View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2000 Proceedings

European Conference on Information Systems (ECIS)

2000

On Specifying Contract Negotiations

Hartmut Wedekind
University of Erlangen

Follow this and additional works at: http://aisel.aisnet.org/ecis2000

Recommended Citation
Wedekind, Hartmut, "On Specifying Contract Negotiations" (2000). *ECIS 2000 Proceedings*. 45.
http://aisel.aisnet.org/ecis2000/45

This material is brought to you by the European Conference on Information Systems (ECIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ECIS 2000 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

search

On Specifying Contract Negotiations

Hartmut Wedekind
University Erlangen-Nürnberg
Informatik 6 (Database Systems)
Martensstraße 3, D-91058 Erlangen
Germany

Abstract -

eCommerce of the Business-to-Business (B2B) type requires comprehensive contract negotiations depending entirely on a contract schema, which must be developed in advance. Contract schemas are modeled according to bill-of-materials. Undefined contract parts are not allowed (Closed World Assumption). In the focus of the discussions are implications, i.e. the contract parts are not independent of one another. Negotiations are conducted in question-answer and reasoning dialogs. A task-logic interpretation of propositional logic due to Kolmogorov is introduced. Contract schemas are specified by XML. It is suggested to implement contract negotiations on top of a modular workflow system.

1 Introduction: Contract and Negotation Schemas

It is a main issue in the general framework of developing electronic commerce (eCommerce) to support the formations of sales contracts. This holds in particular for a business-to-business type (B2B) of contract with rather extensive contract negotiations. Companies are the contracting parties rather than private persons buying commodities on a catalogue basis (business-to-consumer, B2C). Three aspects are brought into focus in a vast literature: First the communications among parties (buyer, seller, third parties like notaries, auctioneers etc.) is outlined with all its technical implications. A second aspect is the contract and its schema as an evidential, editable and structured document. The third, and most difficult aspect pertains to contract negotiations and their schemata. Negotiation is the way to reach the goal, the contract. Contract negotiations are specifying procedures in order to determine legal, engineering and commercial parameters. From this definition the difficulties of our topic is apparent: "Specifying Contract Negotions" is a specifying of specifying, i.e. a meta-specifying. A meta-language level of conctracting requires two schemata: a contract schema, form wich a negotiation schema is derived.

Some short terminological remarks are needed at this point: A schema describes the universal aspects of an object and is required to understand its instances (tokens, occurrences, individuals) describing singular (particular) aspects. The notion of an object-language level (1. language level) and a meta-language level (2. language level) is explained by an example:

94051 is a zip_code

is a proposition on a object-language level.

"zip_code" is decimal-classificatory

is a proposition on a meta-language level. It is required in the logical analysis of languages that *zip_code* is put in quotations marks to indicate a schema as name and not a simple name like 94051.

Decimal-classificatory is a metapredicate, in general explained by a data model, i.e. a description of a decimal classification. Data models like the famous relational model are always meta-models, distinct from object models, i.e. in our case the postal zoning of a country. The naming 94051 in a postal address is a specification, its decimal classification is a meta-specification. Schemas like "zip_code" are sometimes called name space. Defined name spaces are very important in universal markup languages like XML.

2 Developing contract schemas

In the case of eCommerce a great number of contract schemas must be developed, each one is the basis of a negotiation schema, which in turn renders at the end a particular contract as an instance. At the very beginning some assumptions are made: Contracts are configured, i.e. they consists of defined components called pieces of text (pt). Within the framework of a contract schema everything is known in advance, nothing is new, which would have to be constructed. In general the term "Closed World Assumption" (CWA) is used for our environment. It is a principle applied in many sciences to facilitate the business of schematizing. What it means to abandon CWA is discussed at the end of the section on negotations schemata.

The datatypes of pt's – the components of a contract – remained undefinied in this paper. pt's may be composed by connectives: Conjunction C ($\wedge$), Alternative (Exclusive Or) A(∇) and Implication I ($\rightarrow$). A pt_i is a schematized or syntactical character string used in logic to express a form (gr. schema =$\sigma\chi\eta\mu\alpha$). Hilbert, the famous mathematician, called the characters "notifying characters" (Mitteilungszeichen) to indicate that terms of an object level exist, but are not referenced explicitly. Schematized or syntactical characters are used for example, if p_1, p_2, …, p_n are representing different predicates on a meta-level without introducing specific predicats like "green", "blue", …, "black" on an object level.

