ELECTRONIC NEGOTIATIONS IN A HIGHLY CONCURRENT ENVIRONMENT

Willy Picard, Wojciech Cellary
Department of Information Technology
The Poznan University of Economics
Mansfelda 4, 60-854 Poznan, Poland
{picard, cellary}@kti.ae.poznan.pl

Abstract: In the context of economy globalization, the need for globally distributed negotiations involving a high number of negotiators communicating through the Internet becomes an important business issue. In such negotiations, the amount of information describing the negotiation process is too high to be easily understood by humans. In this paper, a negotiation support model adapted to highly concurrent environments is presented. The proposed model consists of a multiversion contract model and a multi-facet hierarchical analysis mechanism. The history of negotiation positions of different negotiators is modeled by a sequence of contract versions authored by them. Agreements between negotiators are modeled by shared contract part versions. The multi-facet hierarchical analysis mechanism provides synthesized views of the negotiation process. In this mechanism, mapping functions extract abstract objects corresponding to structured information concerning the negotiation process, which are further classified by a hierarchical classification algorithm basing on ultrametrics.

Key words: electronic negotiations, contract, versioning, hierarchical classification

1. INTRODUCTION

Negotiation is a fundamental act in business. Every business transaction is basing on a contract that has been previously negotiated. In the context of economy globalization, companies doing business with other companies all around the world need to negotiate at a global scale. Such negotiations are needed not only for multinational enterprises spread in many countries, but

also for small and medium enterprises, which are working more and more in an international environment.

However, classical ways of conducting negotiations are not well adapted to negotiations at the global scale. People involved in a negotiation process are used to personally meet to exchange information and to confront their interests and goals. Personal meetings are, however, costly in terms of time and money, as well as difficult to organize, in particular if negotiators work in different countries. In classical negotiations only a small number of participants are involved.

With the rise of Internet, geographical location of negotiators becomes unimportant. Internet allows a potentially unlimited number of negotiators from the whole world to negotiate on a given contract. Now, the problem is how to organize and manage remote negotiations conducted by a great number of negotiators.

An attempt to achieve this goal is to delegate the responsibility of the negotiation from a human negotiator to a computer. In such case we talk about "automated negotiations". The negotiation is said to be fully automated if negotiations are conducted by software agents without human intervention in the negotiation process. Research topics involved in automated negotiation are the following [6]:

- negotiation protocols defining types of participants, valid actions, negotiation states, and events that cause negotiation states to change [1];
- establishment of ontologies defined as agreements among the negotiators about how the negotiation objects are defined and what is the meaning of these definitions. XML Schemas [9][5] and UML [3][8] have been proposed as candidates to the design of ontologies;
- decision-making models that are used by software agents to achieve their goals.

However, in the case of multi-attribute contracts with both aggregable attributes (e.g. price, quantity, etc.) and non-aggregable attributes (e.g., legal clauses, appendices, quality clauses, etc.), automated, humanless negotiations are not a viable solution. Software agents cannot operate on non-aggregable attributes, because of the lack of semantics concerning these attributes.

With such contracts, negotiations must be conducted by humans. However, humans without any support are unable to deal with negotiations involving a high number of negotiators of a range of hundreds or more. The amount of data generated during such a negotiation process is too high to be understood by humans. Therefore, negotiation support systems that may facilitate massive negotiation processes conducted via the net, are required.

In this paper, we propose a negotiation support model adapted to highly concurrent environments. The proposed solution consists of a multiversion

contract model and a multi-facet hierarchical analysis mechanism basing on ultrametrics. The multi-facet hierarchical analysis mechanism provides synthetic views of the negotiation process. The paper is organized as follows. In Section 2, requirements for such a negotiation support system are given. In Section 3, a multiversion contract model that addresses the problem of scalability in highly concurrent environment is presented. In Section 4, a multi-facet negotiation analysis mechanism is discussed. Section 5 concludes the paper.

