Enactable Electronic Contracts In E-Commerce: Models, Technologies And Architectures

Anna Burunova[1], Andrew Ponomarev[2,3], Nikolay Teslya[3]
[1]Saint Petersburg Electrotechnical University ETU "LETI", St.Petersburg, Russia
[2]SPIIRAS, St.Petersburg, Russia
[3]ITMO University, St.Petersburg, Russia
burunovaav@gmail.com, {ponomarev, teslya}@iias.spb.su

***Abstract*—In an e-service environment contracts are important for attaining business process interoperability and enforcing their proper enactment. An e-contract is the computerized facilitation or automation of a contract in a cross–organizational business process. Currently, there are many different approaches to create electronic contracts and support their execution. However, when creating a system based on e-contracts it is not always clear what model/technology/architecture to choose and what possible ramifications are. To address the problem, in this paper four most popular e-contract models (and systems supporting them) are taken and examined across several dimensions (e.g., e-contract model, e-contract lifecycle, solution architecture, supporting technologies). It appears that all of taken models have similar main line of actions and goals, however each solution has some additional specific features that is why technical implementation is very different.**

I. Introduction

Electronic contract (e-contract) is a machine interpretable specification, instantiated as a set of obligations that are fulfilled between parties, refused or waived when future events occur. E-contacts proved to be a useful tool for organizing correct work between parts of a distributed system (e.g., services). They afford to create, execute and manage interactions between different services without any human's involvement.

The idea of encoding mutual obligations between parties in a form of an electronic entity that can be used for monitoring and actual fulfilment of obligations (under that name of smart contract) was first introduced to the information systems research in 1997 by Nick Szabo [1]. With fast development of modern technologies, many different fields for application electronic contracts were found. One of them is electronic commerce, which this article is dedicated to. The main idea is that all stages of contract lifecycle are carried out and tracked by electronic systems, without or with minimal human involvement.

However, despite a two-decade history there are still many practical and research questions. What formal techniques should be used to encode and process electronic contract statements? How to ensure the consistency of a contract? What protocols should be used to negotiate about terms of the contracts? What current technologies and can support the creation of a distributed system based on electronic contracts.

The aim of this article is to review the most popular electronic contracts' architectures and principles of work to identify common features and ideas of current and future development. It should also be noted that to limit the scope of the paper we explicitly focus on the declarative contracts, not touching the burgeoning area of imperative contracts (e.g., those referred to as smart contracts in the context of distributed ledger [2]). In particular, we have found following approaches to implement enactable electronic contract models in electronic commerce scenarios: [3], [4], [5] and [6].

The rest of the paper is structured as following. Section II briefly introduces existing e-contracts models and frameworks used in the area of electronic commerce. Section III derives common core elements of each model and analyzes them.

II. Models of electronic contracts.

A. Secure e-contract

The main idea of a secure e-contract model proposed in [3] is implementation of the third party organization Electronic Contract Record Center (ECRC), which is aimed to provide security of e-commerce and keep local government informed about active transactions. ECRC can be represented by the government or any other organization which is authorized by the government, as its responsibilities are providing fairness, security and legitimacy and protection from violations, as there is always a third party, which is able to punishments, if one of the sides does not follow their duties. Also this method involves digital signature, as one of protection ways.

Technically ECRC is a web site server and an information platform, which provides publishing contract templates, downloading contract templates, uploading the valid contracts from clients, querying the records of clients' contracts, providing the existing contract documents as evidence and providing kind of statistic reports to government. Every active e-contract should be sent to the ECRC right after signing by all parties.

Main facilities to establish an e-contract are following: e-contract system and ECRC. The quantity of e-contract systems depends on number of parties, because it can keep private information. E-contract systems are used to download contract template from ECRC, create a contract with input information based on the template, signing signature on the contract, sending the contract to other parties, receiving and reading contract sent by other parties, copying the valid contract and sending it to the ECRC.