Arcs to the conjunction (C) and alternative (A) denote a part-whole relation. pt_1,… , pt_n are subschemas. The unsymmetrical implication node reads as follows: If subschema pt_i is selected, then select subschema pt_j. Often pt_i and pt_j are com-

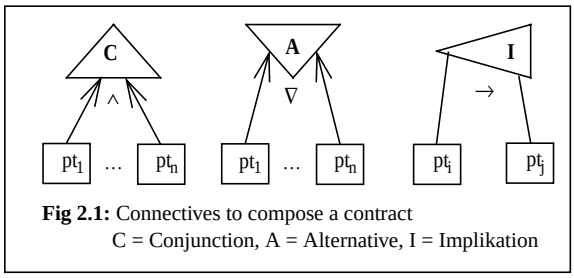


Fig 2.1: Connectives to compose a contract
C = Conjunction, A = Alternative, I = Implikation

ponents of an alternative node A. Then pt_i belongs to a premise node A_p and pt_j is a selection out of a conclusion node A_c. The significance of connectives is best demonstrated by showing an illustrative example: A contract may consist of a General Part (GP), specifying the parties, data, place of jurisdiction etc., an Engineering Part (EP) containing a schematized bill of material of a product type to be sold, and a Commercial Part (CP) on prices, discounts, and supply conditions etc..

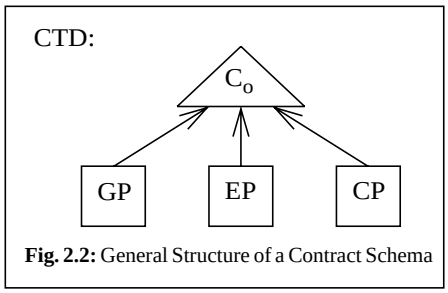
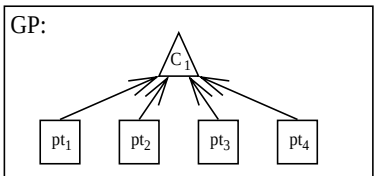


Fig. 2.2: General Structure of a Contract Schema

The conjunctive node C_0 in Fig. 2.2 is the name of a contract schema to be exemplified. In analogy to XML where the term "Document Type Definition (DTD)" is used, we coin the term "Contract Type Definition" (CTD). Certainly, a CTD may be represented in a XML specification.

Fig. 2.3 describes the General Part (GP). We assume that only one variant C_1 exists.

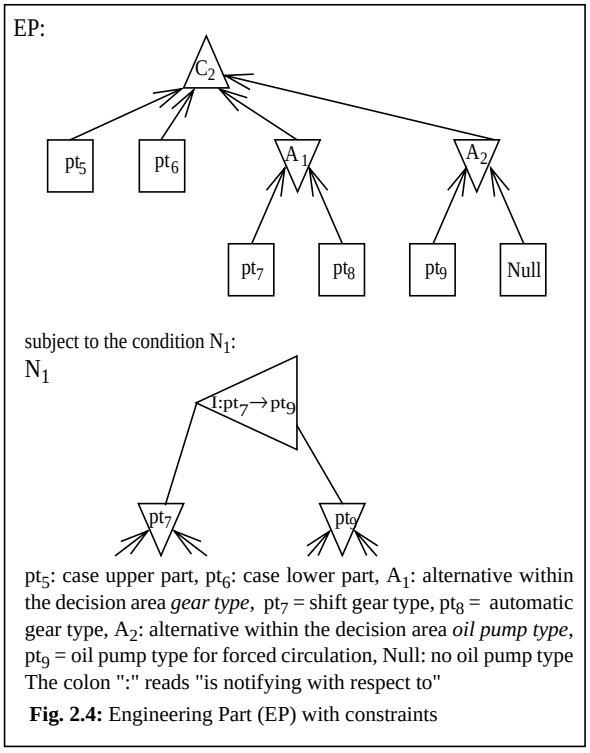


pt1: buyer, pt2: seller, pt3: place of jurisdiction, pt4: date. The colon ":" reads "is notifying with respect to"

Fig. 2.3: General Part (GP)

The Engineering Part (EP) is assumed to be more complex. Mechanical gears are supposed to be the subject of a contract schema.

