A CONTRACT BASED MODEL FOR CREATING STRUCTURED VIRTUAL DOCUMENTS: KEY ISSUES FOR REUSABILITY AND COLLABORATION

A. Boukottaya, C. Vanoirbeek , A.V. Nguyen Ngoc, Y.A Rekik , K. Zeramdini
Media Research Group
Swiss Federal Institute of Technology (EPFL)
In Ecublens, 1015 Lausanne, Switzerland
E-mail: {aida.boukottaya, christine.vanoirbeek,
anhvu.nguyenngoc, yassine.rekik, karim.zeramdini} @ epfl.ch

Abstract

The importance of structured document paradigm, especially with the intensive use of Internet and the adoption of XML technology has made a large amount of heterogeneously structured information widely available. In this framework, sharing and reusing structured pieces of data by several applications and users is of major concern. Although, significant results have been achieved in the domain of information reuse and computer supported collaborative work, several problems remain unsolved when applied to structured documents. The paper describes our so-called contract-based approach to combine and reuse structured document fragments to collaboratively build virtual documents. To get more insights into the problem, this approach will be validated in the educational domain. Although, it will be validated in a specific field, our approach wants to be general and applicable in the generation of structured virtual documents some is the applicability.
The paper has two goals (a) to raise problems related to reusing, sharing and assembling structured documents (b) and to present our framework: the FAVORITE project which objective is to define new models, methods and algorithms to create structured virtual course support from repositories of pedagogical material.

Keywords

Structured virtual Document, Reusability, Collaboration, Flexible document Model.

Introduction

The extensive use of documents in professional daily life activities makes them a very significant source of information. In this respect, the growing use of mark-up languages opens attractive perspectives: documents become dynamic and rich components of distributed information systems. They act as processable pieces of data that integrate a semantic dimension, supported by hyperlink facilities and use of metadata. In this framework, producing, sharing and reusing documents fragments between several applications within distributed environments is becoming a necessity. Building virtual documents, in a collaborative way, that takes benefit from existing structured document fragments brings up new challenges.

Two key issues are addressed in the paper. The first concern is about the authoring facilities to be provided to the users, allowing them to assemble structured pieces of information in such a way that both content and structure of document may potentially be reused. The second aspect is to take advantage of the document structure for anchoring

collaboration mechanisms that facilitate the cooperative authoring process.

We argue in favour of a contract-based approach that introduces flexibility when combining structured document fragments and, serve as the basis for the negotiation process between authors in order to reach an agreement on the content and the structure of final document.

This approach will be validated in the framework of a recent project : FAVORITE (**F**ragmenting and **A**uthoring **V**irtual **O**bject-based documents for **R**euse in **I**nteractive **T**eaching **E**nvironments) [Favorite] which objective is to define new models, methods and algorithms to create structured virtual course support from repositories of structured pedagogical material.

Currently, the research community has started to focus on pedagogical materials reuse and sharing. The idea is to provide collective shared databases for pedagogical materials. The goal is to reuse those materials by large number of authors and teachers when creating new interactive courses. ARIADNE described in [Cardinaels98] [Forte97] [ARIADNE] and SEMUSDI described in [Delestre98] [SEMUSDI] are two European research projects aiming to provide distributed shared databases for pedagogical material. The specificity of the FAVORITE project comparing to these environments is to give a major interest to the logical structure which greatly facilitates the reuse of content. We are interesting in the reuse of structured pedagogical materials to build a virtual course support conforming to a given model. Our approach is based on two major concepts. The first is what we call *Dynamic Structured Document Fragment*; an independent, structured and self-described piece of information, which can be reused and adapted to be integrated within new documents. The second is the concept of external element or what we call *External Fragment*. This concept is similar to the *External DTD Referencing* mechanism offered by XML. However, the definition of an external fragment will not be expressed by a fixed structure (DTD) but using a set of constraints which introduce a lot of flexibility in document modeling.

The paper is organized in the following way. The first section briefly summarizes the limits of current approaches that support collaborative authoring of structured documents. Section 2 points the way to the need for new flexible and modular document model that addresses both reusability and collaboration. Section 3 describes the main aspects of our proposed flexible model language based on the concept of contracts. Section 4 deals with constraint classification. Section 5 shows how contract based model supports collaboration. Finally, we conclude the paper with a discussion and future work .

