calling the release interface for news release. At the subscribing end of the message, the subscriber needs to subscribe to the subscribing interface to declare a subscription to a message event associated with a topic and pass in the specific callback processing logic. The callback processing logic defines the specific parsing process for the message. When the message event of the topic is generated in the publish / subscribe message middleware, the message middleware pushes the message to the subscriber's process engine. The callback processing logic will parse the specific of the event message, and the role of the process engine.

Summary

The proposed event-driven multi-process collaborative interaction platform solves the problem and achieves the expected results and satisfies the basic functions of the practical application requirements. However, the existing platform still needs to be improved: if the existing message routing and filtering only consider the interaction of multiple business processes, can’t distinguish between multiple instances of the process. In the follow-up work, will carry on the further research to the above-mentioned question.

References