Fig. 2.4 is a schematical bill of material (BOM) with variants. A_1 represents a mandatory variant (must variant). A decision has to be made between a shift gear type and an automatic gear type. A_2 is an optional variant (may variant). An oil pump type may be selected or not. Null represents no piece of text at all. The condition requires that from an engineering point of view shift gear types need an oil pump type. If an automatic type is chosen, one is free to select an oil pump type or not.



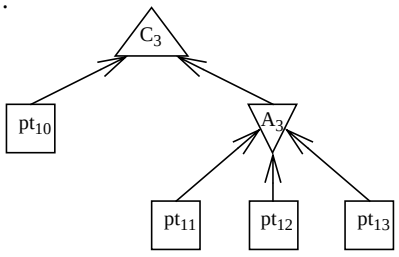
pt_5: case upper part, pt_6: case lower part, A_1: alternative within the decision area *gear type*, pt_7 = shift gear type, pt_8 = automatic gear type, A_2: alternative within the decision area *oil pump type*, pt_9 = oil pump type for forced circulation, Null: no oil pump type
The colon ":" reads "is notifying with respect to"

Fig. 2.4: Engineering Part (EP) with constraints

Within the engineering part only variants of a final assembly are discussed in contract negotiations. There is in general a "confidential attitude" as far as details are concerned. "Contract relations are relations of confidence." Distrust ruins any contract negotiations.

The Commercial Part (CP) is simpler than the Engineering Part. We assume constant supply conditions and three price options. It is not amazing that a huge literature in eCommerce is tackling the problem of competive bidding with respect to pricing. Contract negotiations are reduced to price setting and bargaining, which is a very narrow approach.

search



pt_{10}: supply conditions, pt_{11}: low price conditions, pt_{12}: regular price conditions, pt_{13}: high price conditions.
The colon ":" reads "is notifying with respect to".

Fig. 2.5: Commercial Part (CP)

The described example looks toyish. Indeed, it lacks the high complexity of practical cases. The purpose of the example is the description of schema variants in a part-whole relationship. If there are m_i variants for an alternative i and n alternatives, then there are

$$V = \prod_{i=1}^{n} m_i$$

schema variants, if constraints like in Fig. 2.4 are neglected. Take m=6 (constant) and n=6 then there are $V=6^6=46656$ schema variants; one of these is the schema of a committed contract selected in contract negotiations. Practical schemata are much more complicated in particular with respect to nested constraints, reducing the cardinality V, but increasing the complexity of contract negotiations drastically. A main function of computer supported contract negotiations is to guide the contract parties through a "jungle" of conditions.

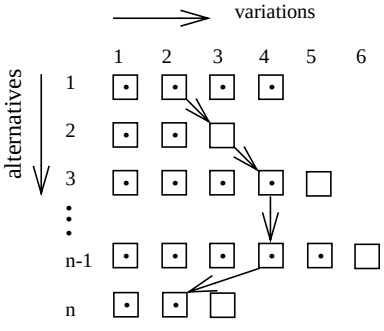


Fig 2.6: Combinations of schema variants as paths trough a network. Alternatives having a square without a dot are optional

3 Negotiation Schema

3.1 The significance of a negotiation phase

Contract negotiations are the mutual work of parties on ingredients of a contract schema. A negotiation schema is generated from a contract schema by adding order relations: "Negotiation_Schema = Contract_Schema + Order_Relations" is an easily remembered short version. In daily life order relations are called agendas. Designer of agendas can make only proposals. The final order is determined by the parties. The term "dialog" is another word for the parties' self-determination.

Form a generic point of view contract negotiations are workflows. Particularly, contract negotiations are procedures to create documents. The high complexity of all aspects of a general workflow [3] is out of scope. Three phases may be distinguished [6]:

1. Initiation Phase
2. Negotiation Phase
3. Execution Phase

In the first phase the study of catalogues and the selection of a representative of the other party (buyer or seller) is an important issue. It is not a trivial problem to find a competent dialog partner within a structural organization on the other side. In [2] the design and implementation of a generic selection operation getAgent ( ) is described in some detail. However, the second phase is by far the most difficult one, because schemas are subject to self-determination. The negotiation phase is important also with respect to the third phase, because its result, the contract controls the execution.

3.2 An initial ordering

An order relation – denoted by a semicolon ';' – is in our case an ordering of time (before or after, resp.) having the properties "reflexive", "antisymmetrical" and "transitive". The part-whole-relation describing the structure of a contract schema is an order relation as well, with the same properties mentioned above.