ISSN 2305-7254

B. Three-layer contract

Authors of [4] describe the model from three sides: document layer, business layer and implementation layer. It affords people of different jobs to use the most appropriate description. Document layer defines parties, involved in the contract, their roles, activities and clauses. Business layer depicts e-contract as a business process, which consists of rules (specify contract's clauses), events (trigger rules), actions (capture the activities, consequences and roles of the parties in each case) and entities (set of data objects). Implementation layer contains action implementation and cross-organizational event interface.

As contract defines responsibilities and duties of the involved parties of a business process, apparently, one of the hardest problem is solving exceptions, which occurs during the partnership. Clauses are supposed to fix that and regulate relations between the parties. In this model they are divided into three types: obligations, permissions and prohibitions. Obligations define business actions, which should be performed by a deadline. Prohibitions are used to prevent unwilling (prohibited) actions. Permission is a temporary allowance to perform otherwise prohibited actions, they often identify duration. In order to let computer do the monitoring job and find exceptions, clauses are transformed into event-condition-action (ECA) rules, the method of event-driven computing, which triggers actions based on current events and existing specific conditions. So when an event occurs, it is checked on its suitability to a certain condition, and after this appropriate action is called.

The architecture of this model doesn't require a central facilitator or moderator, because e-service provider of each party hosts following systems and subsystems: database with event repository, event subscribers list and business entities, contract enforcer, contract enactor, event adapter, timer, external web service interface. All of them are responsible for the certain options: contract enactor – performing regular business activities for service contract enactment, contract enforcer – detecting contract breaches and then triggers relevant business actions, event adapter – collecting internal events from the contract and external events from the external Web service interface, publishing and subscribing them, event enforcer – accepting the structure of transformed and filtered collected events, timer – generating temporal events.

C. ER^{EC} model

In an ER^{EC} model, proposed in [5] (and further developed in [7]–[11]), a contract specifies how it will be executed, the restrictions on the parties involved, and payment/delivery terms. All relationships among the activities parties and clauses are strictly identified and are constantly checked against violations. To define all of them Activity-Party-Clauses (APC) constructs are extracted from an XML-contract document. They describe interrelations between activities, parties and clauses, and make detecting of exceptions possible.

Payment transactions are always between the parties and they should be processed with a great attention. In order to monitor the transaction commit and to maintain the log, Active Commit Diagrams (ACD) are used. They represent and monitor the sequence of atomic activity transactions and keep the log of activities which are to be carried out. Commission of every following activity is possible only if the current one is completed successfully, otherwise the process will be rolled back to the pre-determined point. An activity transaction is said to be committed if all its atomic transactions are committed. It is worth to say that rolling back isn't the only method in case of an error, also failure compensation, alternative activity, time-base retry or re-execution can be applied. Anyway each case is individual and should be defined beforehand according to the expectations.

ECA rules are also used in ER^{EC} model, they help to monitor and execute the contract, by searching for a specific activities or behavioral-related works to trigger activities or actions, like if-else, contract violations and etc. Authors suppose that the combination of ECA and ACD provide the safety of transaction and the way the way to deal with the violations, when they occur.

The ER^{EC} is also able to adapt to run-time changes, connected with contract updates, exceptions or failures.

D. Policy-based e-contract

A policy-based e-contract model [6] is markedly different from previous ones. Firstly, because it uses web-agents, secondly – here authors describe B2C e-contract model, which has its own specialties to mention. For example, while communicating with consumer, it is necessary to have more trustful relationship, as a consumer always wants to have attention, feel safe and support, otherwise there is a risk of him leaving to rivals. To make this real, all approach is based on web-agents as they are capable to study from themselves and other parties, make decisions based on previous experience. Each agent has its own policy – a set of data and rules, which exactly determines its behavior and abilities.

Authors consider e-contract as a set of mainstream goals, each of which has a number of sub-stages, often divided into steps. Any goal's achievement goal implies negotiating process between the parties, which is aimed to be successful only if all goals completed successfully. For this reason, negotiating process is sequential, if the current goal isn't satisfied, it is impossible to go forward.

