A Methodology and Toolkit for Deploying Contract Documents as E-contracts

Anushree Khandekar[+], P. Radha Krishna[*] and Kamalakar Karlapalem[+]

[+] International Institute of Information Technology, Hyderabad, India.

anukhandekar@students.iiit.ac.in, kamal@iiit.ac.in

[*] Institute for Development and Research in Banking Technology, Hyderabad, India.

prkrishna@idrbt.ac.in

Abstract

Recent research in e-contracts is concerned with the development of frameworks and tools to support contracts. ER^{EC} framework is one that enables modelling and deployment of e-contracts. In this work, we have designed and developed a toolkit for contract visualization and enactment based on ER^{EC} data model. This toolkit provides appropriate services for semi-automated contract execution starting from converting a paper contract document to e-contract document, identification and extraction of various entities involved in e-contracts to workflows that enact the contract. Our e-contract engine has a Pattern Recognition System, Sentence Classifier, ER^{EC} Model Component, Workflow Specification Component, and Xflow Workflow Enactment Component. Our toolkit systematically processes a domain specific paper contract documents into a deployable and executable e-contract. A dictionary is prepared by extracting common contract-Keywords from multiple related contract documents for a specific domain. The pattern recognition system based on a named entity extractor identifies parties, important dates and places. The sentence classifier along with human interaction extracts the activities and clauses based on the contract-Keywords and identify interdependencies to generate ER^{EC} data model. The ER^{EC} model component takes these specifications as inputs and generates ER^{EC} data model for a specific contract. Finally, the data model is mapped to workflow specifications by Xflow specification component, which in turn generates the workflow instances for execution of e-contracts by Xflow workflow enactment component.

1 Introduction

A contract is an agreement between two or more parties especially one that is written and enforceable by law. It consists of several entities such as parties, activities and clauses. Two or more parties are involved in performing activities of a contract and these activities have to satisfy the clauses as specified in the contract document. An e-contract is a contract modelled, specified, executed and deployed by a software system. Contracts are complex to understand, represent and process electronically, and described in voluminous documents. Moreover, the contract documents are prepared using legal jargon besides having several spatial, temporal and social constraints; designing and developing an e-contract system is not straight forward and presents lot of challenges.

In e-contracts, the activities may be carried out by different parties from different organizations and require varied services. Usually, the short-term contracts have duration of few days, while long-term contracts last for months or even years. In addition, an e-contract may have interrelated dependencies within the activities as well as between activities and clauses. Due to the technological advancements and increased scope of e-contracts across multiple domains and applications, the complexity and dynamism of business transactions has been significantly amplified. Because of this nature, there is a need to map e-contract requirements and specifications gleaned from e-contract document to (semi-)automated workflows that enact the e-contract. After specifying an e-contract, it is important to have a procedure for its execution according to its specifications. Information technology can be used to process a textual contract and extract relevant information to be able to map it to a deployed e-contract.

The goal of an e-contract system is to have specification of the activities and clauses, mapping them into deployable workflows and providing execution support for its enactment. The main focus of this paper is to map a document contract into an executable e-contract. In this work, we have designed and developed a toolkit for e-contract visualization and enactment based on ER^{EC} data model. Our MTDC (Methodology and Toolkit for Deploying Contracts) solution has a Pattern Recognition System including Named Entity Extractor, Sentence Classifier, an ER^{EC} model component, Workflow Specification Component and Xflow Workflow Enactment Component. Our toolkit does automation of labor-intensive routine tasks and leaves more demanding tasks to human designers. We have tested our system on few categories of contract documents including sales, marketing and rental contracts.