In our illustrative example it is straightforward to find an initial ordering:

CTD:	K0
R0:	AT; TT; KT
AT:	K1
R1:	pt1; pt2; pt3
TT:	K2
R2:	pt5; pt6; A1; A2
KT:	K3
R3:	pt10; A3

The General Part (GP) proceeds the Engineering Part (EP), because the GP contains basics of contract law and its ramifications. If one fails in the GP, one should quit the negotiations. The EP precedes the GP, because technical specifications are the foundation of commercial accounting.

The order relations are merely proposals. From a generalized view the parties may change the order at any time (see section 3.5).

For teaching purposes – not to reflect practical complexity – it is advantageous to describe the ordering by a well-known structogram used in elementary programming. In Fig. 3.1 we confine ourselves to the Engineering Part (EP)

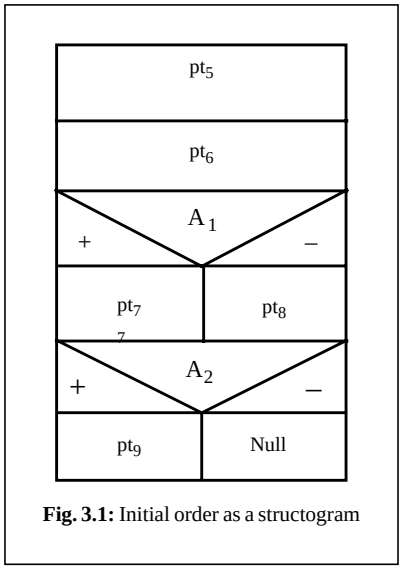


Fig. 3.1: Initial order as a structogram

3.3 Dialogs

The General Part (GP) in our example is particular simple, in order to exhibit an easy contract editing in a question-answer-dialog. One part (the buyer) asks, the other one (the seller) answers. It is easy to imagine how a cursor is jumping from one field to another. The question-answer-procedure is highly efficient, but only applicable in simple cases.

Question-answer-dialogs, suitable to get informed about alternatives, are distinct from reasoning (rational) dialogs. These types of dialogs are required if constraints are met in the contract schema using the full propositional logic (conjunction,disjunction, implication, negation). If nested constraints are encountered, then a dialog monitor is definitely needed to guide the parties through a "jungle". Reasoning dialogs were designed and implemented within the framework of the Dialogical Logic [4] and its pragmatic interpretation [1].

In Dialogical Logic a Proponent and an Opponent is introduced. The Proponent (i.e. the seller) is an acting person, who asserts something and gives reasons for a subject matter. The opponent on the other hand (i.e. the buyer) may doubt or agree. The roles of P and O may reverse.

In fig. 2.4 the constraint N_1 was introduced. In propositional logic it takes the form:

$$pt_7 \leq A_1 \rightarrow pt_9 \leq A_2 \,,$$

where '≤' is the sign of the part-whole-relation. In other words: It pt_7 in A_1 is chosen, then pt_9 in A_2 is chosen.

In form of a table the dialog with respect to the above implication looks as follows:

step	O	P
1.		Ass ($pt_7 \leq A_1 \rightarrow pt_9 \leq A_2$)
2.	Doubt ($pt_7 \leq A_1$)	
3.		Ass ($pt_7 \leq A_1$)

A dialog with respect to A_1 takes place

n - 1.	Ass ($pt_7 \leq A_1$)	
n.		Ass ($pt_9 \leq A_2$)

In step n-1 O agrees.

The dialogical sense of $pt_7 \leq A_1 \rightarrow pt_9 \leq A_2$ is the following: P is committed to assert $pt_9 \leq A_2$, if O is asserting (agreeing) $pt_7 \leq A_1$. This implication is called constructive (intuitionistic); it is distinct from the classical implication, where $a \rightarrow b$ is equivalent to $\neg a \vee b$.

A constructive implication suggests in our case the order A_1; A_2. It makes no sense to decide at first on A_2 and to jeopardize this decision when a component of A_1 is selected. Logic is the theory of riskless transitions (Lorenzen).

The dialog with respect to A_1 in the table above may be understood as a question-answer-dialog. O is asking for alternatives and P answers. Generally reasoning dialogs are interspersed by question-answer-dialogs. Notice that every information about instances belonging to a schema are subject to a question answer-dialog.