Each participant, joining such kind of relationship, expect them to be trustful and successful. To make this real a workflow for successful e-contract negotiation process was proposed. It consists of 11 issues, that are considered most important to pay attention to in order of priority while discussion. Also it is possible to estimate the importance of the particular goal, the final result and the quality of work accomplished for each specific case, using special formulas. After completed negotiation process, parties are moving to implementation of identified tasks and goals. Each successful contract is stored in the system and can be used once again in the future.

III. Comparison of existent models

As it was previously said, the main idea of the article is to analyze the most common electronic contracts' architectures. To achieve this goal, the previous articles on this theme were

examined and core elements of each model were highlighted, it can be seen that they are similar in each models, the differences are only in realization. To visualize information received, the table was created (see Table I). Let's now pay attention on each basic element and consider their role and implementation in each case.

TABLE I. COMPARISON OF THE EXICTING E-CONTRACT MODELS

	A secure e-model	A three-layer model	An ER^{EC} model	A policy-based model
Solution Architecture	Centralized system. Represented by ECRC (Electronic Contract Recorder Center), which gives the possibility to publish and download contract template, upload the contract, query the records of clients' contract, provide the recorded contract as evidence and provide kind of statistic report to government. Also each party should have their own e-contract system to be able to correct contract, sign contract, send to each other and to ECRC All contracts, signed by all sides should be uploaded in ECRC.	Distributed system. Each party hosts an e-service provider, which hosts all necessary technologies. They are: contract enactor, contract enforcer, event adapter, external Web service interface and Timer. E-service providers communicate with each other by web services or Enterprise Java Beans (EJB)	Distributed system. Each party should have: - Relational Database - Application Specific Components - Web Service Server - E-ADOME workflow Engine.	Distributed system. Represented by intelligent agents.
Layers	-	Document Business Implementation	Document Conceptual Logical Implementation	-
Entities	Contract, parties, exchange value, clauses, activities, signature	E-contract template, contract clauses (obligation, permission, prohibition), template variable, accepted value, parties	Contract, clauses, activities, parties, exceptions, roles, budget, payments	-
Lifecycle	1) Contract template (is published by government) 2) Draft contract (isn't signed and is used for negotiations, can be modified) 3) Middle contract (signed by half of the parties, isn't possible to modify) 4) Valid contract 5) Accomplished contract 6) Terminated contract 7) Revising contract 8) Canceling contract	1) Business information exchange 2) Contract negotiation 3) Contract enactment 4) Contract enforcement	1) Contract preparation 2) Contract negotiation 3) Contract fulfillment	
Obligations determining	There is an input information in downloaded template. It can be modeled, if need exist. Obligations come into play, when all parties signed the contract.	ECA-rules	ECA-rules, APC constructs, IF–THEN–ELSE constructs	JESS 2008, Semantic Web Rules
Commitments enforcing	-	Contract enforcer	Activity Commit Diagrams (ACD), Contract Enactment Monitor	Defeasible logic
Supporting functions	-	-	Constant comparing payments with the current budget.	Strategy evaluation, Performance evaluation, Negotiation workflow
Protection against violations	E-signature All e-contracts are recorded and preserved by third party ECRC, represented by government, which is aimed to provide legal templates to make an e-contract and to make an arbitration, if something happens.	If the occurred event or exception happens, the publish Web service is invoked by the event adaptor. And then it sends to the valid subscribers a notification by different kinds of protocols like e-mail, fax and etc.	ACD drive the workflows execution. Any mistaken execution in an activity leads to rolling back to a particular state. The corresponding workflow is committed only if all the activities are completed. The publish Web-service notifies relevant business partner about the occurred event or exception by different kinds of protocols.	-
Technologies	ECRC system, UML	Web services, Enterprise Java Bean (EJB), UML, XML schemas	XML, Workflow Management System (WFMS), software components, Web Services, E-ADOME, Enterprise Java Beans (EJB)	EMERALD, RuleML, RDF model, AYPS