1. Critical issues

1.1 Reusing structured documents

It is commonly accepted that structuring documents greatly facilitates their exchange and reuse [Quint94][Quint95]. However, despite the obvious advantages conveyed by structuring document, reusing them within different users' environments raises a number of problems to transform or to adapt them according to user needs and context. A number of research works are dedicated to structure or re-structure them in such a way that facilitates their reuse [Bonhomme96]. Depending on the context, several approaches have been

proposed to address these problems. Transforming existing structured documents in order

to fit a different target structure may be either performed explicitly through descriptive rules to guide the transformation process such as XSL standard or automatically [Akapotsui97]. A combined approach has also been proposed to take benefit from the two mentioned approaches [Bonhomme97][Bonhomme98]. The issue of structures transformations remains complex. Explicit transformation makes the task of customisation in the document edition process difficult because it requires specific languages knowledge. Automatic transformation causes in most cases the degradation in term of quality of the reused fragment [Rekik01]. Our approach aims at minimizing the effort of structure transformations by providing the user with a simple method to specify document content in a declarative way that fit the requirements of the authoring tasks.

1.2 Collaborative process for the document production

The rapid development of computer networking in the last decade has provided new possibilities for developing collaborative systems, in which users collaborate to perform common tasks. For document production, many group editors such as Sepia [Haake and Wilson, 92], Duplex [Pascull et. al., 94], or Reduce [Yang et. al., 00] were developed. In those systems, the collaboration is supported by providing CSCW methods and tools [Borghoff and Schilichter, 00]. However, those group editors are all not centered on the concept of structured document, that means the composition/decomposition of document from/to different fragments is not supported. In other words, it prevents the flexibility and reusability of documents. Other editors like Alliance [Decouchant et. al, 96] and Grif [Quilt and Vatton, 86] are structure-driven editors. Nevertheless, users cannot modify the document structure based on their needs. The document-based collaboration is somehow supported in document workflow systems. But these systems are mostly called "single document workflow", i.e. the basic input document is a document to which annotations or comments may be attached. We propose a document-centered approach that allows us to anchor the collaboration mechanisms on the document components themselves.

2 How to provide Reuse and Collaboration when modelling document?

The formal description of a document class aims at constraining the logical organisation of document instances and, thus guarantees their adequacy to a given model. The need to build such descriptions in a modular way, taking benefit from existing document fragment description has been rapidly identified and addressed in different ways. However XML Schema introduced mechanisms allowing the reuse of existing structures and documents, the document instance is generally considered and manipulated as an atomic information block. To facilitate document reuse, ongoing research works have mostly adopted an object-oriented design for the appropriate representation of documents models [SOX99][Abiteboul99]. This approach is relevant in many respects. It allows first the precise definition of pieces of information and aims at facilitating the exchange and reuse of document fragments between heterogeneous applications. In this framework, we propose a new approach for document modelling. This model is based on two main concepts which are *modularity* and *flexibility*.

2.1 Modularity

At modelling level, we propose the concept of *module* that defines the decomposition of the final document. The document structure is described through a set of modules. To each module is attached a description of its structure and content. Figure 1 illustrates a modular

vision of a document "course" composed of two modules "Definition" and "Exercise". The instance of a modular structure is called *Composite Document*. A composite document is seen as a collection of independent parts called *fragments* representing document objects having their own structure definition (module). Defining a modular structure clearly improves reusability and collaboration. It facilitates the exchange of document fragments and can be regarded as a definition of a contract constraining the decomposition of the future document between several authors.

2.2 Flexibility

The second concept introduced by our model is the flexibility in document modelling. The idea is that in many times, a designer could not have a precise idea about some modules definition. For example, when defining a document class structure for a course support, it is clear to involve modules of type exercise. However, it is possible to have no precise idea about the micro-structure of these exercises. Our goal is to offer the possibility to define "*constrained module*". Such a module is called constrained, because its structure is not described by a sequence of elements and attributes. Rather than this, constrained modules are described just by a set of constraints called *contract*.

Using such flexible structures permits the reuse of existing document fragments without dealing with the classical problem of instance-structure conformity. To illustrate this idea, let's take an example. Suppose that a professor is editing a new course, and wants to reuse existing exercises available in same given repository of well-formed XML fragments. With classical modelling, he needs to transform the exercise to the target structure. Transforming existing structured documents in order to fit a different target structure may be performed either explicitly or automatically. In both cases, structure transformation is an additional work accompanied by information loose. With our approach, the professor needs just to specify a contract (set of constraints): for example the topic of the exercise, which is used as selection criteria to determine which fragment to use (Figure 2). Contracts provide a unified way to specify at the same time the document structure, its content, and to control the authoring process itself especially when multiple authors are involved.