3.4 Negotation Logic as a Task Logic

At the beginning the author would like to appease the reader. There is no new logic called negotiation logic. We just go back to an interpretation of logic due to the famous mathematician A. Kolmogorov [11]. In 1932 Kolomogorov arose the interest of logic experts by specifying a "task logic", i.e. not a logic with propositions, but with problems or tasks as basic elements. The implication "$a \rightarrow b$" means: Reduce the solution of task b to a solution of task a[1]. The implication of our example

search

$pt_7 \leq A_1 \rightarrow pt_9 \leq A_2$

appears in a different light. "$pt_7 \leq A_1$" is a problem to be solved by negotiations prior to the problem "$pt_9 \leq A_2$". Komogorov's interpretation of an implication was the starting point for Lorenzen [12] to develop his constructive logic. Lorenzen did not agree with the undefinied term "task". In our context, however, the term "task" is well defined; it is the treatment of a piece of text by contract parties in a dialog.

We would like to show how nested implications in a contract schema are built (fig.3.2).

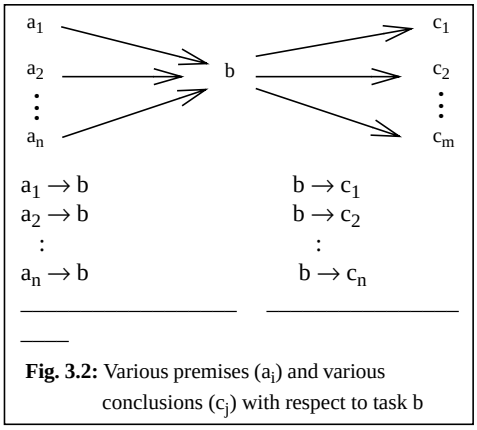


Fig. 3.2: Various premises (a_i) and various conclusions (c_j) with respect to task b

In case of common premises (a_i) it is simpler to introduce the logical OR($\vee$): $a_1 \vee a_2 \vee ... \vee a_n \rightarrow b$. Common conclusions ($c_j$) lead to logical AND's ($\wedge$): $b \rightarrow c_1 \wedge c_2 \wedge ... \wedge c_m$. If an implication ($\rightarrow$) is introduced in a positive task logic, then conjunctions ($\wedge$) and disjunctions ($\vee$) are straightforward. Thus a nested situation is described by simple means. If a conjunction is available then task b in fig. 3.2 may be looked upon as a task composed of subtasks: b $\leftrightharpoons$ b1, b2, ... ,bp. Thus we can write: $b1 \wedge b2 \wedge ... \wedge bp \rightarrow c$. There are well defined rules how to treat conjunctions and disjuncations in a dialog. It is beyond the scope of this paper to show the dialogical treatment of "jungles" like the one in fig. 3.2.

3.5 Classification of Dialogs

There are three broad classes of dialogs:

A) Question-Answer-Dialogs

B) Rhetorical Dialogs

c) Rational Dialogs

Question-Answer-Dialogs are not further classified in this paper.

Very close but substantial different from Rational Dialogs with a reasoning component are Rhetorical Dialogs. A dialog is called rhetorical or persuasive (in an evil-minded sense) if an agreement is reached by appealing – against one's own knowledge – to a naive prejudice of some person (Friedrich Kambartel). A well-known example of a Rhetorical Dialog is the following story: In a sales situation a farmer buys a milking machine and gives away his last cow for payment. A Rhetorical Dialog is performed in a simple minded sense, if the persuading person has no knowledge either. Rational Dialogs are non-persuasive.

From a technical point of view Rhetorical Dialogs were designed and implemented very early. One landmark in the development was the Issue Based Information System (IBIS) at MCC by the end of the 80th [13]. It is a hypertext system supporting rhetorical dialogs even in a real time environment (rIBIS) [13]. IBIS systems may traced back to two historical sources. Once to Horst Rittel (1970) with his idea to structure dialogs in state-transition-diagrams and secondly to the legendary Vannegar Bush (1945), the "father" of hypermedia systems. Not linear texts but rather hierarchically structured texts (hypertext) support our associative thinking more adequately. Hyptertext systems today are elements of every-day life. IBIS has shown that it is an ideal basis to support dialogs. Every node in a directed hypertext graph is a dialog step in a formal or natural language. A directed link between two nodes represents a speech action. IBIS is confined to assertions.

A Rhetorical Dialog may be formalized as follows (fig. 3.3).