The rest of the paper is organized as follows: Section 2 describes the background related to ER^{EC} framework, which is basis for our prototype system. Section 3

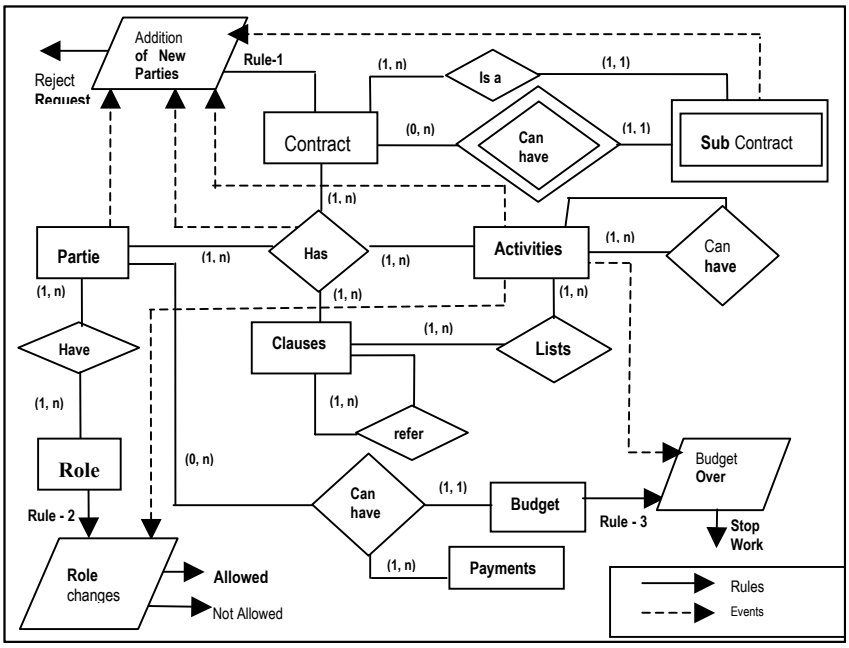

Figure 1: An ER^{EC} Meta-Model for E-Contract

presents our MTDC approach for implementing the system and section 4 briefly describes various components of the system. In section 5, we illustrate our approach with a sample contract. Section 6 presents the related work and section 7 concludes the paper.

2 Background: ER^{EC} Framework

The ER^{EC} framework facilitates designing e-contract processes, a mechanism that allows modeling, management, deployment and monitoring of e-contract. Figure 1 shows ER^{EC} meta-model for an e-contract (Karlapalem, Dani and Krishna, 2001), which allows us to capture the conceptual level details about the elements involved in a contract. The elements in the meta-model are parties, activities, clauses, budget, roles and payments. The ER^{EC} meta-model also facilitates to model the rules and exceptions that arise during e-contract execution.

For a particular contract, the data model can be instantiated from the ER^{EC} meta-model. This data model helps in modeling relationships among various entities in an e-contract and visualizing them. The execution of an e-contract involves fulfilling several activities. These activities are modeled as set of workflows. Since contracts are complex, an e-contract may involve a set of inter-related workflows. Hence, the workflows for an e-contract must be carefully specified and related to meet the contract requirements. The ER^{EC} framework facilitates conversion of e-contract to a set of workflows using the approach described in section 4.4. A detailed description of EREC framework can be found in the works by Krishna, Karlapalem and Chiu (2004) and Krishna, Karlaplalem and Dani (2005).

3 MTDC Approach

Contracts are voluminous text documents which are difficult to understand, visualize and execute. However, most of the contracts have many common statements and clauses (for example, payment related terms). In this paper, we develop our MTDC approach to make our earlier work on ER^{EC} meta-model for e-contracts pragmatic by building a prototype system to support a stepwise methodology to deploy e-contracts adhering to document contracts with appropriate human assistance wherever required.

Figure 2 shows MTDC approach and various components that are involved in this approach. The input to the toolkit is a contract document for which the domain specific Contract Type is known. This is given to the Named Entity extractor which finds the organizations, persons, dates and various entities present in the contract. We also take a list of Domain Specific Keywords, which was prepared offline and store both these keywords in the database. After this, the statements in the contract document are extracted and classified as clause, activity or exception using some rules and assisted by a designer. ER^{EC} Model is obtained from these entities. From this model, a collection of workflows involved in the contract are identified. After this, workflows are specified using Xflow specification and execute the workflows corresponding to the e-contract using Xflow workflow enactment sub-system. In the next section, we describe the various components of MTDC.

4 MTDC Components

MTDC has a Pattern Recognition System (Named Entity Extractor), Sentence Classifier, an ER^{EC} model component, Workflow Specification Component and Xflow Workflow Enactment Component.

4.1 Pattern Recognition Module

The Contract Pattern Recognition System identifies all the important keywords, which helps in extraction of clauses and activities, by two methods as described below.