In addition to constrained modules, we can describe a module by its exact structure (sequence of elements and attributes) just like existing document models. These modules are called *Fixed modules*.

Addressing constrained modules, we are focusing our work first on a collection of constraints dealing with the internal logical structure, the content of a module and the dependency between modules. Second, we are concentrating in expressing these constraints in a simple and declarative way.

3 The Flexible Document Model language (FDM)

The FDM language we propose defines the syntax of a family of conforming XML documents. A composite document intended to conform to a given FDM. A FDM

specification is itself an XML document. In this section, we will not give the FDM formal syntax description but just describe the main aspects of the FDM language and its meaning. The FDM language is not yet finalized. For the moment only modules, global attributes, and constraints declarations are provided.

```
< DocumentClass name=”Course” >
<Gattribute name=”topic” type=”string” use=”required”>
  < FixedModule name= “Definition”>
    ---content specification---
  </FixedModule>
....
  <ConstrainedModule name =”Exercise”>
      ---contract specification---
  </ConstrainedModule>
....
</DocumentClass>
```

Figure 1: A modular vision of a document “course”

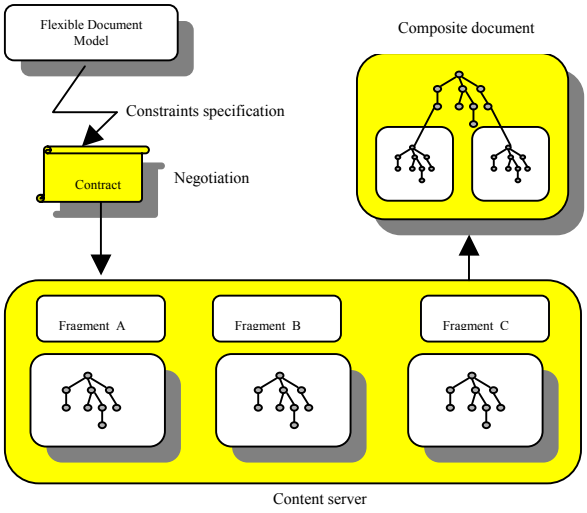

Figure 2: Architecture for contract based document fragment assembly

3.1 Document Class definition

Like in any document structure description language, a root element is declared using the tag <DocumentClass> in which an attribute “name” is associated to specify the document class’s name. Each document class is described by a set of global attributes and may contain a sequence of fixed and constrained modules.

Documentclass_dec → < DocumentClass name= namevalue >
(GlobalAttribute_dec)*
(Module_dec)+
</ DocumentClass>

Module_dec → ((FixedModule_dec) | (ConstrainedModule_dec))

3.2 Global attribute declaration

Global attributes are related to root elements (DocumentClass), fixed modules and constrained modules. The appellation “global attributes” makes these attributes different from attributes related to elements within fixed modules. A global attribute declaration consists of a name and a type together with occurrence information and (optionally) a default value.

GlobalAttribute_dec →<Globalattribute name= namevalue type= atttype use=(optional | required | fixed | default) value= attvalue/>

3.3 Fixed Modules declaration:

A Fixed Module is described with a set of global attributes and a content description.

FixedModule_dec → < FixedModule name= namevalue >
(GlobalAttribute_dec)*
(Content_dec)
</ FixedModule>

3.4 Constrained Modules declaration

A constrained Module is described with a set of global attributes and a contract witch is a set of constraints.

ConstrainedModule_dec → < ConstrainedModule name= namevalue >
(GlobalAttribute_dec)*
(Contract_dec)
</ ConstrainedModule>

3.5 Contract declaration

A contract is a set of constraints describing document structure and content. A constraint is a statement of a relation (in the mathematical sense) that we would like to have hold. Our notion of constraint combines several ideas: A constraint should captures first document structure: the legality of attributes, attributes values and element. Second express dependencies between attributes, attributes values, element contexts, and content. A preliminary classification of constraints has been completed. In our model, we classify the constraints into two categories: *Simple constraints* and *Dependency constraints*. A valid fragment that can be inserted into a composite document has to fulfill these constraints.

