Consistency Management of Financial XML Documents

Andrea Zisman[1] and Adamantia Athanasopoulou[2]

[1] City University, Department of Computing, Northampton Square,
London EC1V 0HB, UK
a.zisman@soi.city.ac.uk

[2] Singular International SA, R&D Department, 31 Polytechneiou St.,
Thessaloniki 54626, Greece
aath@si.gr

Abstract. In the financial domain a large number of inconsistent documents are produced every day. Up to now, many of the consistency management activities are executed manually, generating significant expense and operational risks. In this paper we present an approach for consistency management of financial XML documents. The approach includes the activities of consistency checking and consistency handling. It is based on consistency rules, used to express relationships among elements and documents, and resolution actions, used to restore the documents to a consistent manner. A prototype tool has been developed to demonstrate and evaluate the approach.

1 Introduction

A large number of financial documents are produced every day, either as a result of financial transactions involving multiple actors with different views and opinions, or as a result of accessing and manipulating data in trading systems. These systems are generally created and administered independently, differing physically and logically. The heterogeneity of these systems are exhibited in the use of different programming languages, the availability of different platforms and operating systems, and the various ways of storing, manipulating and exchanging data. Inevitably, the produced documents are often inconsistent.

It is important to manage these inconsistencies to allow interoperability and integration of front-, middle- and back-offices, and to support tasks like trade confirmation, trade settlement, and trade collateral matching. Up to now, many of the consistency management activities are executed manually, generating significant expense and operational risks.

For example, consider a trade confirmation process, where each party produces a document with its own view of a trade that has been agreed to over the phone. In a normal scenario, the parties exchange these documents via fax or electronically in order to check inconsistencies of the documents' content, such as settlement date, rate, amount, name of the parties. The parties produce consistent versions by checking

© Springer-Verlag Berlin Heidelberg 2001

documents manually and discussing changes via phone and fax before executing the settlement.

With the development of the Internet and eXtensible Markup Language (XML) [4], new standard data interchange mechanisms for the financial domain have been proposed. Examples of these standards are the Financial product Markup Language (FpML) [15], Financial Information Exchange Protocol Markup Language (FIXML) [16], Network Trade Model (NTM) [19], and Open Trading Protocol (OTP) [22]. These standards have been extensively used to support many financial activities, ranging from Internet-based electronic dealing and confirmation of trades, to interchange data between front-, middle-, and back-office services. The result is the creation of various documents representing instances of the standards, generated by different applications and person everyday. However, consistency management of these documents is still an open question.

In this paper we propose an approach for consistency management of financial XML documents. Our approach is simple, lightweight, and concentrates on inconsistencies in the documents' instance level. It is based on *consistency rules*, used to express relationships among elements, and *resolution actions*, used to restore the documents to a consistent state. When dealing with financial documents the resolution of inconsistencies is necessary and important to avoid mismanagement of data.

The proposed approach tackles the activities of *consistency checking* and *consistency handling*, in the consistency management process [31]. Consistency checking is concerned with the tasks of specifying *consistency rules* and identifying inconsistent elements, by checking for violations of the consistency rules. Consistency handling is related to the tasks of specifying *resolution actions* for dealing with inconsistencies, selecting the resolution actions to be executed, and restoring the documents to a consistent format by applying the resolution actions[1].

The work presented in this paper extends previous work for consistency management proposed in [10][34]. In the previous work the authors proposed an approach to allow identification and detection of inconsistencies in distributed XML documents, based on consistency rules, where the related elements are associated through hyperlinks named *consistency links*. The approach presented in this paper complements this former work by describing a way of handling inconsistencies, and applying the whole approach in a specific domain, i.e. finance.

The rest of this paper is organized as follows. Section 2 describes the approach being proposed. Section 3 presents a formalism to express the consistency rules, a classification for the different types of rules, and examples of these various types. Section 4 describes a formalism to express the resolution actions and one example of these actions. Section 5 addresses the implementation of XRule tool to support the approach. Section 6 discusses related work. Finally, section 7 summarizes the approach and suggests directions for future work.

[1] As outlined in [31], in a consistency management process the consistency handling actions depend on the type of inconsistencies and can be of various types. Examples are actions that modify documents, restore or ameliorate inconsistencies, notify users about inconsistencies, and perform analysis. In this paper we concentrate our work on actions that restore the documents, called hereafter as *resolution actions.*

2 The Approach

Our approach is based on XML and related technologies such as XPath [7] and XSLT [6]. Fig. 1 presents an overview of the approach. The main component of our approach is called *XRule* and is composed of a *consistency checker* and a *consistency handler*.

