Abstract

Activity diagrams are a recognized form of modelling real time systems and have been included in the list of UML techniques for modelling the dynamic aspects of Object Oriented Systems. In this paper, we explain how workflow techniques can be used for developing models of larger chunks of systems. These chunks are expressed through the use of worklets. This can then be used as the basis of modelling systems where timeliness is of importance. However, this requires extending workflow ideas to incorporate flexibility, handling of exceptions and adaptability. Extensions of the activity diagrams are then proposed for expressing the models obtained from such workflow techniques. This will allow evolution of real time systems development towards a component based approach.

1 Introduction

Several techniques have been introduced in the literature for modelling the dynamic aspects of systems including state diagrams, state charts, collaboration diagrams, sequencing diagrams use cases, activity diagrams and coloured peti nets. The first two are effective for modelling object life cycles while the rest are directed at modelling the inter object dynamic aspects. Of these, the last two are most effective at capturing multithreaded, asynchronous systems. However, in their current use most of these approaches model the system at too low a level of granularity and therefore are basically not compatible with a component based approach to the development of real time systems. What one needs is an approach at “chunking” the system to increase the granularity at which the original system is visualised and then to express this "chunked" high granularity system in a semiformal representation that reduces ambiguity. The necessity for modelling the system at a higher level of granularity has increased with the appearance of large complex real time systems that involve many processes running on several different processors connected together through a network.

In this paper, we propose a methodology for doing this ‘type of modelling’...
This paper considers extended workflows that have flexibility, modularity and timeliness representations.

Flexibility can be distinguished into flexibility by selection/configuration and flexibility by adaption [5, 8]. Flexibility by selection gives the user a choice on how to execute a workflow by providing several different execution paths. It is useful when the number of execution paths for a workflow are either known and/or anticipated. However, flexibility by selection itself is limited by the requirement that it has to be anticipated and has to be included into the workflow model at design time. To support flexibility by adaption, the workflow management system has to provide additional functionality and tools to change the workflow instance and to integrate these changes during run time [3].

Another approach to support flexibility by adaption is by means of advanced inheritance concepts, which are proposed in [5]. The inheritance concepts can be used to limit change, manage multiple version/variants, and avoid errors.

Other problems that have to be considered are correctness, timeliness and consistency issues particularly for real time systems which may also have a safety critical dimension. It is essential that workflow systems are able to cope with changes, but it is also important that it has to verify the correctness and the consistency of the workflow definition, after the changes have been made. Changing a workflow definition can lead into a deadlock or a livelock. Transferring a case from an old workflow definition to a new one can cause a task to be executed more than one time or not executed at all (skipped). Note that all these problems could happen when flexibility by adaption is applied.

In this paper, we are only going to focus on flexibility by selection. It means that we focus on the specification of a flexible workflow execution behavior to express a structured and unstructured set of processes in advance. Although, flexibility by adaption can overcome the lack of flexibility of selection, flexibility by adaption should be reduced to unforeseen and exceptional exceptions. Otherwise, the need of flexibility by adaption indicates an inadequate workflow model, which results from an insufficient model building paradigm or from inappropriate modeling language constructs. [8]. Such flexibility by selection could be important in real time systems that require integration of human and machine actions such as air traffic controllers, train controllers, and logistics management systems.

Since flexibility by adaption requires human intervention, it is important to have a workflow modeling language at a high level of abstraction, which is easy to use and supports the visualization of its contents [2, 14].

3.0 CONTENTS OF A WORKFLOW MODEL

In general a workflow is considered as a collection of activities for the accomplishment of tasks. Each sequence of activities has to be executed in accordance with the processes of the system at hand [6]. These activities could be carried out by humans or machines in an integrated fashion. So, what should be the contents of a workflow model? Typical contents of traditional workflow models are: [11]

- Organizational unit, which describes the organization of a system or sub system
- Role
- Activity, also called task or function, which describes a piece of work within a process:
- Data, which describes relevant data for the process.
- Event, which describes an incident that might occur during the accomplishment of an activity.
- Resources, which described a device, a machine or a tool, which will be required to accomplish a task.
- Process, which contains several activities in the order they have to be executed to attain a goal of the system.