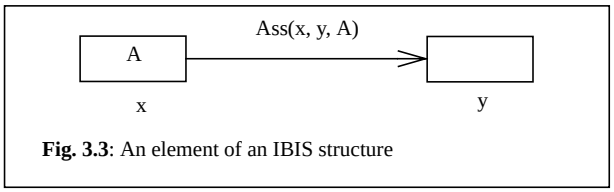


Fig. 3.3: An element of an IBIS structure

Fig. 3.3 describes that a source node x asserts a proposition A with respect to a node y. A may be arbitrarily complex. x and y are variable names for nodes. If the nodes are persons then x is called the author's and y the addressee's node. Fig. 3.3 represents the speech action: "x asserts with respect to y that A". "Assert" may be viewed as a three-place predicate: Ass(x,y,A).

Opposed to a Rhetorical Dialog a Rational Dialog is specified by a five-place predicate (fig. 3.4).

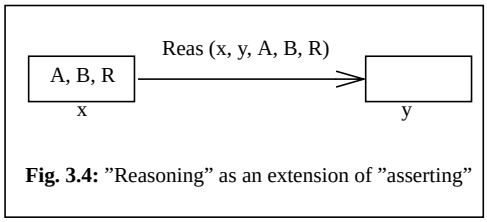


Fig. 3.4: "Reasoning" as an extension of "asserting"

In Reas(x,y,A,B,R) A is called conclusion, B the premise, and R a class of rules. If documents (D) are explicitely introduced,then a six-place predicate is given: Reas(x,y,A,B,R,D).

1. If negation ($\neg$) is available, then $(a \rightarrow b) \rightarrow (\neg b \rightarrow \neg a)$ holds as well.

Most important in contract negotiations are rational dialogs, although rhetorical ones can never be excluded. Rhetorical dialogs are in general not recorded and therefore fugitive. Advertisement is a monological category.

We procede now in classifying Rational Dialogs and its monitors referring to Gethmann and his classificatory work [1].

1) Productive (progressive), horizontal dialogs

The described dialog is called productive, because it is started at the beginning without any specification of an instance. The dialog procedes piece by piece progressively. The ordering (rule system) remains untouched. Thereby a horizontal direction is taken.

2) Reductive (regressive), horizontal

One starts at the end, i.e. there are pattern of complete contract schemata. Pieces of text are stepwise changed, if necessary.

Productive and reductive dialogs are entirely different. A productively working dialog monitor should be developed seperately from a reductive one.

3) Productive (progressive), vertical

The opponent (buyer) is allowed in a productive dialog, to attack the ordering. Thus the opponent may determine the "agenda".

4) Reductive (regressive), vertical

Like under 3), however in a reductive mode. Type 3) and 4) are leading to a new schema, because schemata are changed, if the ordering is changed. Ordering relations are only uncommitted proposals.

All four dialog types may be extended, if unknown ingredients are allowed. Dialogs without the CWA assumption should be dialog definite, i.e. there are definite rules guaranteeing a definite termination [4].

4 Implementation Issues

4.1 Implementing a Contract Schema

As mentioned in section 2 an arbitrary contract schema can be formulated using the extensible markup language (XML). The current paragraph sketches the structure of a possible document type definition (DTD) thus setting up a grammar for writing valid contract schemas. It is a general concept of XML (and SGML) that a new markup language - in our case for articulating general contract schemas - is created by defining a DTD that itself is positioned on the next higher language level. So the DTD to be developed in the following is part of the meta-meta level.

The necessary elements of the desired DTD can be seen directly in fig. 2.1: the leaves of the schema trees are pieces of text comprising arbitrary data, e.g. single clauses of a contract. Furthermore it can be thought of a special null schema for constructing optional alternatives as shown in fig. 2.4.

With regard to implication nodes it is necessary to add a unique identifying attribute (ID) to both, pieces of text and null schemas.

In the next step connectives like conjunctions, alternatives and implications are to be implemented. The first ones are very similar in structure: both contain at least two substructures which themselves can be pieces of text, conjunctions or alternatives again. For alternatives also the existence of one null schema is allowed (null schemas do not make sense as members of conjunctions). The notation for specifying these requirements in the DTD is a mixture of EBNF and regular expressions. Finally the implication consists of one premise and one conclusion both referring to a piece of text. Thus they are modeled as IDREFs, i.e. as references to IDs mandatory for each piece of text (see above).