Contract_dec → <contract>
((SConstraint_dec)| (DepConstraint_dec))+
</contract>

4 Contract Specification

The concept of contract will be illustrated through our "Course Creation" example: a group of professors provide a repository of well-formed XML fragments such as *exercises*,

theorems, *exercise solutions*, *theorem proofs*, *definitions*, *citations, images, videos*…etc. The goal of this repository is to assist the professors to create new structured courses taking benefits of existing fragments rather then creating them from scratch. Based on our flexible document model, professors define the course structure through constrained and fixed modules according to their needs. The defined structure traduces professors' know-

how, which describes possible course content and organization. A structured virtual course is an instance of this structure. From this point of view, many virtual course supports can be generated conforming to structures imposed by professors and this structure it self can be reused by several professors to create their own courses. In the second case, the professors' know-how is shared and reused.

4.1 Simple constraints

Contrary to dependency constraints that express dependencies between two or more variables, simple constraints concern one variable. A variable is a module, an element, a global attribute or an attribute.

Simple constraints deal first with *modules logical structure*: elements data types, attributes values, order and occurrence of elements, element inclusion, element exclusion…etc. Figure 3 illustrates, in an informal syntax, the definition of a flexible document class exam, in which the professor wants to insert in his document exam a module exercise that does not contain an element solution.

Simple constraints may have also a semantic dimension. For example, a professor may add that the document exam should contain a *difficult* exercise having for *topic "java classes"*. Figure 4 illustrates this example. As we can see through these examples, the professor is given a lot of freedom to control which fragments can be inserted into his document exam. The constraints may either be very rigid or very flexible according to professors needs.

4.2 Dependency constraints

Dependency constraints are important to provide adaptability. Existing document definition languages lack appropriate methods to express dependencies between elements and attributes. We focus our work on what we call "*context dependencies*": Often modules and global attributes are allowed only in certain syntactic context. A context can be either a sequence of modules or global attributes or combination of modules and global attributes. Three context dependency examples are given, the first describes dependencies between two modules, the second between global attributes and modules. The third example deals with integrity constraints.

Example 1: Module dependencies:

Express dependencies between two modules. For example, a module Exercise is allowed only if it is preceded by a module example having the same topic (Figure 5).

Example 2: Modules dependent on global attribute values:

The content of a module or sequence of modules, sometimes depends on global attributes or their values (Figure 6): the composition of the course document vary depending on the value of the global attribute "topic".

Example 3: Integrity constraints:

Like in databases, integrity constraints play an important role to convey an essential part of the semantic of the data. For example, if a global attribute called "duration" is attached to each module, we should ensure that the sum of modules durations is equal to the document class duration. A lot of research activities focus on these constraints, a formal syntax have been proposed to express integrity constraints [Mckernan00].

```
<DocumentClass Exam>
 <ConstrainedModule name =”Exercice”>
   <contract>
     <SConstraint>
        <Element name = “solution” type=”Excluded”/>
     </SConstraint>
  </contract>
 </ConstrainedModule>
 <FixedModule Exercice>
   -----content description-----
 </FixedModule>
</DocumentClass>
```

Figure 3: An example of simple constraint dealing with modules logical structure.

```
<DocumentClass Exam>
  <ConstrainedModule name =”Exercice”>
    <contract>
      <SConstraint>
          <Globalattribute  name=”Topic” value=”java classes” type=”Required”/>
          < Globalattribute  name=”Difficulty” value=”difficult” type=”Required”/>
          <Element name = “solution” type=”Excluded”/>
      </SConstraint>
    </contract>
  </ConstrainedModule>
  <FixedModule Exercice>
        -----content description-----
  </FixedModule>
</DocumentClass>
```

Figure 4: An example of simple constraint dealing with modules semantic.

```
<DocumentClass Course>
 <FixedModule  name = “Example”   use=”optional”>
   <Globalattribute name=”topic”  type=”string”  use=”required”/>
          -----conentt specification------
 </FixedModule>
 <ConstrainedModule name = “Exercise”   use=”optional”>
  <Globalattribute name=”topic”  type=”string”  use=”required”/>
     <contract>
    <Depconstraint>
        <FixedModule  name = “Example”
        Topic = “Exercise. Topic”
        t ype=”Required-in-front”/>
    </Depconstraint>
  </contract>
  </ConstrainedModule>
</DocumentClass>
```

Figure 5: Dependency between two modules

```
<DocumentClass Course  >
<Globalattribute name=”topic”  type=”string”  use=”required”/>
 <contract>
    <Depconstraint>
          <case  Course.topic =” Computer science”>
            <FixedModule >
              -----content description-----
            </FixedModule>
            <ConstraintModule Exercice >
              -----contract description-----
            </ConstraintModule>
          </case>
          <case  Course.topic =” Medecine”>
            <FixedModule QCM>
              -----content description-----
            </FixedModule>
          </case>
    </Depconstraint>
  </contract>
</DocumentClass>
```