4.1.1 Domain Specific Keywords

A contract can be classified to different categories or types depending upon its context. Few examples of these contract types are: "Employment Agreements", "Loan Agreements", "Stock Agreements", "Sales and Marketing Agreements" and "Lease Agreements". For a particular Contract Type a considerable number of documents are scanned and a dictionary of frequently occurring important keywords in these contracts is prepared. First the stop-Keywords are removed and stemming is performed for the rest of the terms.

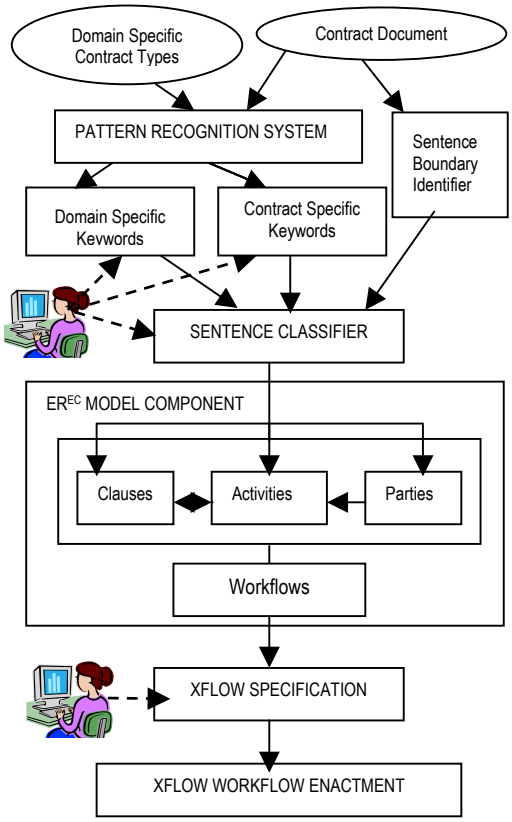

Figure 2: MTDC Approach

Zipf's Law (Rijsbergen, 2004) is applied to the remaining terms to prepare dictionary of domain specific keywords. We compute the total collection frequency of each of these Keywords. Let Keyword *w* be present in document *d* with frequency $Freq_{wd}$ and let the total number of contracts for a particular Contract Type be *N*. Then the total collection frequency is defined as

$$TotalFreq_w = \sum Freq_{wd} \text{ for } d=1, 2, ..., N$$

We establish two cut-offs, upper cut-off and lower cut-off. The Keywords exceeding upper cut-off were considered to be very common and Keywords below the lower cut-off are very rare. So, both of them may not contribute significantly to the Contract Type Specification. The cut-offs can be established on trial and error basis. In our case study, upper cut-off is used as 10-20% of the Keywords obtained and lower cut-off is 5-10% according to Contract domain. The designer can set the values based on his/her experience and type of the contract. All these Keywords grouped by their Contract Type are stored in database for further use. These Keywords are termed as 'Domain Specific Keywords' as they are different for each domain of contracts.

The keywords identified in this step are useful for realizing the sentences corresponding to activities, clauses, and parties from the contract document.

4.1.2 Contract Specific Keywords

A general named entity extractor is used to identify named entities (persons, locations and organizations), temporal references (date and time) and certain numerical expressions (payment, quantity) from the textual contract. This named entity extractor uses both syntactic and contextual information. The context information is identified in the form of POS tags of the Keywords and used in the named entity rules.

All the person names and organization names are candidate Party names for the contract. The dates can be the date of start of the contract, termination of the contract or date on which some important activity/ clause should be satisfied. The numerical expression can be the payment, which has to be made by one party to another or the quantity of goods to be supplied from supplier to retailer. Hence, all such keywords play significant role in identifying the elements of an e-contract.

We extract and store such Keywords in the database for every input contract. Unlike 'Domain Specific Keywords', these Keywords are specific to the contract and hence different for different contracts in same Contract Type. We term these Keywords as 'Contract Keywords'.

Domain Specific Keywords combined with Contract Specific Keywords are the 'Contract Keywords', which are important for visualization of the contract. The designer can add more Keywords in the database, which are not captured from contract document, but important in order to specify the contract.