Jablonski and Bussler also discussed the main components of a workflow model [7]. They discussed them as perspectives of the workflow model. There are five perspectives, which are important that every workflow model should capture:

- Function perspective. It describes the functional units of a work process that will be executed.
- Operation perspective. It describes how a workflow operation is implemented.
- Behavior perspective. It describes the control flow of a workflow.
- Information perspective. It describes the data flow of a workflow.
- Organization perspective. It describes who has to execute a workflow or a workflow application.

A workflow model also can contain conditions or constraints because activities cannot be executed in an arbitrary way. Two types of conditions can be applied in a workflow model, which are pre-conditions and post-conditions. Pre-conditions have to be satisfied in order to execute a workflow. Post-conditions have to be satisfied in order to finish a workflow. These kinds of conditions also can be applied to a single task.

Reference [9], uses a norm specification to indicate the condition, the agent and action to be taken. This norm represents the rules that are imposed on the particular process. The norms also allow exceptions to be specified in them. Besides handling the rules and exceptions, the norm provides a degree of flexibility that allows the modeler to introduce additional exceptions that may be discovered in the later stages of analysis.
Moreover, in a workflow model the following minimal information should be associated with each activity [13]:

- Who has control over the activity through the assignment of activities to qualified users or application functions.
- Input/output of the activity, i.e. the data and control information required for task accomplishment
- Pre and post conditions of the activity.
- Which other activities are required to complete the activity.

4.0 WORKLET

Worklet is a small workflow, which is carried out by the organizational unit. It contains one or more actions. One could think of a worklet as simply a sub-workflow, however, the main difference is a worklet is a sub-workflow that is carried out only by one organizational unit. Each worklet has a unique WorkletID.

The idea of the worklet is to reduce the complexity of process models. It reduces the difficulty for the modeller in both modeling and maintaining the models by abstraction. Another objective of the worklet is to achieve independence of enactment of processes. Let us consider a company, which produces a product where more than one department is involved in the production. We believe that the workflow models will be very large. Moreover, if the production time is very long then departments, which have done their jobs, have to wait till the process finishes. This will be very time consuming and inefficient.

Action in worklet is able to interact with other actions in other worklets through its interface. An Interface has two attributes: Destination and Method. Destination is WorkletID of an action that the source action wants to communicate to. Method represents the means that the actions use to communicate. It can be e-mail, mail, telephone, etc. Actions could have more than one interface. To support flexibility, the modeler can add new interfaces and assign the WorkletID and Method of the interfaces during run-time.

Each worklet must have at least one Entry constraint. The entry constraint has to be satisfied to start a worklet. A worklet can be started either manually or automatically. To start a worklet manually, the user simply "fools" the system. For example, when a customer places an order via telephone, there is no way for the system to automatically start processing the order. Human intervention is needed to start the worklet. The user should be able to enter the Entry constraint and run the worklet (e.g. by pressing "start" button). On the other hand, to start a worklet automatically, human intervention is not needed. The system simply has to check the Entry Constraint frequently, if it is satisfied then it will start the worklet automatically.

Since the worklet can be started in two ways, therefore we have another attribute for the worklet, called Start Method. There is only two possible values for this attribute, manual and automatic.

Like other attributes of worklet, the value of Entry Constraint and Start Method can be changed at run-time and the number of Entry Constraints can be added or reduced.

4.1 ACTION

An action is a single indivisible task which is done by an actor or actors. Actions can be categorized into two types in terms of how they are done, manual-action and automatic-action. Manual-action is an action that is manually done by actors (e.g. filling an order). Meanwhile, an automatic-action is an action that is automatically done by workflow management systems (e.g. sending an e-mail).