2. REQUIREMENTS FOR A NEGOTIATION SUPPORT SYSTEM IN A HIGHLY CONCURRENT ENVIRONMENT

As mentioned in the Introduction, in this paper we aim at providing contract negotiators communicating via Internet with a support system. Such a negotiation support system is particularly important in a highly concurrent environment, in which it is almost impossible to remember all the propositions made by a great number of negotiators.

The first problem arising when a negotiation support system is designed is the need to represent negotiation history. Storing conversation among negotiators is not a right solution, because of natural ambiguities and difficulties in intention interpretations. However, one may notice that a contract, which is under negotiation, is usually modified many times until the final agreement. The various versions of the contract reflect various propositions made by negotiators. Thus, a partially ordered set of contract versions represents the multi-thread history of negotiations. We propose to use the partially ordered set of contract versions as a basis of a negotiation support system.

In massive negotiations, in which the number of negotiators is high, and the number of contract versions is very high, a negotiation process is possible only if negotiators have an access to synthetic views of the negotiation process. A fundamental element of every negotiation strategy is the planning process ([7], pp. 40-51). The planning process is mainly basing on various analyses of the current status of negotiations. In highly concurrent environments, negotiators cannot conduct these analyses manually, because the amount of data to be analyzed is too high. Therefore, a *multi-facet analysis mechanism* is proposed to be integrated in the negotiation support system.

3. MULTIVERSION CONTRACT MODEL

A multiversion contract can be formalized as a set of points in a three dimensional space, denoted *C*. The space *C* is defined as $P \times V \times N$, where *P* is the contract part space, *V* is the contract version space, and *N* is the negotiator space. In Figure 1, a graphical representation of this model is presented: two negotiators N_1 and N_2 are negotiating a contract composed of two parts (P_1 and P_2). Negotiator N_1 has two versions of the contract containing two different versions of part P_1, but no version of P_2. Negotiator N_2 has one version of the contract containing parts P_1 and P_2, each one in a single version.

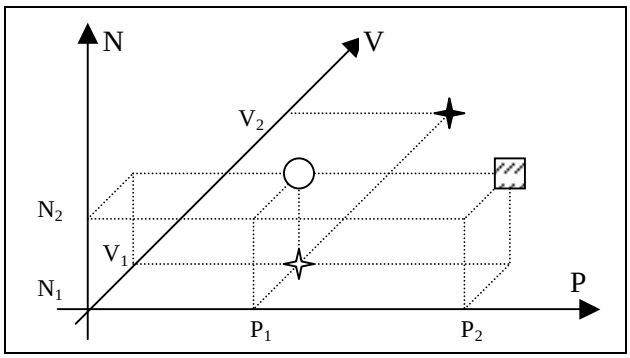


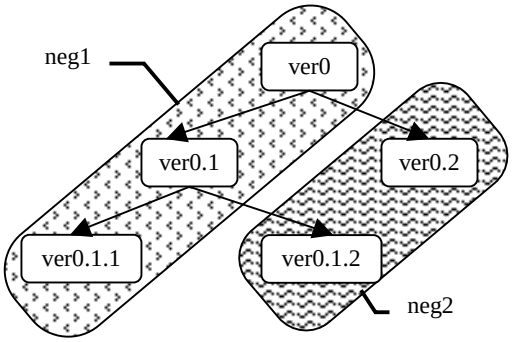
Figure 1. Graphical representation of space C.

The contract model proposed in this paper is basing on the multiversion database model presented in [2]. In the proposed model, a contract is multiversion. Each version of the contract is associated with only one negotiator, however a negotiator usually is an author of several contract versions. Each version of the contract represents one state of contract, as proposed by the negotiator that is its author. A sequence of contract versions authored by a negotiator represents history of his/her negotiation positions. It is worth to note that a contract version may not be complete from the beginning, indeed some its parts may be added, removed and modified during the negotiation process.