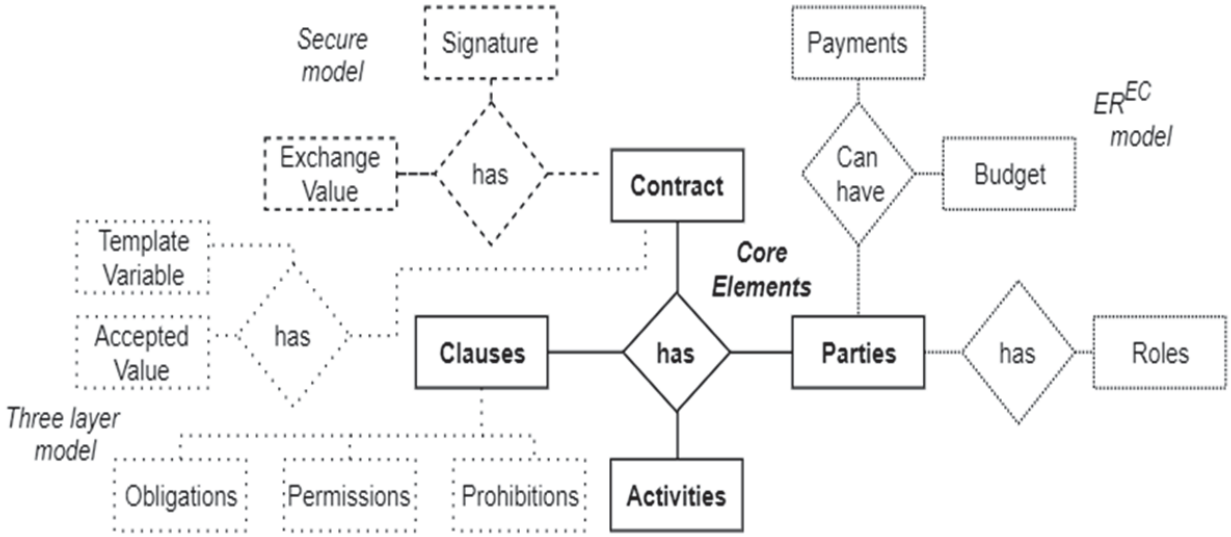

Fig.1. Generalized ER model

A. Solution Architecture

The architectural decision of the secure e-model [3] seems to be pretty simple, as there is only ECRC server, where parties can upload and download contracts. This model does not imply tracking the implementation, so if some violations occur one of the party should inform the ECRC's representative, which is supposed to solve the conflict based on the uploaded contract.

Three-layer [4] model contract contains such components as contract enactor and contract facilitator. It means that the machine can do enacting and facilitating processes by itself, which makes the full process easier. However, contract enactor and facilitator should be set on each party's computer and deal mostly with internal events, although events can also be transferred among collaborating parties, but only if it has been agreed beforehand.

ER^{EC} [5] framework contains shared contract enactment monitor. It triggers actions based on the events, received from all parties.

Policy based model [6] has intelligent agents, which are supposed to contract and agreement efficiently by themselves.

B. Layers

It is supposed that layers' existence affords to represent certain model more fully.

Layers of the three-layer [4] model and ER^{EC} model are quite similar. The only difference is that documental layer of the second one makes it real for person without any technical background to look through contract and understand the main aspects of the deal. While the same layer in three – layer model looks more like the conceptual layer of the ER^{EC} [5].

Accordingly, it is feasible to create a model, which describes both previous layered architectures. It consists of three layers. The first one describes current interrelations between parties that are under negotiation, such as obligations, permissions, prohibitions, exceptions and so on. All technologies, which are used to implement and track this, are outlined in the second layer. And the last one contains software components.