After the definition of leaves and nodes of the schema tree a contract schema representing the root of our tree is the combination of at least one conjunction followed by an arbitrary number of implications. So the DTD developed above looks like this:

```
<!ELEMENT Piece_of_Text ANY>
<!ATTLIST Piece_of_Text
   Name CDATA #IMPLIED
   id ID #REQUIRED>

<!ELEMENT Null_schema EMPTY>
<!ATTLIST Null_schema
   Name CDATA #IMPLIED
   id ID #REQUIRED>

<!ELEMENT Alternative
   ((Piece_of_Text|Alternative|Conjunction|
      Null_schema),(Piece_of_Text|
      Alternative|Conjunction)+)>
<!ATTLIST Alternative
   Name CDATA #IMPLIED>

<!ELEMENT Conjunction
   ((Piece_of_Text|Alternative|Conjunction),
      (Piece_of_Text|Alternative|
      Conjunction)+)>
<!ATTLIST Conjunction
   Name CDATA #IMPLIED>

<!ELEMENT Implication EMPTY>
<!ATTLIST Implication
   Name CDATA #IMPLIED
   Premise IDREF #REQUIRED
   Conclusion IDREF #REQUIRED>

<!ELEMENT Contract_schema
   (Conjunction+,Implication*)>
<!ATTLIST Contract_schema
   Name CDATA #IMPLIED>
```

Using this meta-meta level DTD a meta level contract schema can be instantiated obeying to the rules explained above.

The following example represents the XML encoding of the contract schema developed in section 2 of this paper:

```
<?xml version='1.0'?>
<!DOCTYPE Contract_schema SYSTEM "ctd.dtd">
<Contract_schema Name="Contract_schema">
   <Conjunction Name="General_Part">
      <Piece_of_Text Name="buyer" id="pt1"/>
      <Piece_of_Text Name="seller" id="pt2"/>
      <Piece_of_Text
         Name="place of jurisdiction"
         id="pt3"/>
      <Piece_of_Text Name="date" id="pt4"/>
   </Conjunction>

   <Conjunction Name="Engineering_Part">
      <Piece_of_Text
         Name="case upper part" id="pt5"/>
      <Piece_of_Text
         Name="case lower part" id="pt6"/>

      <Alternative Name="gear type">
         <Piece_of_Text
            Name="shift gear type"
            id="pt7"/>
         <Piece_of_Text
            Name="automatic gear type"
            id="pt8"/>
      </Alternative>

      <Alternative Name="oil pump type">
         <Null_schema id="n1"/>
         <Piece_of_Text
            Name="forced circulation"
            id="pt9"/>
      </Alternative>
   </Conjunction>

   <Conjunction Name="Commercial_Part">
      <Piece_of_Text
         Name="supply conditions" id="pt10"/>
      <Alternative Name="pricing">
         <Piece_of_Text
            Name="low price conditions"
            id="pt11"/>
         <Piece_of_Text
            Name="medium price conditions"
            id="pt12"/>
         <Piece_of_Text
            Name="high price conditions"
            id="pt13"/>
      </Alternative>
   </Conjunction>

   <Implication
      Premise="pt7"
      Conclusion="pt9"/>

</Contract_schema>
```

The single parts of the contract schema are emphasized for better orientation. The distinction between the structure of the contract on the one hand and the implication(s) on the other can be seen clearly. Obviously implications can only be defined after the definition of the pieces of text they refer to. The implication in the example schema above indicates that if the piece of text called ’pt7’ is chosen, also ’pt9’ has to be included into the contract schema.

The contract schema formulated in XML in turn can be used as the basis of a contract that is located one language level below that of the schema. For that the XML document above has to be transformed into a DTD to be used as grammar for another markup language. It is not sure if this can be achieved using the extensible stylesheet language (XSL) that offers possibility to transform XML document trees. Anyway the transformation can be performed by an application program making use of the DOM API for accessing XML documents. But this process is out of the scope of this publication.

4.2 A sketch of Implementing Contract Negotiations

Non-routine contract negotiations are in general an intricate, sometimes tough business. Implications and Kolmogorov’s interpretation give rise to look upon contract negotiations as a very particular type of workflow. If this is accepted, then an impressive gate opens into a big arsenal of well-conceived implementation aids.

Workflow-Management-Systems try to span spheres of control over arbitrary complex working processes. Dataflow, controlflow, assignment of people and application software to processes, archiving, etc. [3] are just a few aspects to specify a schematized division of labor. Workflow Systems are represented by complex networks with many attributes.