Contract versions are created by *derivation*. A derivation operation applied to a contract version called “the parent contract version” creates a contract child version which, just after derivation, is a logical copy of the parent. A contract version may have as many children as required. Once created, the new contract version will evolve autonomously. So will the parent contract version. Derivation relationships between contract versions are stored in the form of a tree represented by a *contract version table*. The

contract version table consists of triplets: *(parent_version, {children_versions}, negotiator)*.

Figure 2 illustrates the case when a contract exists in five versions: *ver0*, *ver0.1*, *ver0.1.1*, *ver0.1.2* and *ver0.2*. Contract versions *ver0.1* and *ver0.2* are derived from *ver0*. Contract version *ver0.1* has two child versions, *ver0.1.1* and *ver0.1.2*. Two negotiators *neg1* and *neg2* are involved in the negotiation process. Negotiator *neg1* is the author of three contract versions, *ver0*, *ver0.1*, and *ver0.1.1*. Negotiator *neg2* is the owner of two contract versions, *ver0.2* and *ver0.1.2*. The contract version table, on the right side, corresponds to the derivation tree on the left side.



parent	children	negotiator
ver0	ver0.1, ver0.2	neg1
ver0.1	ver0.1.1, ver0.1.2	neg1
ver0.1.1	∅	neg1
ver0.1.2	∅	neg2
ver0.2	∅	neg2

a) derivation tree b) contract version table

Figure 2. Storage of a derivation tree (a) in a contract version table(b).

A multiversion contract may be alternatively seen as a set of *multiversion contract parts* that are for example legal clauses, and values such as delivery dates or prices. A set of multiversion contract part is common for all the contract versions, i.e., all the contract versions are formally composed of the same parts. However, some part versions may be null that means “non-existence”. For example a null version of a contract part representing a legal clause in a particular contract version means that the negotiator do not included this clause in his/her proposal.

A multiversion contract part consists of a set of part versions, one part version associated with a contract version. A part version may be shared by several contract versions. For each multiversion contract part, an *association table* maintains a mapping between part versions and contract versions. More formally, a version part, denoted *svp*, is uniquely identified by *svID*. A multiversion part *mvp* consists of *(mvID, aTable, {svp})*. Each multiversion contract part is identified by *mvID*. The association table *aTable* is a set of couple *(svID, {contractVersions})*.

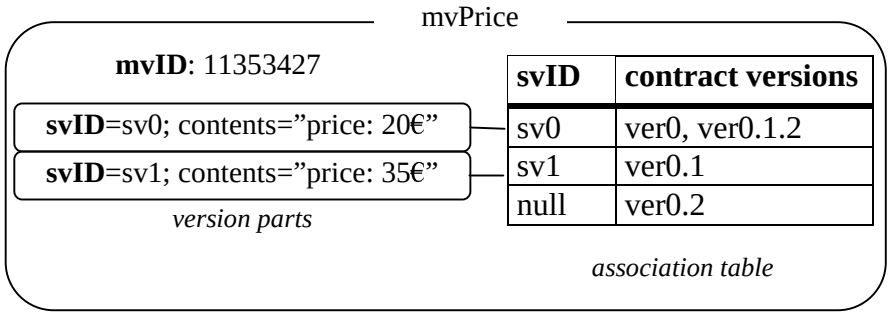


Figure 3. Example of a multiversion part representation.

Figure 3 illustrates the representation of a multiversion part, denoted *mvPrice*, of the multiversion contract introduced above. In contract versions *ver0* and *ver0.1.2*, the value of *mvPrice* is "price: 20€". In contract version *ver0.1*, the value of *mvPrice* is "price: 35€". In contract version *ver0.2*, the value of *mvPrice* is *null*, which means that the multiversion part is absent in this version. Null version part is defined for contract version *ver0.1.1*. The value of *mvPrice* for contract version *ver0.1.1* is thus inherited from the parent version, that is *ver0.1*. Therefore, the value of *mvPrice* for contract version *ver0.1.1* is "price: 35€". We can conclude that the first price (20€) proposed by negotiator *neg1* in contract version *ver0* is rejected by *neg2* in contract version *ver0.2*. Negotiator *neg1* proposes in contract version *ver0.1* a new price (35€) which is also rejected by *neg2*. However, in contract version *ver0.1.2*, negotiator *neg2* agrees with the first proposed price. Negotiator *neg1* maintains her/his offer in contract version *ver0.1.1*.