C. Entities

As it can be seen, some of the entities, like clauses, activities and parties, are similar for all models, which proves the same core ideas. Still each of the model has its own subjects, emphasizing specialties.

The secure model [3] has "exchange value" and "signature" entities. "Exchange value" contains all information about exchange products or service, full collaboration process and payments. It is the most essential part, and should be carefully detailed in order to make the proceedings easier in case of any mistake. "Signature" requires electronic signature from each party to avoid falsification of electronic documents.

Special objects of the three-layer model [4] are "template variable", "accepted values". "Template variable" describes parameters of a contracts that has to be agreed (usually, by negotiations). Agreed values of the template variables correspond to "accepted values". Also the "contract clause" entity consists of three parts: "Obligation", "Permission", and "Prohibition". Each of them "encourages" drafters to pay attention, so it can help to avoid some mistakes or misunderstandings while drafting or executing contract, as more cases are considered.

The ER^{EC} [5] model has such entities as "roles", "activities", "payments" and "budget". "Roles" describes numerous functions of each party. "Activities" is a following clause after successful execution one or more activities. Entities "payments" and "budget" help to realize payment process, as budget is constantly controlled during the payments not to create the situation, when spending is higher than current budget.

Fig. 1 is aimed to visualize and sum up this paragraph. It depicts relationships between common objects and extra entities, which is inherent in particular model.

D. Lifecycle

Here we can see, that some stages are also similar in one or another way, however let's pay attention on differences.

The secure model [3] is the only model that provides contract terminating during fulfillment, to make this real at least one party's signature is needed. If all parties sign terminated contract, then it can be revised or canceled.

The three-layer model [4] has contract enforcement stage, which ensures in monitoring and handling of breaches by different techniques during contract's execution.

There is a process of authorizing in the policy-based model, the main idea of which is to reuse previous information about clauses and events' patterns in order to let parties leverage new e-contracts. Also it considers the possibility of renewing an expired contract.

In general, any e-model's lifecycle can be described by the following stages: taking decision about making a contract, contract negotiation, contract enactment, fulfillment, contract completion with the possibility of renewal.

E. Obligation determining

As it was previously said the secure model [3] does not have any electronic surveys, so all obligations are taken from a common template or added while negotiation on the natural language.

The tree-layered [4] and ER^{EC} models [5] do have electronic monitoring, so they both use event-driven computing methods. The ER^{EC} not only defines exceptions with ECA rules, but also identifies the relationship among the activities, parties and clauses with the help of from Activity-Party-Clauses (APC) constructs. In the Table II transformation contract clauses into ECA rules on the example of a delivery company can be seen (the example is taken from [4]).

TABLE II. ECA RULES

Enforcement rule / Clause type	Event	Condition	Action
Obligation	*onDay(deadline(BAO))*	*NOT occurred(BAO)*	*raise(exception(BAO))*
Prohibition	*onOccurred(BAO)*	*prohibitionCondition (BAO)*	
Permission		*NOT permitted(BAO)*	

Here, BAO (business action object) represent an object that contains a business action, execution of which triggers the object creation. From this table the following ECA rule can be taken:

E: onDay (deadline(BAO))
C: NOT occurred (BAO)
A: raise (exception (BAO))

In this example system constantly checks the deadline of delivery, and if it is expired, some actions is triggered.

Speaking about APC constructs, they are extracted from the contract documents and are written in XML. The APCs consist of three sets of tags: the parties involved, the activities involved and the clauses in the contract. Tags are provided for exceptions raised during the execution of their respective activities, which is performed by the set of concerned parties bound to the clauses associated with them. The example of APC specification can be seen in the Fig. 2.

```
Party       <Party>
            <Party Number>…<Party Name>…+<Party>

Activity    <Activity>
            <Activity Number>…<Description>…<Start Date>…<End
            Date>…<Previous Activity>…<Next Activity>…
            <Parties Responsible>…</Parties Responsible>+
            <Clauses>…</Clauses>+
            <Exceptions>…</Exceptions>+
            </Activity>+

Clauses     <Clauses>
            <Clause Number> <Description>
            <Activity Number>…<Party Number>+</Clauses>
```

Fig.2. Example of APC construct (from [5])

In the policy-based model [6] agent's internal policy and behavior is represented by the Jess knowledge base and rule engine.

