E-Contract Enactment Using Meta Execution Workflow

Pabitra Mohapatra[1], Pisipati Radha Krishna[2], and Kamalakar Karlapalem[1]

[1] Center for Data Engg., International Institute of Information Technology, Hyderabad, India
pabitra.mohapatra@research.iiit.ac.in, kamal@iiit.ac.in
[2] Infosys Labs, Infosys Limited, Hyderabad, India
radhakrishna_p@infosys.com

Abstract. E-contract fulfillment has many challenges because enactment is cross organizational and it involves interdependencies among its various elements namely parties, activities, clauses, exceptions, payments and commitments. E-contract needs inter organizational workflow services for monitoring its enactment. In this paper, we introduce the concept of e-contract elements based workflow views in order to bridge the gap between different aspects of e-contract and the services provided by a meta-workflow system. The enactment of e-contract is carried out using the meta execution workflow. It enables coordination among the workflow views and the workflow system for successful fulfillment.

Keywords: E-contract, Workflow Views, Dependencies, Exceptions.

1 Introduction

This paper focuses on e-services driven meta-workflow system for monitoring e-contract enactment. Usually, e-contract enactment and deployment necessitate inter- and intra- organizational workflows. Enterprises implement their own workflows, however, capturing and handling various dependencies that exist among six contract elements (namely *parties*, *activities*, *clauses*, *payments*, *exceptions* and *commitments*) is a challenging task. The interdependencies among elements for e-contract fulfillment include (i) activities carried out by specified parties(at times involves sub-contracts to carry out some of the activities), (ii) satisfaction of a set of clauses associated with activity executions, (iii) handling exceptions, if any clause violation and (iv) ensuring payments before commitments [2][3]. Thus, a workflow management system (WFMS) should support the functions of each contract element, besides the activity/task execution. Our research question is how to monitor the activities' execution from multiple perspectives of e-contracts such as activity commitment, clause violation and payments due. We propose an Execution Workflow based services for effectual monitoring and successful completion of e-contract. For e-contract monitoring, we have associated *workflow views* for each e-contract element.

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 714–717, 2013.

2 E-Contract Enactment System

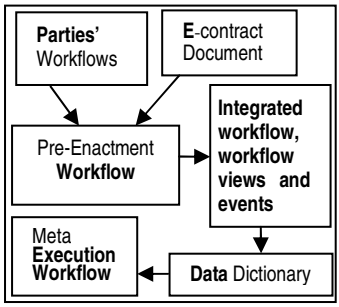

Fig. 1. System Overview

Fig. 1 shows an overview of our system, which consists of two major components: (i) *Pre-enactment workflow* and (ii) *Meta execution workflow* [4]. The *pre-enactment workflow* builds an integrated contract workflow that comprises of parties' workflows. The e-contract *meta execution workflow drives* the e-contract execution and enables workflow views to support the contract elements. We specify individual workflows for parties, each having some tasks, and the tasks have some pre-events and post-events. The pre-enactment workflow is a *specification component* and the input to this component is the workflows belonging to different parties and the e-contract document. This component maps the contract elements and their dependencies into an *integrated contract workflow*. Further, workflow views are derived from the integrated workflow. All the specifications are stored in the Data Dictionary.

A major problem of having multiple workflows is that incorporation of tasks and events related to clauses, exceptions and payments at appropriate places might get ignored. In e-contract, the tasks are attached to the elements so that it ensures the execution of the tasks for e-contract enactment. Further, clauses are dependent on the execution of the tasks by one or more parties. So, multiple workflows defined for various tasks are difficult to coordinate among various parties as there are dependencies between tasks. The question is how to monitor and control the execution of an e-contract using workflows. The parties need to know the events and the coordination between the parties, and they should be specified explicitly. The parties involved can resolve exceptions by triggering alternate workflows or at times human intervention is required for resolving the exceptions.