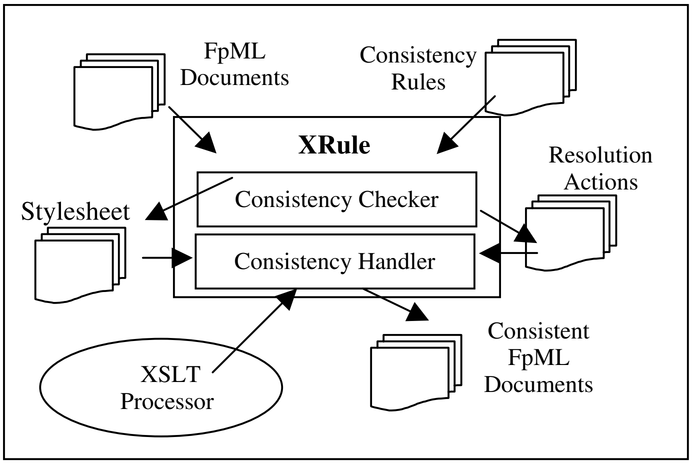


Fig. 1. An overview of the approach

The consistency checker is responsible for identifying inconsistencies in the participating XML documents. It receives as input financial XML documents (e.g. FpML instance documents) and consistency rules previously defined. The consistency rules describe the relationship that should hold between the participating documents. In the next step, the consistency checker verifies for violation of the consistency rules in the XML documents, i.e. the relationships that do not hold. In cases where inconsistencies are detected, the consistency checker generates resolution actions related to these inconsistencies and a XSLT stylesheet document.

The resolution actions specify the parts inside the XML documents that are inconsistent and to which value they should be changed in order to eliminate inconsistencies. The XSLT stylesheet describes how to transform one XML document into another XML document. The stylesheet and resolution actions are used by the consistency handler to restore the XML documents into a consistent state. The restoration of the documents is supported by XSLT processor, which transforms inconsistent XML documents into 'new' consistent XML documents[2].

The resolution actions and stylesheets are dynamically created during the consistency management process. This is due to the fact that it is not possible to know which parts of the participating documents are inconsistent before executing the consistency checks. In addition, depending on the type of inconsistency and on the inconsistent element, the user needs to interact with the system to specify a 'new value' to which the inconsistent element should be modified.

[2] A detailed description of XSLT [6] is beyond the scope of this paper.

3 Consistency Rules

In this section we present the syntax used to express the consistency rules and the different types of consistency rules that we can express using the syntax. The consistency rule syntax is similar to the formalism proposed in [34]. However, it involves logical quantifiers (forall, exists) and is described in terms of XML [4] and XPath [7] syntax. The reasons for using XML and XPath are (a) to provide an open and standard way of expressing the rules; (b) to facilitate and standardise the construction and execution of a consistency rule interpreter; (c) to aid generation of resolution action; and (d) to facilitate access to and modification of XML documents.

A consistency rule is composed of two parts. Part 1 is related to relevant sets of *elements* in various documents to which the rule has to be verified. Part 2 is concerned with *conditions* expressing the relationships between the elements in part 1 that have to be tested.

Fig. 2 illustrates the Document Type Definition [4] for the consistency rule syntax. It contains a root element called ConsistencyRule composed of six element contents and one attribute *id.* Attribute *id* uniquely identifies the consistency rule. The element contents are described below.

- Description – it contains a natural language description of the consistency rule;
- Source & Destination – they contain XPath expressions for identifying sets of elements to be checked against the consistency rule. It is possible to have more than one type of Destination elements to be checked against the same type of Source elements. This occurs when a consistency rule refers to more than two types of element sets in the participating documents. Thus, for each type of Destination element set in a rule there is a unique identification represented by attribute *dest_id*, which is referenced as an attribute in element Condition;
- Condition – it is composed of six attributes:

• *expsourcequan*t - a quantifier that can have value “forall” or “exist”, which is used to specify if the condition has to be satisfied for all elements, or at least one element, respectively, in the Source element set;

• *expsource* - an expression related to the Source element set;

• *op* - an operator associating expsource with expdest, which can have the following values: *equal, not_equal, greater_than, less_than, less_equal, greater_equal,* and *sum*;

• *dest_ref* - a reference to the unique identification of a Destination set;

• *expdestquant* - a quantifier that can have value “forall” or “exist”, which is used to specify if the condition has to be satisfied for all elements, or at least one element, respectively, in the Destination element set;

• *expdest* - an expression related to the Destination element set.

- Operator – this element is related to the situation in which the rule is composed of more than one condition. It contains an attribute value, which can have the Boolean content “AND” or “OR”.