4.2 Sentence Classifier

After extraction of Parties participating in the contract, we need to extract clauses and activities to be able to design ER^{EC} data model. Since, contract clauses have lot of ambiguities; the text document is converted into a canonical form by removing the ambiguities such as same party referred by different names.

Second, a Sentence Boundary Identifier is used to separate all the sentences in the contract. A unique Sentence ID is assigned to all the sentences obtained. We also assign a unique Keyword ID to all the Keywords present in 'Specific Keywords'. A (Sentence ID, Keyword ID) pair is stored in the database, if the Keyword occurs in the sentence. The sentence and keyword pairing is used for activity/ clause extraction.

4.2.1 Conversion of Contract to 'Canonical Form'

The input for the toolkit is a text contract, which is specified in natural language without any specific underlying structure. There can be some naming ambiguities present in the contract. To ensure that, we

proceed with validated elements specified in the contract human intervention is necessary. The keywords obtained from the named entity tagger are presented to the user for verification.

In a typical contract, a party can be given aliases or can be referred with different names at different places. For example, in a contract between Amazon.com and 3M, Amazon can be referred to as party 1 and the other as party 2. These names can also be used interchangeably. In general, the ambiguity can be more complicated than those mentioned above. For example, in a typical buyer-seller contract, the parties can be referred to both as receiving and providing party. When the contract quotes about seller sending product to buyer, buyer is the receiving party, whereas, when buyer makes payment seller is the receiving party.

To overcome all these problems, a designer is given the authority to finalize these names. The designer is shown the list of 'Contract Specific Keywords' along with contract parties highlighted for his/her reference. Party Names, Dates and Locations can be added or deleted if required. Merging of two or more entity names of same type is also allowed. For example, if party 1 refers to Amazon.com, then both these Keywords can be merged together to mean a single entity. In this case, every occurrence of these Keywords is replaced with the new name selected by the user. Finally, the contract is converted into a 'Canonical Form' without naming ambiguities. A check is performed by designer to see if there are other ambiguities and to rectify them.

4.2.2 Extraction of Clauses, Activities and Exceptions

Clauses in a contract are similar to conditions or rules. The Specific Keywords extracted in the section 4.1.1 helps in identifying candidate clauses in the contract. All the sentences which have at least one 'Specific Keyword' can be considered to be a potential clause. Since, this alone cannot be said as the criteria for a sentence to qualify as a clause; our semi-automated system will require a designer intervention to filter out some sentences, which are not a clause and also add some sentences from the remaining contract which are actually a clause.

A clause is fulfilled by successfully executing one or more activities. We use a list of 'Active words' such as buy and contact; to check if there is an action associated with sentence. All such sentences with atleast one 'Specific Keyword' will be considered potential activities.

Exceptions model deviations from fulfilment of the contract. An exception will generally be a sentence of the form 'if … then,' or it may have some negative Keywords such as fine, punishment, reimbursement, penalty and deduction of amount.

4.3 Mapping to ER^{EC} Model

E-Contracts are complex to understand and visualize. We model them using ER^{EC} Model proposed by Karlapalem, Dani and Krishna (2001). All the elements required for the construction of ER^{EC} Model namely parties, clauses, activities and exceptions have been identified as specified in section 4.2. The ER^{EC} data model for a particular contract is instantiated from ER^{EC} meta-model. Relationships can be modelled by providing a link between the entities of the contract and maintain these relationships in the database.

The ER^{EC} data model helps in monitoring and visualizing the e-contract, in addition to the design and development of e-contract system.

4.4 Workflow Specification Model

Workflows are the operational aspect of work procedures. They are used for automation of different activities in an e-contract. In workflows, information is passed between the various participants according to some rules or some goal to be reached.

```
Algorithm:
GetWorkflows ():
senID= List of all sentence IDs
Keywords = CurrentKeywords = NULL
Workflows = CurrentWorkflow = NULL
while (EachSenNotVisited):
  SelectedSenID= SelectSenID (senID which is not yet present
                    in Workflows)
  CurrentWorkflow.append(SelectedSenID)
  CurrentKeywords = {Specific Keywords in SelectedSenID}
  foreach s in senID:
    if NotPresent (CurrentWorkflow,s)==1:
      foreach c in CurrentWorkflow:
        if Similar(s, c) or NKeywordCommon(Keywords,s,n):
            CurrentWorkflow.append (s)
            CurrentKeywords = {Specific Keywords in s}

      Workflows.append (CurrentWorkflow)
      Keywords.append (CurrentKeywords)
      CurrentWorkflow = NULL
      CurrentKeywords = NULL
```