We build an *integrated workflow that combines individual workflows (of parties) in the required order and allows the flexibility of augmenting additional tasks and events (with the help of user interaction) as required by the contract specifications.* To construct the integrated contract workflow the steps followed are given below:

- Identification of the six elements in the e-contract document using MTDC approach [1].
- After mining the document, the output will be a set of tasks executed by the parties.
- Determine coordinated set of tasks using the Workflow Specification Model of MTDC.
- Arrange the tasks in the sequence of their execution, the events generated after task completion is received by the intended party's task. It will help understanding the interdependencies among the tasks and orchestration of workflow execution.
- Derive/augment integrated workflow with the coordinated tasks in case these tasks are executed in a sequence, otherwise derive independent workflow and complement it with the integrated contract workflow.
- Represent the workflow using Yet Another Workflow Language (YAWL) [5].

Consider the Seller-carrier-buyer Example. Individual workflow of the parties is shown in Fig. 2 and the integrated workflow is shown in Fig. 3. The three parties involved are *buyer*(P1), *carrier*(P2) *and seller*(P3). Buyer's Activity (BA), Seller's Activity (SA) and Carrier's Activity (CA) are BA1, SA2, SA3, SA4, CA5, SA6, SA7, SA8, CA9, CA10, BA11, BA12, BA13, BA14, BA15, BA16, CA17, BA18.