Figure 6: Modules dependent on global attribute values

5 How contract based model supports collaboration?

Our proposed document model allows collaborations in two different levels. Users can collaborate to achieve a final document either in the document definition phase or in the document usage phase. In the first case, the collaboration occurs when users define the fragments, retrieve a fragment from the repository or compose some fragments to produce

a composite document. This mostly concerns with the document *structure* activities. In the upper level, users collaborate by contributing to the *content* of the document. The discussions and examples below could be applied to both of these two levels of collaboration.

In [Schoop and Quix, 00], the authors stated that negotiations are needed towards reaching an agreement that is manifested in a contract. Normally, the *negotiation process* is supported by a message exchange and/or notification mechanism. For instance, a professor can negotiate with his teaching assistants to define the constraints for a "course" fragment. The assistants will all be notified after the fragment is completely defined.

According to the contract, a fragment may be routed around many users. The fragment paths help identifying the collaboration routes among users. Furthermore, because the *sub-contracting* is allowed, the collaboration routes can be modified on the fly. The routes help showing the workflow and dataflow between users when performing together a common task.

Constraints can be processed in different ways. A so-called *validation engine* at each user side can serve as a filter to accept or reject the arriving fragment. The validation process is based on the constraints included in the arriving fragment contract and the *user's contract profile*, which could be defined depending on the user's existing fragment contracts and other data inputs representing the *user's needs*. When receiving a fragment, the validation engine extracts the constraints from the fragment contract, and then checks whether these constraints have conflicts with the user's contract profile. If this is the case, according to the *local policy*, the input fragment can be rejected or forwarded to another user or a renegotiation process is invoked. If the received fragment is accepted, the user can edit that fragment. The edition of course cannot violate the included constraints. The constraints can be added, removed, or modified if the users can reach a consensus. By using the contract, users can obtain the appropriate document fragments based on their preferences.

In some situations, the constraints (or at least some of them) are not processed at all. In this case, they serve as a means to provide awareness information, which is defined by [Dourish and Bellotti, 92] as an understanding of the activities of others, thus helps providing the context for each member's activities. The awareness information is really useful for an effective collaboration between users.

6 Discussion and future work

This paper has addressed the problem of reusability of structured documents in distributed environments. Its objective is to define new models, methods and algorithms that consider a document as the composition of several pieces that might be shared and reused according to user needs. A new document model to build structured virtual documents in a collaborative way taking benefit from existing document has been

proposed. To answer reusability and collaboration requirements, this model is based on two main concepts: *modularity* and *flexibility*.

Our vision of a modular structured document model is based on an object-oriented approach in which the document is seen as a collection of what we call "fragments" which are consistent and relatively independent document parts used as building blocks for the final document to be composed.

As we have seen, defining a module just by fixing a contract offers a high level of flexibility to reuse existing documents in the instantiation phase. The idea is to provide the user with a simple declarative language to constraint his document. Contracts are used first to ensure the global document's coherence and second as a set of criteria for searching interesting fragments or components to be included in the resulting document.

Currently, the flexible document model (FDM) language is not yet finalized, we intend to first devote future research on the formalisation of this language.

Second we will deal with the design of a "*Fragment Server*". This task must take into account three major points:

- How to manage and store dynamic structured fragments? How to wrap an existing document (plain text, text with markup, image, sound, etc.) with meta-data and operations in order to create a document fragment?

- What are the access functions and mechanisms a fragment server has to offer? Two promising approaches may be adopted for this task. The querying based approach represent the first direction and is based on recently proposed document querying languages. The second direction is based on navigational models. This will be based on the recent work done in the domain of hypertext navigation and information retrieval.

- How to exchange dynamic fragments between the Fragment Server and authoring tools or document processing applications?

In addition to FDM language formalisation and fragment server design, our interest is to design a *Collaboration Edition Framework*, which will ensure the reusability of fragments taken from a content server and facilitate the collaboration of many authors in the edition of a structured document. The framework should therefore allow generating on the fly a *Personalized Editor* that respects the document model. Each module of the document model is therefore seen as a GUI component of the Editor. The semantic and structural coherence of the GUI content will be guarantied by the constraints attached to the module. Each GUI will provide the methods allowing negotiating with the content server the conformity of the fragments to be inserted with the document model.