Association tables and the contract version table simplify modifications of the contract. If a negotiator derives a new version, only the contract version table is modified by the addition of the new version and the modification of the couple *(old_version, {children_versions})*. Association tables are not modified as the version parts are inherited from the parent version if they are not defined in the association table. The addition of a new contract part in contract version *cVer* causes the addition of a new multiversion part *newMvPart* in the contract. The value of *newMvPart* is *null* in the root version and for the potential version children of *cVer*. The value of *newMvPart* is set for *cVer*.

Association tables allow easy differencing of various contract versions. In the proposed contract model, the full tracability of the contract evolution in time and among various negotiators is achieved at the part level, due to association tables. A negotiator can take advantage of the tracability to detect difference between different contract versions. Two contract versions are identical if all their contract parts versions are identical, which means

that association tables of contract parts associate the same part version to the given contract version. As the tracability is achieved at the part level, no additional text parsing is needed during difference detection. In this way, system performance is improved.

4. MULTI-FACET ANALYSIS

A negotiation support system has to provide negotiators with a possibility of various analyses of contract versions authored by different negotiators to well understand different aspects of a conducted negotiation process. For instance, a negotiator may want to analyze the involvement of different negotiators in the negotiation process, or analyze the correlation between a contract part defining a delivery date and other contract parts.

To analyze the multi-thread history and the current status of a negotiation, both the abstract objects to be analyzed and the analysis criteria must be defined. An object is defined by a set of attributes. An attribute is a pair (*name, value*). An object is provided with an identifier unique in a given set of objects. For instance, a negotiator may be modeled as an object identified by negotiator's identity card number, and the associated attribute set may be {(*firstName, John*), (*lastName, Smith*), (*companyPosition, CEO*), (*involvement, 12)*}.

Formally, objects to be analyzed are generated by a mapping function *f* from space *C* to a set denoted D_f. Different mapping functions are used to analyze different aspects of the negotiation process. For example, to analyze negotiator's involvement, a function *f* may be defined to generate a set of objects comprising an attribute "involvement". The value of this attribute for a given negotiator is the number of contract versions generated by him/her.

In a highly concurrent environment, the result of the analysis should be a hierarchical classification. Given a set of objects, a classification splits it into subsets of similar objects, denoted *classes*. Hierarchical classification provides negotiators with a set of embedded classes. Negotiators can then choose a granularity level (the number of classes) of the classification. For instance, the same set of negotiators will be split into few classes if a general involvement characteristic is required, or into many classes if detailed characteristics of negotiator's involvement is required.

The criteria used to analyze set D_f are defined in a "human understandable" way, so that negotiators may choose and eventually define the criteria they want. The concept of similarity, which is the basis of classification, is closely related with the concept of distance. Two objects are similar if they are close to each other. The concept of distance is intuitive

and easily understandable. The proposed solution is based on metrics, and ultrametrics.

The mathematical definition of an ultra-metric is the following:

$$d \text{ is an ultra - metrics on space } D$$

$$\Leftrightarrow$$

$$\begin{cases} d : D^2 \rightarrow \Re^+ \\ \forall (x,y) \in D^2, d(x,y) = 0 \Leftrightarrow x = y \\ \forall (x,y) \in D^2, d(x,y) = d(y,x) \\ \forall (x,y,z) \in D^3, d(x,y) \leq \max(d(x,z), d(y,z)) \end{cases}$$

It has been proved that the concepts of ultrametric and hierarchical classification are equivalent [4]. In the proposed multi-facet analysis mechanism, the definition of a new classification criterion means the definition of a new ultrametric.