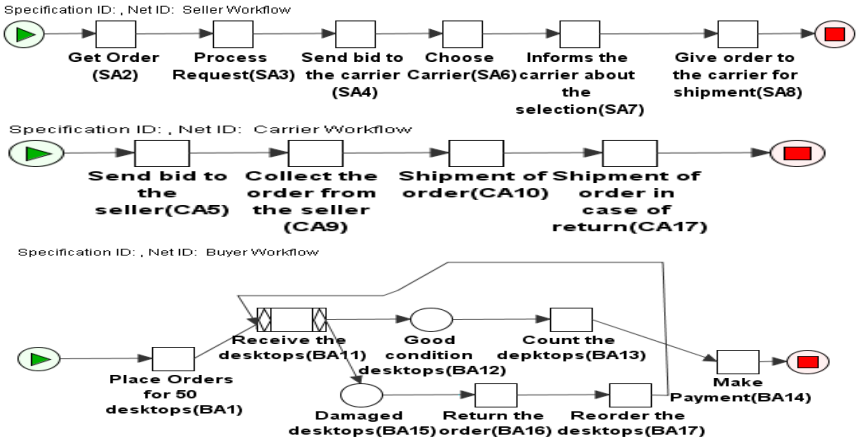

Fig. 2. Individual workflows for seller-buyer-carrier

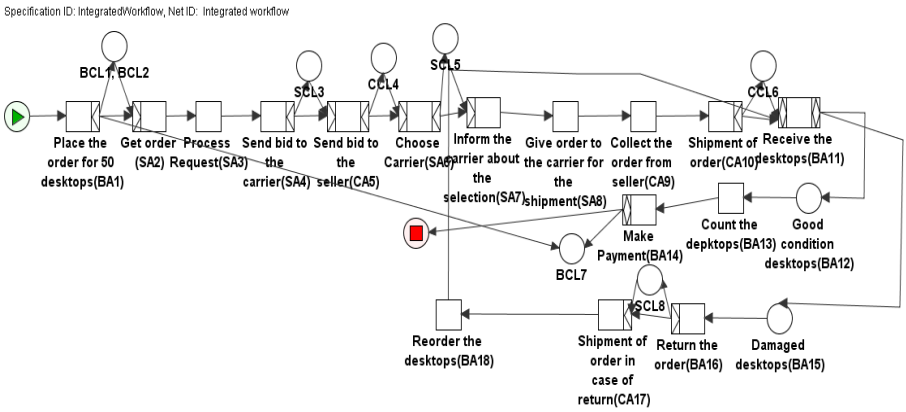

Fig. 3. Integrated workflow for seller-buyer-carrier

The clauses are denoted as BCL (buyer's clause), SCL (Seller's clause), and CCL (carrier's clause) and the exceptions are denoted as BX (buyer's exception), SX (seller's exception) and the CX (carrier's exception). BCL1 says the quantity has to be specified by the buyer while placing the order, if it fails then the BX1 will occur and the seller would not fulfill the order. BCL2 says customer details should be provided, if not done the BX2 will occur and the seller would not give the shipping address to the carrier. SCL3 says the carrier should be chosen by a bidding process. CCL4 says the response to the bid by the carrier to the seller should be done in three working days. SCL5 says the carrier has to be chosen in five working days. CCL6 says the shipment should be done in three working days if it fails to do so then CX3 will occur and the seller will choose another carrier. BCL7 is regarding the mode of payment, it had to be decided at the time of placing order if it fails to do that the BX4 will occur that the seller would not ship the item. SCL8 represents free shipment in case of damaged goods. The payment task is BA14. Once all the tasks are done then commitment

of the contract will happen, which is decided by the parties. The results of this approach will help setting up a coherent relationship among the elements.

We developed e-contract elements based views for six elements namely *Activity view*, *Party view, Clauses view*, *Exceptions view*, *Payments view* and *Commitments view* in order to monitor the progress of e-contract execution during its enactment. Mapping of the tasks of integrated workflow is done with the respective views. The views will have information regarding the input and output events for their respective set of tasks. Along with HOW, WHEN, WHERE and WHO of tasks got executed, the views can monitor the workflow execution based on the events captured during e-contract execution. Here, the views (or sub-views) serve as a black box, and focus more on the input and output data, conditions and events.

Activities View: BA1→SA2→SA3→SA4→CA5→SA6→SA7→SA8→CA9→CA10→BA11→ BA12→BA13→ BA14→BA15→BA16→CA17→BA18→BA11→BA12→BA13→BA14.
Parties View: P1: BA1→BA11→BA12→BA13→BA14→BA15→BA16; P2: SA2→SA3→SA4→ SA6→SA7→ SA8; P3: CA5→ CA9 → CA10 → CA17;
Clauses View: BCL1: BA1→SA2; BCL2: BA1→SA2; SCL3: SA4→CA5; CCL4: CA5→SA6; SCL5: SA6→SA7; CCL6: CA10→ BA11; BCL7: BA1→BA14; SCL8: BA16→CA17
Exceptions View: BX1:BCL1→BA1→SA2; BX2:BCL2→BA1→SA2;CX3:CCL6→CA10→BA11; BX4:BCL7→BA1→BA14
Payments View: In our example, the activity A9 corresponds to payments. PY: BA14
Commitments View: C1: CA10→BA11; C2: BA14

Meta Execution Workflow (EW) defines and executes the integrated workflow. The change in workflow definition occurs when some exception occurs. For instance, if any of the exceptions like BX1, BX2, CX3 or BX4 occurs, then the EW will try for another EW definition. In our approach, the workflow instance and the EW instance are coupled together. Here, the EW helps in keeping track of all the events that in-turn helps in monitoring the e-contract elements successfully.

3 Conclusion

In this paper, we developed pre-enactment workflow to take the individual workflows from parties and generate integrated contract workflow. We also presented an execution workflow component that facilitates execution of integrated workflow and monitors the execution through workflow views specified for various contract elements. We illustrated our methodology by using the seller-carrier-buyer contract.

References

1. Khandekar, A., Krishna, P.R., Karlapalem, K.: A Methodology and Toolkit for Deploying Contract Documents as E-contracts. In: Tutorials, Posters, Panels and Industrial Contributions at ER 2007, vol. 83, pp. 91–96. Australian Computer Society Inc. (2007)
2. Krishna, P.R., Karlapalem, K.: Electronic Contracts. IEEE Internet Computing 12(4), 80–88 (2008)
3. Krishna, P.R., Karlapalem, K., Chiu, D.K.W.: An ER^{EC} Framework for E-Contract Modeling, Enactment and Monitoring. Data and Knowledge Engineering 51(1), 31–58 (2004)
4. Sharma, S., Karlapalem, K., Krishna, P.R.: A Case for a Workflow Driven Workflow Execution Engine. In: Proc. of Workshop on Info. Tech. and Systems (WITS), Florida (2012)
5. van der Aalst, W., ter Hofstede, A.: YAWL: yet another workflow language. Information Systems 30(4), 245–275 (2005)