References

[Abiteboul, 99] S. Abiteboul, R. Goldman, J. McHugh, V. Vassalos, Y. Zhuge (1999). "Views for Semi-structured Data", Tech. Report, Dept of Computer Science, Stanford University.

[Akapotsui, 97] E. Akpotsui, V. Quint, C. Roisin (1997). Type Modelling for Document Transformation in Structured Editing Systems. *Mathematical and Computer Modelling*, vol. 25, num. 4, pp. 1-19.

[ARIADNE] Web site of ARIADNE project: http://ariadne.unil.ch/

[Bonhomme, 96] S. Bonhomme, C. Roisin (1996). Interactively Restructuring HTML Documents. *Computer Network and ISDN Systems*, vol. 28, num. 7-11, pp. 1075-1084.

[Bonhomme, 97] S. Bonhomme, C. Roisin (1997). Transformations de documents électroniques. *Document Numérique*, vol. 1, num. 3.

[Bonhomme, 98] S. Bonhomme (1999). *Transformations de documents structurés : une combinaison des approches déclaratives et automatiques.* Doctorat d'informatique, Université Joseph Fourier.

[Borghoff and Schilichter, 00] Uwe M. Borghoff, Johann H. Schichter (2000). *Computer Supported Cooperative Work.* Springer.

[Cardinaels98] Cardinaels K, Hendrikx K., Vervaet E., Duval E., Olivi H., Haenni F., Warkentyne K., Wentland-Forte M. and Forte E., A knowledge Pool System of Reusable Pedagogical Elements. In proceeding of CALISCE'98, 13-17 June 1998, Goeteborg, Sweden.

[Decouchant et. al., 96] D. Decouchant, V. Quint, M.R. Salcedo (1996). Structured and distributed editing in a large-scale network. Groupware and Authoring", *Academic Press*, London.

[Delestre98] Delestre N. and Rumpler B., Architecture d'un Serveur Multimdia pour les Sciences de l'Ingnieur. In proceeding of NTICF'98, 18-20 November 1998, INSA de Rouen, France.

[Dourish and Bellotti, 92] Paul Dourish and Victoria Belloti (1992). Awareness and Coordination in shared workspaces. ACM 1992.

[Favorite] http://lithpc8.epfl.ch

[Forte97] Forte E., Wentland-Forte M. and Duval E., The ARIADNE Project, (part II) - Knowledge Pools for Computer Based Telematics Supported Classical, Open Distance Education. In European Journal of Engineering Education, Vol. 22, No 2/1997 (June), pp. 153-166.

[Haake and Wilson, 92] Jorg M. Haake, Brian Wilson (1992). Supporting collaborative writing of Hyperdocuments in SEPIA, *CSCW92*.

[Mckernan00] Timothy D. McKernan and Bharat Jayaraman (2000). CobWeb: A Constraint-based XML for the Web. *Technical Report 2000-06, Department of Computer Science and Engineering, State University of New York at Buffalo, May 2000.*

[Pascull et. al., 94] F. Pascull, A. Sandoz, A. Schiper (1994). DUPLEX: a distributed collaborative editing environment in large scale, *ACM CSCW94*.

[Quint and Vatton, 86] V. Quint, I. Vatton (1986). Grif: an Interactive system for structured document manipulation. *Cambridge University Press.*

[Quint, 94] V. Quint, I. Vatton (1994). Making structured documents active. *Electronic Publishing* - Organisation, Dissimination and Design, 7(2): 55-74.

[Quint, 95] V. Quint, I. Vatton, J. Paoli (1995). Active Structured Documents as User Interfaces. *vol. User Interfaces for Symbolic Computations. Springer Verlag*.

[Rekik, 01] Y. A. Rekik (2001). Modélisation et manipulation des documents structures. *PHD Thesis number 2396. Swiss Federal Institute of Technology (EPFL)*

[Schoop and Quix, 00] M. Schoop, C. Quix (2000). Towards effective negotiation support in electronic marketplaces. *WITS 2000.*

[SEMUSDI] Web site of SEMUSDI project: http://semusdi.insa-rouen.fr/

[SOX, 99] W3C, W3C note, Schema for Object-Oriented XML 2.0. 30 Jul 1999. http://www.w3.org/TR/NOTE-SOX/

[Yang et. al., 00] Y. Yang, C. Sun, Y. Zhang, X. Jia (2000). Real-time Cooperative Editing on the Internet. *IEEE Computing 2000*.