F. Commitments enforcing

In three-layered model [4] the process of commitments enforcing is released with the help of contract enforcer. It is usually based on the ECA rules, so each incoming event is checked on the certain condition and then trigger an action.

Apart from the ECA rules, ER^{EC} [5] model uses Activity Commit Diagrams, which are specially constructed from the APCs to organize execution of atomic transaction sequentially and keep the log of activities which are carried out. The example of an activity commit diagram for fund transfer activity is shown on in Fig. 3.

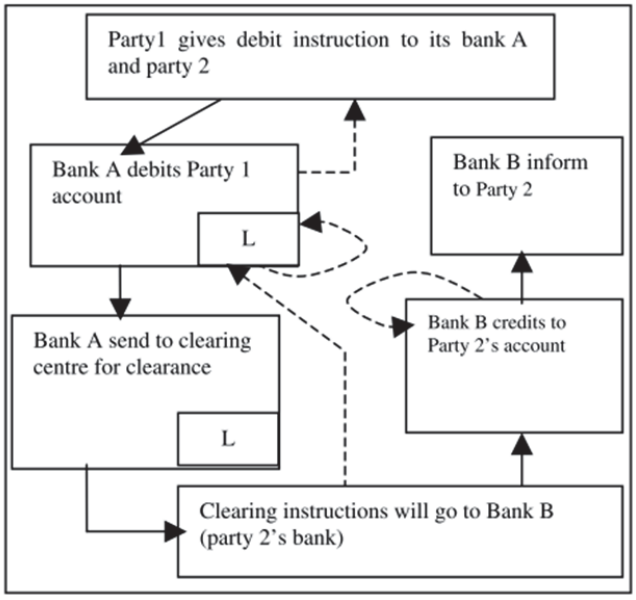

Fig 3. The example of ACD (from [5])

Here, the money is transferred from party 1 with an account in Bank A to Party 2, which has an account in Bank B. On the picture the solid arrows show the next transaction to be performed and the dashed arrows demonstrate rollback point, where transaction can be rolled back if it is not committed.

In the last contract model [6] defeasible logic is used. It consists of indisputable statements, represented in form of states of affairs or actions that have been performed and is the reason why electronic agents can draw reasonable conclusions from inconclusive information.

G. Supporting functions

To help the party to maintain cash balance and not to create a situation when company's budget is below zero, the ER^{EC} model [5] always compares current state of an account with necessary payments, and if the last ones cannot be paid now, certain actions are triggered.

The policy based [6] model uses negotiation workflow, consisted of eleven goals (aspects), which should be mentioned for sure while discussing in order to make a successful agreement. Also it offers the strategy to evaluate management, which is very important for getting useful feedback about something and using current experience in the future. Usually it requires strategy evaluation and performance evaluation. Strategy evaluation helps parties to negotiate strategically and maximize their efficiency value, by giving the mark to each of the goals and then using a special formula. There is also an equation that counts performance value.

H. Protection against violations

In the secure e-contract [3] protection is represented by the third party in the face of government or some other governmental organization. In order to avoid falsification a contract should have electronic signatures of all parties and to be uploaded in the ECRC, where it is kept in safe. All modifications are valid only if new contract is added to the system.

The three-layer [4] model informs its client about violations, impossible to handle through the Web-services.

The ER^{EC} [5] seems to be safer than previous ones, as it uses Activity Commit Diagrams. They execute all activities sequentially, and if the following step cannot be done, the system either waits until it is possible to continue or makes a rollback to the previous reliable state. The model also has an ability to notify about violations.