Figure 3: Algorithm for generating workflows

The toolkit supports e-contracts that can be specified by a collection of activity/ task oriented workflows, which may or may not be interrelated, and clauses and activities that can be indicated by various parties in online/offline mode. A clause can encompass the combined behaviour of a set of activities. An attempt is made to group all the activities, which should be performed in some predefined order to satisfy one or more clauses. Each of such collections will be specified as a workflow.

The algorithm shown in figure 3 produces groups of activities that share important linguistic elements and hence can form a workflow. First, we pick an activity at random, which is not yet present in any of the workflows obtained till now. The CurrentKeywords list has all the Keywords, which occur in any of the activities present in CurrentWorkflow. Now, we find another activity from the list senID, which is either highly similar to any of the activities in CurrentWorkflow or has at least *n* Keywords common with CurrentKeywords list for CurrentWorkflow. By highly similar, we mean a sentence which has minimum α % of its Keywords common with any of the activities selected till now in the workflow. This α can vary according to contract type. In our experiments, we use α as 50 and *n* as 2.

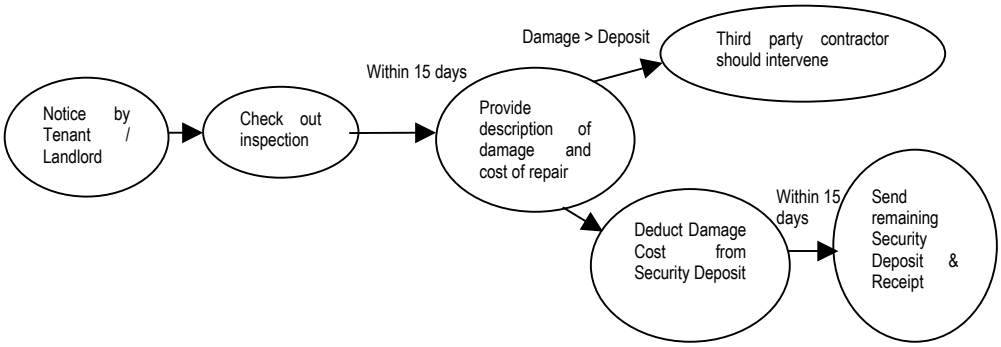

Figure 4: Workflow for activity 'Vacating'

4.5 Xflow Specification and Execution

XFlow is a J2EE platform for specifying, executing and managing business workflows**.** Xflow components are Models, Rules, Tools, Web Services, Events and Security.

The Xflow execution engine runs with JBoss and Tomcat containers. We generate the xflow specification for each of the workflows obtained in section 4.4 and deployed it on the Xflow Workflow engine.

5 Case Study

A Rental Unit Lease Agreement is an agreement, which specifies the obligations to be followed by Tenant and Landlord for the duration of tenancy. We have applied MTDC Methodology to a standard Rental Unit Lease Agreement. We have selected this contract for ease of explanation of our approach.

A Rental Contract mainly has three sections; first section describes the course of actions, which take place when the tenant decides to shift to the house, second section describes conduct of the tenant during the course of his stay and third section describes the various activities to be performed for the termination of the contract by tenant or by landlord. Some of the activities involved in this contract are:

1. Security Deposit: The Landlord keeps this deposit so that if tenant do not follow any provision detailed in the lease agreement then he can terminate the Lease Agreement and may apply all/part of deposit to the payment of accumulated rent or any damages caused by the tenant.
2. Inspection Condition Checklist: This is to provide an opportunity for Tenant and Landlord to inspect and agree upon the condition of premises. The checklist serves as a record from which to judge the return of security deposit after the termination of the agreement.
3. Other Duties: Some more activities to be performed by Tenant and Landlord such as maintaining cleanliness, maintenance of common areas, property alterations, subletting or assignment, storage areas, keys.
4. Notice: The Lease will include the form of notice to be provided by Tenant or Landlord for entry, repairing, termination of the contract, vacating of premises etc.