Ideally, an action cannot be done in an arbitrary way. There should be conditions that need to be satisfied first to enact the action. These conditions are called pre-conditions. These pre-conditions could be the rules governing the process. Post-conditions are conditions that need to be satisfied so that a task can be said as 'complete' or 'done'. An action also can have input and output, in form of data, documents or other objects (e.g. order form).

Like a worklet, human intervention is needed for manual-action. There is no way for the system to know if a manual-action is done or not. Users must let the system know if they have finished doing their jobs (e.g. by pressing "done" button).

4.2 EXCEPTION HANDLER

To support flexibility, we like to propose another object and it is part of the action object. This object is for handling exceptions that might occur at the process instance level. The object is called ExceptionHandler. It has two attributes, Condition and WorkletID. Condition attribute represents the condition that will be evaluated in the worklet for handling exceptions. This condition is specified when an exception happens. WorkletID attribute represents which worklet needs to be run to handle the exception. This worklet can be modeled even if process has been run. This technique allows the modeler to change and extend the original workflow for handling an exception.

The authorized user has to fill the condition and specify the WorkletID and then s/he could press the 'start' button to start the exception handling procedure.

However, one has to remember that this change only applies to a single process instance or case. This change will not effect to original process/worklet definition, therefore other process instances still refer to the old definition. This technique can be used if the exception will not happen in the future. However, if the exception
happens so frequently, we suggest that the changes should be applied directly to the process definition so that remodeling the workflow at run-time can be avoided.

5.0 TERMS AND CONCEPTS OF ACTIVITY DIAGRAM

An Activity Diagram is one of the five diagrams in UML for modeling the dynamic aspects of systems. An activity diagram is basically a diagram that shows flow of control from one activity to another activity[16]. The activity is triggered by one or more events and the activity may result in one or more events that may trigger other activities or processes. Events, which are message flows in UML, start from the start symbol and end with the finish marker having activities in between connected by events.

All activity diagram can be thought of as defining the reaction to that process, which continues until everything that needs to be done is done. Activity diagrams represent the decisions, iterations and parallel/random behavior of the processing. Graphically, an activity diagram is collection of vertices and arcs.

A common use of activity diagram is to model the dynamic aspects of a system. When we model the dynamic aspects of a system, we typically use the activity diagram in two ways: [10].

- To model a workflow. Here, we focus on activities as viewed by the actors that collaborate with the system.
- To model an operation. Here, we use activity diagram as flowcharts, to model the details of computation.

Activity diagrams commonly contain the following elements:

- A filled-in circle, which indicates the start of overall activity. (Fig. 1)
- An action state, which represents an executable computation that cannot be decomposed anymore. It appears on an activity diagram as a lozenge shape (a symbol with horizontal top and bottom and convex sides). (Fig. 2)
- An action state, which represents a set of action states. It also appears as a lozenge shape, containing the name of the activity and perhaps some other information. (Fig. 3)
- A transition, which represents a change from one action state or activity state to another. A transition appears on an activity diagram as a solid line with a feathered arrow pointing to the new action state or activity state for the given object. (Fig. 4)
- A branch shows decision points at which one path may become two or more new paths. It appears as a diamond on an activity diagram. (Fig. 5)

UML also has a very useful technique in modeling a workflow called swimlane. It is used in activity diagrams too. A swimlane is a group. A group here represents the organizations, which are responsible for activities. Therefore, by having a swimlane, we partition the activity states on an activity diagram into groups. This technique gives us benefits in modeling and reading a workflow.

6.0 EXTENDED ACTIVITY DIAGRAM

By adopting the idea from activity diagram of UML, we try to extend it so that it will allow us to model these flexible workflows.