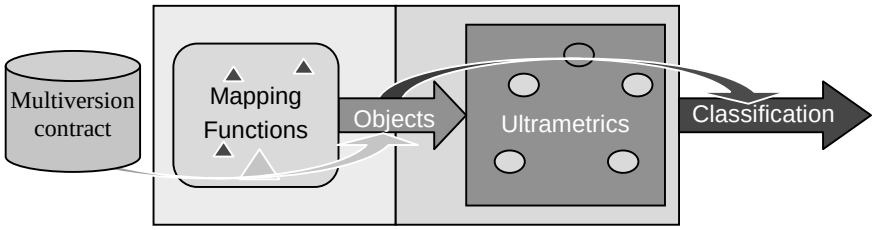


Figure 4. Multi-facet Analysis Process.

In Figure 4, a graphical representation of the multi-facet analysis process is given. A selected mapping function (left-side arrow in *Figure 4*) extracts data from the multiversion contract and generates a set of objects to be analyzed. Then a chosen ultrametric (right side arrow) processes the classification of the previously generated objects. As the result, a hierarchical object classification is obtained.

To illustrate the above technique, we present an example of a negotiation process analysis. In this example, we want to evaluate the weight of the various contract parts in the negotiation process. We assume that the more a given part was modified, the higher the interest of this part is. Function *f* is used to generate space D_D that consists of a set of objects denoted O_i. Each object O_i represents a multiversion contract part. Each element of D_D consists of all versions of a given contract part. The identifier of an object is the identifier of the multiversion contract part *mvID* introduced in Section 3.

The set of attributes of each object is restricted to only one attribute whose name is *numberOfVersion* and whose value is set to the number of versions of the corresponding multiversion part. In this example, we assume that card(D_D)=5 and O_1, O_2, … O_5 are the various objects representing the multiversion contract parts. The values of the ultra-metrics d on D_D^2 are given in Figure 5.

x	y	d(x,y)
O1	O2	7
O1	O3	5
O1	O4	5
O1	O5	7
O2	O3	7
O2	O4	7
O2	O5	3
O3	O4	1
O3	O5	7
O4	O5	7

Figure 5. Values of D on D_D^2

The hierarchical classification of D_D built on the basis of theses values is presented in Figure 6.

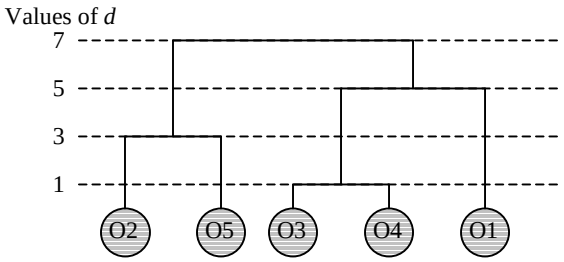


Figure 6. Classification of D_D according to d

This hierarchical classification can be seen at various detail levels. Having a given threshold T, the space D_D can be partitioned into different classes. In Figure 7 (respectively Figure 8), the partition obtained with threshold T=4 (respectively T=6) is presented.