First, a dictionary of Domain Specific Keywords is prepared by scanning 30-40 Lease Agreements. Then, the contract document is given as input to the Pattern Recognition Module which identifies the Parties, Provinces and Dates. Those sentences which have at least one 'Specific Keyword' can be assumed to be a clause and are extracted. The elements required for the construction of ER^{EC} Model namely parties, clauses, activities and exceptions are identified. Relationships can be specified by mapping the clauses, activities and parties that occur in closely related sentences. A link exists between two entities, if they are related to each other. We find the similar sentences as described in section 4.3 and form ER^{EC} Model of the Rental Unit Lease Agreement.

From the ER^{EC} Model a collection of workflows is formed by running the algorithm described in Section 4.4. Some of the workflows identified are:

<Shifting of Tenant>: {'Decide term start date and Length of term', 'Decide Monthly Rent', 'Obligations of Co tenants', 'Appointment of realtor', 'Submission of Security Deposit'}

<Vacating>: {'Notice by Tenant/Landlord', '{Check Out Inspection}', 'Provide description of damage and cost of repair', 'Deduct Damage Cost from Security Deposit', 'Send remaining Security Deposit and Receipt of Security Deposit'}

<Check Out Inspection>: {'View Checklist prepared during the start of agreement', 'Check walls, floors, carpeting, curtain, doors, furniture', 'Record the damage', 'Return the keys'}

<Repairing>: {'Tenant reports the damage to Landlord', 'if damage in common areas (Landlord will pay) else (Tenant should pay)'}

Workflow for the activity 'vacating' is shown in Figure 4 and its Xflow specification in Figure 5.

For example, for the activity 'vacating', the workflow is obtained as follows:

Original Workflow got form the contract document

(In the case of a termination of the lease, the tenant is to vacate the premises, return all keys, remove all personal property, and leave the property in its pre-rental condition, except for normal wear and tear.) , (The checklist serves as a record from which to judge the return of the security deposit after the termination of tenancy.), (Landlord must provide Tenant(s) with (1) itemized listing of all deductions from security deposit and (2) payment of any amount due Tenant(s)), (If damages exceed amount of security deposit, a third-party contractor may be needed to intervene.)

Modified Workflow after Designer inputs

<Vacating>: {'Notice by Tenant/Landlord', '{Check Out Inspection}', 'Provide description of damage and cost of repair', 'Deduct Damage Cost from Security Deposit', 'Send remaining Security Deposit and Receipt of Security Deposit'}

This Xflow specification document is then given to a Workflow Management System (WFMS), which executes it. WFMS system allows the user to start a workflow, view the active workflows at a particular time, abort or deploy a workflow.

```
<xflow name="Vacate">
  <nodes>
    <node id="StartNode" type="Start"/>
    <node id="P1" name=" Notice by Tenant/Landlord"
          type="Process"/>
    <node id="P2" name=" Check Out Inspection" type="Process"/>
    <node id="P3" name=" Provide description of damage and cost
          of repair" type="Process"/>
    <node id="P4" name="Third Party Contractor should intervene"
          type="Process"/>
    <node id="P5" name=" Deduct Damage Cost from Security
          Deposit" type="Process"/>
    <node id="P6" name=" Send remaining Security Deposit and
          Receipt of Security Deposit" type="Process"/>
    <node id="EndNode" type="End"/>
  </nodes>
  <transitions>
    <transition from="StartNode" to="P1"/>
    <transition from="P1" to="P2"/>
    <transition from="P2" to="P3">
      <rule> [Integer.days &lt; 15] </rule>
    </transition>
    <transition from="P3" to="P4">
      <rule> [Integer.damage &gt; Security_Amount] </rule>
    </transition>
    <transition from="P3" to="P5">
        <rule> [Float.damage &lt; Security_Amount] </rule>
    </transition>
    <transition from="P5" to="P6">
        <rule> [Integer.days &lt; 15] </rule>
    </transition>
    <transition from="P4" to="EndNode"/>
    <transition from="P6" to="EndNode"/>
  </transitions>
</xflow>
```

Figure 5: Xflow Specification of workflow for activity 'vacation'

In our methodology, a contract document has been processed in a step-by-step manner to generate a set of workflows that can deploy the processes needed to enact the contract. In more complex contracts, the number of workflows and interdependencies among them will be more complex. As with any kind of methodology for mapping such requirements to an implemented solution that can be deployed, appropriate human/designer assistance is needed. Thus, the viability of our methodology aided by the toolkit has been established.