In this project, we suggest that a workflow model should contain these following objects:

- **Data**: represents the data that will be used or produced in an action (e.g. e-mail, order form). Data can be categorized in two kinds, **Input Data** and **Output Data**. Input Data is data that will be used for an action and Output Data is data that is produced by an action.
- **Conditions**: represents the conditions of an action. Conditions can be divided into two kinds, namely **pre-conditions** and **post-conditions**. Pre-conditions are conditions that need to be satisfied so that an action can be enacted. Post-conditions are conditions that need to be satisfied to tell an action has been finished.
- **Resource**: represents ‘things’ which will be required to do an action, (e.g. machine, device).
- **Actor**: represents employees who will do an action.
- **Action**: represents piece of work, it can be manual action or automatic action.
- **Worklet**: represents an activity/business process that carried out by an organizational unit.

7.0 NOTATION FOR EXTENDING ACTIVITY DIAGRAM

Here we propose some new notations for the extended activity diagram.
An action can be viewed as a big object that contains other objects. It is still represented as lozenge shape, but it has other rectangles above and below it. Figure 8 shows the notation.

To represent the interface of one action in one worklet to another action in another worklets we use pentagon symbol (Figure 9).

To represent an exception handler object, we use a hexagon symbol (Figure 10).

For the rest of notations, we keep them still the same as in the UML activity diagram. We expect that these notations will help the modeler to have more flexibility in modeling a workflow. In the next section, we try to use this extended activity diagram to model a real world process of one of the information logistic companies in New South Wales, Australia and Hong Kong.

8 Illustrative Example

Logistics Management is part of Supply Chain management system. Logistic Management, in its widest definition, is concerned with the strategy and management of the movement and storage of materials and products from suppliers, through the firm’s distribution systems to retail outlets and customers in a timely fashion. The scope of logistic management is the physical movement of goods starting with the sources of supply and ending at the point of consumption [4].

In today’s business environment, good logistics management often determines the success of a business. Retailers are well aware of how excess inventory, frequent stock-outs, poor item turnover, and excessive markdowns can cut into profits. Logistic management attempts to achieve a balance between holding minimum stock while providing the best services possible to the customer.

SeaPower Resource International (SRI) is one of largest Cold Storage Warehouse and Logistics Company in Asia Pacific region. Its main service is to provide space for customers who want to store their goods in the warehouse and shift their goods from origin location to destination location. SRI provides detailed logistic services for its customer to move their goods from one place to another place. In this project, we only concentrate on logistic part.

SRI has many types of logistic orders and each order has its own flow of work. Some of these orders are:

- Import
- Export
- Local Delivery

TRN-DEPT WORKLET

The main duty of Transportation Department is to schedule and assign SRI’s trucks to a particular e-Order. The first task is to query its database and find out the availability of the required truck from an e-Order. If the truck is not available, then it has to contact another transporter company and book a truck from it.

After it finds the required truck, it will send a confirmation to the Logistic Department, indicating that the required truck is available. The next task is to ask the truck to go to warehouse to pick up the goods. Note that, this task can only be performed when the goods in the warehouse are ready (pre-condition of the task). It is because we do not want the truck waiting too long at the warehouse, especially when the truck comes from another transporter company, and this will make the charges of the truck will get higher.

Once the goods are in the truck, then it will deliver to the Customer B. Customer B then will produce proof delivery and give it to truck driver. The truck then goes back to Transporter Department and gives the delivery proof to Transporter Department. Delivery proof is then sent to the Logistic Department via e-mail and then the worklet reaches the finish state. We assign the WorkletID for Transporter Department to 0003. The entry constraint for it is a truck-booking request, in form of e-mail, from Logistic Department is received.

This is only an worklet out of twenty or so to model the entire logistics process. Space does not allow us to show the rest here.

Conclusion

In this paper, we presented a method of “Chunking” large complex real time system using the concept of a worklet and workflow. We then introduced extensions to the UML based activity diagrams to allow modelling of these flexible adaptive workflows. This is illustrated using a real life example from a logistics system.
References


WorkletID: 0003.
Entry condition: Truck booking e-mail is received from LOG-Dept.

Figure 5.6: Transport Department worklet.