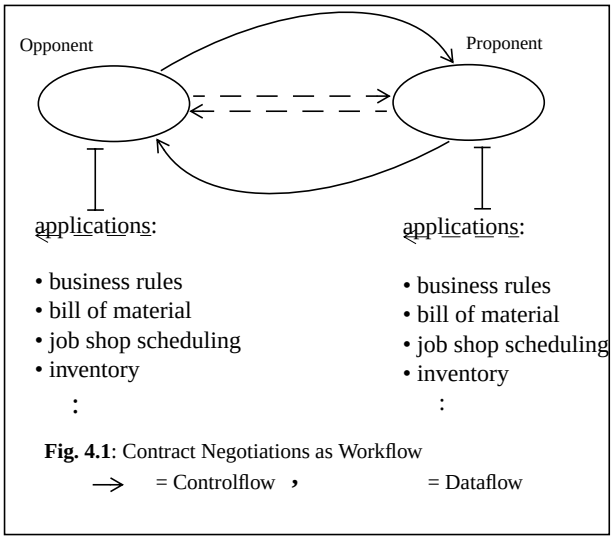


Fig. 4.1: Contract Negotiations as Workflow

search

Fig. 4.1 shows the rather simple version of an negotiation workflow.

As in all workflows, control and dataflow are separate. To fulfill a task in general a support of various application modules is needed.

At this time it is too early to go into more details and to present an archtiecture of a dedicated negotation workflow. However, on top of a aspect-modular system like MOBILE [3] the implementation is an achievable task. Modular decomposition pays off.

Conclusion

Although eCommerce may be considered as a young discipline two generations are distinguishable. The first generation is described by independent contract ingredients. Question-answer-dialogs suffice to conduct the negotiations. The independence may be achieved by factoring out restrictive implications. For every product a separate contract schema is provided which may result in myriads of schemas. The decision which product to select is out of system. In this paper we presented a system of the second generation with compact conditional schemas providing more selection support for the contract parties.

Acknowledgement

The author would like to thank Wolfgang Hümmer for his support in particular for his help regarding the XML specification.

References

[1] Gethmann, C. F.: Zur formalen Pragmatik der Normenbegründung. In: Mittelstraß, J. (Hrsg.): Methodenprobleme der Wissenschaften vom gesellschaftlichen Handeln, Suhrkamp Taschenbuch Wissenschaften, Frankfurt, 1979, S. 46-76.

[2] Jablonski, S., Schlundt, M., Wedekind, H.: Concepts and Implementation of Computer Supported Structural Organizations (to be published).

[3] Jablonski, D., Bussler, C.: Workflow Management. Modeling, Concepts, Architecture and Implementation. International Thompson Publishing, Bonn, 1996.

[4] Lorenzen, P.: Constructive Philosophy. The University of Massachusetts Press, Amherst, 1987.

[5] Milosovic, Z., Bond, A.: Electronic Commerce on the Internet. What is still missing? (http://www.isoc.org/HMP/PAPER/096/html/096.html).

[6] Merz, M.: Electronic Commerce. Marktmodelle, Anwendungen und Technologien, dpunkt verlag, 1999.

[7] Merz, M., Tu, T., Lamersdorf, W.: Electronic Commerce, Technologische und organisatorische Grundlagen. In: Informatik Spektrum, Band 22, Heft 5, Oktober 1999, S 328-343.

[8] Merz, M., e.a.: Supporting Electronic Commerce Transactions with Contracting Services. In: Internat. Journal of Cooperative Information. World Scientific Publishing Company, 1998.

[9] Tolkersdorf, R.: XML und darauf basierende Standards. Die neue Auszeichnungssprache des Web In: Informatik Spektrum, Band 22, Heft 6, Dezember 1999, S. 407-421.

[10] Wedekind, H.: Konstruktionserklären und Konstruktionsverstehen. In: Zeitschrift für wirtschaftliche Fertigung (ZwF), Band 84, Heft 11, 1989, S. 623-629.

[11] Kolmogorov, A.: Zur Deutung der intuitionistischen Logik. In: Mathematische Zeitschrift, Jahrgang 35, Heft 8, 1932, S. 58-65.

[12] Lorenzen, P.: Einführung in die operative Logik und Mathematik, Springer Verlag, 2. Auflage, 1969, S. 48 ff.

[13] Conklin, J. und Begeman, M.: gIBIS: A Hypertext Tool for Explanatory Policy Discussion, in: Proc. on CSCW'88, Portland, Oregon, Sept. 88, S. 140-152.