6 Related Work

Angelov and Grefen (2000) presented a survey on various projects and systems related to e-contracts. Benjamin and Poon (2003) developed a system *SweetDeal*, which allows software agents to create, evaluate, negotiate and execute e-contracts with substantial automation and modularity. This approach represents contracts in RuleML and incorporates process knowledge descriptions based on the ontologies. Chiu, Cheung and Till (2003) developed an e-contract deployment system based on multiple layer framework in which the e-contracts are modeled in UML and the implementation architecture is based on cross-organizational workflows using Enterprise Java Bean and Web services. Grefen et al (2000) developed a system *CrossFlow* for bilateral contracts. This system specifies the e-contracts in XML and allows template creation using pre-existent e-contracts. ER^{EC} framework presented by Krishna, Karlapalem and Chiu (2004) and Krishna, Karlaplalem and Dani (2005) centers on the ER^{EC} model that bridges between the XML contract document and web services based implementation model of an e-contract. Kwok and Nguyen (2006) describes a method for pattern extraction from PDF contract documents. Pattern Recognition system used for the toolkit can be accessed at http://search.iiit.net/~xtract/Xtract_pattern/index.cgi.

7 Conclusions

Many of the businesses are governed by the large number of contracts they get into. Contract management is a challenging task. There is a need for a toolkit that supports a top-down approach that starts with a contract document and ends with deployed e-contracts. In this paper, we present a methodology and the supporting toolkit built around ER^{EC} meta-model framework for conceptual modeling and enactment of e-contracts. Our toolkit uses natural language processing software and human intervention to specify system specific workflows by following our step-by-step methodology. The viability of our approach and utility of our toolkit is established by enacting workflows to support domain specific contracts such as rental agreements (detailed in this paper), marketing and sales. In future, we plan to incorporate commitment support for enacted e-contracts and further enhance the machine learning components to reduce the amount of designer intervention needed to deploy complex contract documents.

8 References

Benjamin N. Grosof and Terrence C. Poon. (2003): SweetDeal: Representing Agent Contracts with Exceptions using XML Rules, Ontologies, and Process Descriptions, *Proceedings of the 12th International Conference on the World Wide Web*.

Chiu, D. K. W., Cheung, S. C. and Till, S. (2003): A Three-layer Architecture for E-Contract Enforcement in an E-service Environment, *Proc. of 36th HICSS36.*

Karlapalem, K., Dani, A. R. and Krishna, R. P. (2001): A Frame Work for Modeling Electronic Contracts, *20th International Conference on Conceptual Modeling (ER2001)*, November 27-30, Yokohama, Japan, LNCS, vol. 2224, Springer.

Kwok T. and Nguyen T (2006): An Automatic Method to Extract Data from an Electronic Contract Composed of a Number of Documents in PDF Format*, Proceedings of the 8th IEEE International Conference on E-Commerce Technology*.

Angelov, S. and Grefen, P. (2000): B2B eContract Handling: A Survey of Projects, Papers and Standards, CTIT Technical Report 01-21, University of Twente.

Rijsbergen C. J. (2004): *The Geometry of Information Retrieval*, Cambridge University Press, New York, USA.

Grefen, P. , Aberer, K., Hoffner, Y. and Ludwig H. (2000): Crossflow: Cross-organizational Workflow Management in Dynamic Virtual Enterprises, *International Journal of Computer Systems Science and Engineering*, **15**(5): 277-290.

Krishna, R. P., Karlapalem, K. and Chiu, D. K. W. (2004): An ER^{EC} Framework for E-Contract Modeling, Enactment and Monitoring, *Data and Knowledge Engineering,* **51**(1): 31-58.

Radha Krishna, P., Karlapalem, K. and Dani, A. R. (2005): From Contracts to E-Contracts: Modeling and Enactment, *Information Technology and Management Journal,* 4 –1.