I. Technologies

Nearly all models use UML, two of them use XML schemas and web-services. Other technologies are different, as they fully depend on specialties of each model.

IV. Conclusion

This paper presents detailed review of the well-known e-contracts model. Each model individually was examined and then similarities in approaches of organizing models were found. By highlighting common features, it became possible to compare and contrast e-modes' architecture and working principles.

It can be said now that ER^{EC} model [5] looks more carefully designed and ready to be implemented than other models, those the questions of safety and security are not fully solved yet in none of existent models, so anyways big future work is required.

A policy-based model [6] looks really promising. It can highly improve the way of intercommunicating among parties, if electronic agents will do the biggest park of this work, based on the previous experience and special learning, however the question of implementation is still open here.

But it is also wrong to forget about the other models under review, though they are not that highly developed, they are aimed to make the process of contract execution safer and protect the system from different kind of violations

To sum up, the question of correlation each e-contract model to the specific situation in real life is open. It still cannot be said for sure, in which special occasions it is better to implement each model. Our future work is dedicated to this study and creating of guidelines on how to choose the most suitable e-contract representation and processing approach for the specific system.

Acknowledgment

The paper was partially supported by the Russian Foundation for Basic Research (grant # 19-07-00630), the Russian State Research # 0073-2019-0005 and the Government of Russian Federation (grant 08-08).

References

[1] N. Szabo, "Formalizing and Securing Relationships on Public Networks", *First Monday*, Volume 2, Number 9, 1997, Web: https://ojphi.org/ojs/index.php/fm/article/view/548/469.

[2] C.D. Clack, V.A. Bakshi, and L. Braine, "Smart Contract Templates: foundations, design landscape and research directions", 2016, pp.1–15. Web: http://arxiv.org/abs/1608.00771.

[3] Xiao Yu, Yueting Chai, Yi Liu "A secure model for electronic contract enactment, monitoring and management", 2009 Second International Symposium on Electronic Commerce and Security.

[4] D.K.W. Chiu, S.C. Cheung, S. Till, "A three–layer architecture for e-contract enforcement in an e-service environment", *Proc. of the 36th Annual Hawaii International Conference on System Sciences*, 2003.

[5] P.R. Krishna, K. Karpalem, D.K.W. Chiu, "An EREC framework for e–contract modelling, enactment and monitoring", *Data Knowl. Eng.*, vol. 51, pp. 31–58, 2004.

[6] K. Kravari, N. Bassiliades, G. Governatory, "A policy-based B2C e-contract management workflow methodology using semantic web agents", *Journal Artificial Intelligence and Law*, vol. 24, issue 2, pp. 93-131, 2016.

[7] K. Vidyasankar, P. R. Krishna, and K. Karlapalem, "A multi-level model for activity commitments in e-contracts," *Lect. Notes Comput. Sci.*, vol. 4803 LNCS, no. PART 1, pp. 300–317, 2007.

[8] N. Madaan, P. R. Krishna, and K. Karlapalem, "Consistency detection in e-contract documents," *Proc. 8th Int. Conf. Theory Pract. Electron. Gov. - ICEGOV '14*, pp. 267–274, 2014.

[9] P. Mohapatra, P. R. Krishna, and K. Karlapalem, "E-contract enactment using meta execution workflow," *Lect. Notes Comput. Sci.*, vol. 8186 LNCS, pp. 714–717, 2013.

[10] H. Jain, P. R. Krishna, and K. Karlapalem, "E-Contract Enactment System for Effective E-Governance," *Proc. 7th Int. Conf. Theory Pract. Electron. Gov. - ICEGOV '13*, pp. 344–345, 2013.

[11] H. Jain, P. R. Krishna, and K. Karlapalem, "Context-Aware Workflow Execution Engine for E-Contract Enactment," *Conceptual Modeling. ER 2016. Lecture Notes in Computer Science,* vol. 9974, pp. 293–301, 2016.