Events Monitoring System on an Event-Based Architecture

Hideyuki KASHIWASE

School of Information Science,
Japan Advanced Institute of Science and Technology

February 15, 1999

Keywords: Event-Based Architectures, Distributed Cooperative Works, Events Monitoring.

1 Background and Purpose

We can comfortably interact with each other during our collaborative work while we can be aware of our situation. When we work together under the distributed computing environment, we should interact with computer systems as well as our colleagues. Therefore, our computer systems should be also aware of the situation of our work.

With several kinds of systems, e.g. structured e-mail or work flow systems, we should tell our system about our situation explicitly. But we can not or do not always tell our system correctly and frequently our situation, because such kinds of additional and troublesome tasks are not always acceptable for collaborative workers.

In this paper, we introduce a method for monitoring the situation of our collaborative work. In this method, we use a general behavioral model of a collaborative work for producing a shared product. And we deduce the transitions on the model from the event histories accompanied with our tool operations, such as drawing diagrams, editing texts, sending messages and so on. This method do not enforce additional tasks for reporting their situations upon collaborative workers. We also performed experiments for our method to evaluate and discuss its advantages and disadvantages.

2 A Collaboration Model

Ochimizu et al. present a general state transition model of a shared product, which transitions are fired by interactions of collaborative workers. But we can not design and implement a supporting system based on this model directly, because the relationships between concrete events and each states and transitions are not defined. So we defined such events for adapting a task of reviewing and refining a object diagram.

The features of states in our model are as follows;

Developing or Modifying ; A leader is making a draft or modifying a draft.

Possibility of errors ; The other workers are detecting the possible errors in the draft made by the leader.

Coordination ; All workers are coordinating about the draft which a leader made.

Stable ; All workers consider the draft has no more errors at that time.

For example consider the sequence of state transitions as follows; "Developing or Modifying" $\rightarrow$ "Possibility of errors" $\rightarrow$ "Coordination" $\rightarrow$ "Developing or Modifying" ...
This loop means improving of class diagrams step by step. If "Possibility of errors" state has no errors, transits in from the state "Coordination" to the state "Stable" directly. A state transit of "Stable" to "Coordination" means re-examination of class diagrams.

3 Event-based Architectures and Monitoring systems

Under the event-based architecture, a distributed software component does not interact with other components directly and synchronously. When a component interacts with another component, it sends a message not to another component directly but to *an event bus*, which knows which components should accept the message. Based on this mechanism, software components can asynchronously work together.

The event-based communication style is characterized by the following properties:

- **Multicast**
- **Implicit Invocation**

This style holds the promise of supporting a flexible and effective interaction among highly reconfigurable distributed software components.

Recently, this style is developed using the communication model implemented by Distributed Objects (CORBA and Java-RMI etc.).

On the other hand, this style holds the promise of events of between components must go through *an event bus.* So, the system can monitor events among components.

In this paper, we developed a shared whiteboard which is a collaboration tool using duplicated distributing style.

We implements the Event-based architecture using Java RMI. The shared whiteboard communicates using Java RMI. According to the *Mediator*, components interact with events implicitly and in multicast.

4 Methods of monitoring and Evaluation

Each state of our model is characterized by patterns of event sequences as follows;

Developing or Modifying State Sequence of drawing events issued only by the leader.
Possibility of errors State Sequence of drawing events issued by all members except the leader.
Coordination State Sequence of drawing events issued by all members.
Stable State No event are monitored.

Based on this characteristics, the system can monitor the situation of collaborative workers.

For evaluating the validity of this method, we performed an experiment distributed meeting for developing the diagrams. As a result, we have the following findings;

- In "Coordination" state, events for exchanging email messages are also monitored as well. Hence we should revise our first model.
- Because we did not define a pattern of event sequences corresponding to "Stable" state in our model, it is not clear to decide whether here is "Stable" state or not. So we should define such patterns with another kind of events, e.g. releasing a new revision of products.
- We append new transition from "Possibility of errors" to "Developing or Modifying", to our first model.
- Several exceptional events are observed.

As we mentioned, we discussed about future works considering these problems.

We discussed about improving monitoring precision about drawing events granularity and contents analysis of an e-mail.

5 Future works

The tasks in the future are shown as follows;

- To improve a state transition diagram model for collaboration works.
- To refine the rules of events analysis.
- To develop an automatically transition system.
- To implement the system on CORBA for "Jizai" in our OCHIMIZU-lab.