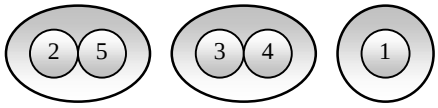


Figure 7. Partition of space D_D with threshold T=4

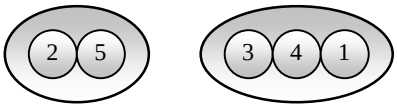


Figure 8. Partition of space D_D with threshold T=6

Let's assume that the multiversion contract part concerning the price, denoted *mvp1*, is identified by *1*. The multiversion contract part concerning legal clauses, denoted *mvp2*, is identified by *2*. The multiversion contract part concerning delivery date, denoted *mvp3*, is identified by *3*. The multiversion contract part concerning delivery address, denoted *mvp4*, is identified by 4. The multiversion contract part concerning number of items, denoted *mvp5*, is identified by 5. Let assume that the number of versions of *mvp1* is higher than the number of versions of *mvp3*, which is higher than the number of versions of *mvp2*. At a high level of analysis, when threshold *T=6*, *mvp1* and *mvp3* are in the same class, while *mvp2* is in another class. We can deduce that the price and the delivery date are of similar importance, while the importance of legal clauses is different. As the number of versions of *mvp1* is higher than the number of versions of *mvp2*, we can conclude that the class consisting of *mvp1*, *mvp3*, and *mvp4* contains parts of higher importance than the class consisting of *mvp2* and *mvp5*. The price, the delivery date, and the delivery address are thus of higher importance than legal clauses and the number of items.

At a low level of analysis, when threshold *T=4*, *mvp3* and *mvp4* are in the same class, while *mvp1* is in another class. As the number of versions of *mvp1* is higher than the number of versions of *mvp3*, we can conclude that the price is of very high importance, while the delivery date and address are only important. Having such a knowledge, negotiators can focus on the price negotiation which is the main issue in the current negotiation process. Such an analysis can be provided with various aspects of the negotiation process. Not only contract part importance may be analyzed but also for instance the involvement of negotiators or the importance of contract versions.

Changing the threshold, it is possible to control the granularity of space partition. In the context of highly concurrent environments, this ability enables negotiators to evaluate efficiently the status of the negotiation process. A high threshold gives a high level classification (with a few

classes) while a low threshold allows fine-grained classification (with many classes).

The choice of the threshold and the possible use of many ultra-metrics provide a very flexible framework for negotiation analyses. The capability to analyze every aspect of the negotiation process combined with the hierarchical classification allows to focus on a given problem and to view the results of the analyses at various detail levels.

5. CONCLUSIONS

The great amount of data resulting from the negotiation process in highly concurrent environments can be analyzed by the negotiation support system proposed in this paper. A prototype is currently under implementation, providing multiversion contract edition and multi-facet analysis. A graphical user interface for contract edition and version management has been implemented in Java. The back-end storage is an Oracle 8i database. Current work is focusing on mapping function definition in an XML dialect. Future works include visualization of the analysis results.

The proposed model provides negotiators with various synthetic views of the negotiation process helping them to identify problems and eventually to converge. Negotiation of complex contracts can be performed by a high number of negotiators communicating by Internet in an efficient way, opening doors to new business models.

REFERENCES

[1] Andreoli, J.M. and Castellani S. Towards a Flexible Middleware Negotiation Facility for Distributed Components. In *Proc. of 12th International Workshop on Database and Expert Systems Applications*, Munich, 2001, pp732-736.

[2] Cellary, W., Jomier G. Consistency of Versions in Object-Oriented Databases. In *Proc. of the 16th VLDB Conference*, Brisbane, Australia, August 1990.

[3] Cranefield, S. and Purvis M. UML as an Ontology Modeling Language. In *Proc. of the IJCAI-99 Workshop on Intelligent Information Integration*, Stockholm, 1999.

[4] Duda, R. and Hart P. *Pattern Classification and Scene Analysis*, New-York, Wiley, 1973.

[5] Fallside, D. C. XML Schema Part 0: Primer. W3C Recommendation, May 2 2001, http://www.w3.org/TR/xmlschema-0/

[6] Jennings, N. R., Faratin P., Lomuscio A. R., Parsons S., Sierra C., and Wooldridge M., Automated Negotiation: Prospects, Methods and Challenges. International Journal of Group Decision and Negotiation, 10(2), 2001.

[7] Lewicki, R. J., Saunders D. M., and Minton J. W. *Essentials of Negotiation*, McGraw-Hill, 2001.

[8] Rumbaugh, J., Jacobson I., and Booch G. *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1998.

[9] Ströbel, M. Communication Design for Electronic Negotiations on the Basis of XML Schema. In *Proc. of the 10th World Wide Web Conference*, Hong Kong, ACM Press New York, 2001, pp.9-20.