The proposed consistency rule syntax allows the representation of different types of consistency rules. We classify these types based on the facts that (a) the consistency management process is executed by comparing the participating documents and the elements composing the Source and Destination sets pair wise;

and (b) in XML documents data can be represented either as *elements* or *attributes*[3]. Therefore, the different types for consistency rules that can be represented by using the syntax shown in Fig. 2 are related to the comparison of documents, elements, and mixture of documents and elements.

```
<!ELEMENT ConsistencyRule (Description, Source,
    Destination+, Condition, (Operator, Condition)*)>
<!ATTLIST ConsistencyRule   id    ID #REQUIRED>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Source (XPath)>
<!ELEMENT Destination (XPath)>
<!ATTLIST Destination dest_id   ID #REQUIRED>
<!ELEMENT Xpath (#PCDATA)>
<!ELEMENT Condition EMPTY>
<!ATTLIST Condition
         expsourcequant  CDATA  #REQUIRED
         expsource       CDATA  #REQUIRED
         op              CDATA  #REQUIRED
         dest_ref        CDATA  #REQUIRED
         expdestquant    CDATA  #REQUIRED
         expdest         CDATA  #REQUIRED>
<!ELEMENT Operator  EMPTY>
<!ATTLIST Operator  value (AND|OR)  "AND" >
```

Fig. 2. Consistency rule syntax

Table 1 summarizes a general classification for the consistency rules. In the table the names in the rows and columns are related to the different types of components being compared: the Source element set and Destination element set, respectively. The Source and Destination sets reference participating documents and XML elements. The content of each position in the table refers to different consistency rules described below.

This general classification can be refined to a more specific classification where we consider the cardinality of the Source and Destination sets. Table 2 presents a specialized classification for the consistency rules, based on the fact that either all elements or at least one element, and either all documents or at least one document in the Source and Destination sets are compared.

In order to illustrate, we present examples of some of the different types of consistency rules related to finance. The rules are specified in XML, based on the DTD syntax of Fig. 2. For the examples we assume documents related to FpML standard [15].

[3] XML has no rules related to when data should be represented as element or attribute. For instance, in the XML Metadata Interchange (XMI) standard [21] all components of a UML model are represented as elements. In this text we use the term element meaning both XML elements and attributes.

Table 1. General classification of consistency rules

Destination Set / Source Set	Element	Document
Element	Type 1	Type 3
Document	Type 2	Type 4

Table 2. Specialised classification of consistency rules

Destination Set / Source Set	∀ Element	∃ Element	∀ Document	∃ Document
∀ Element	Type 1.1	Type 1.2	Type 3.1	Type 3.2
∃ Element	Type 1.3	Type 1.4	Type 3.3	Type 3.4
∀ Document	Type 2.1	Type 2.2	Type 4.1	Type 4.2
∃ Document	Type 2.3	Type 2.4	Type 4.3	Type 4.4

Type 1: Existence of related elements

This type of rule is related to the existence of related elements in different documents or in the same document.

Example: (Type 1.2) - For every two Foreign Exchange (FX) swap trade documents F_1 and F_2, the party references in F_1 has to be the same as the party references in F_2.

```
<ConsistencyRule id=”R1.2”>
<Description> For every two FX swap trade documents representing a trade, the party
references in each of the documents have to be the same </Description>
<Source> <XPath> /fpml:FpML/fpml:Trade/fpml:tradeIDs/fpml:TradeIDs/tid:TradeID/
    tid:partyReference  </XPath> </Source>
<Destination dest_id=”pR”> <XPath>  /fpml:FpML/fpml:Trade/fpml:tradeIDs/
   fpml:TradeIDs/tid:TradeID/tid:partyReference </XPath> </Destination>
<Condition  expsourcequant=”forall”
                expsource=”.”
                op=”equal”
                dest_ref=”pR”
                expdestquant=”exists”
                expdest=”.”  />    </ConsistencyRule>
```

Type 2: Existence of elements due to the existence of documents

This type of rule is related to the situation in which the existence of one or more documents requires the existence of elements in another document.

Example: (Type 2.4) - For every Foreign Exchange (FX) swap trade documents F_1 and F_2, related to a trade involving two parties, the names of the parties must exist in the documents.

```
<ConsistencyRule id="R2.4">
<Description> For every FX swap trade documents, related to a trade involving two parties, the
names of the parties must exist in the documents.
</Description>
<Source> <XPath> /fpml:FpML/fpml:Trade </XPath> </Source>
<Destination dest_id="pR"> <XPath> /fpml:FpML/fpml:Trade </XPath> </Destination>
<Condition  expsourcequant="exists"
        expsource="./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[1]/tid:partyReference"
        op="equal"
        dest_ref="pR"
        expdestquant="exists"
        expdest="./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[1]/tid:partyReference"/>
<Operator value="OR"/>
<Condition  expsourcequant="exists"
        expsource="./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[1]/tid:partyReference"
        op="equal"
        dest_ref="pR"
        expdestquant="exists"
        expdest="./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[2]/tid:partyReference"/>
<Operator value="AND"/>
<Condition  expsourcequant="exists"
        expsource="./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[2]/tid:partyReference"
        op="equal"
        dest_ref="pR"
        expdestquant="exists"
        expdest="./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[1]/tid:partyReference"/>
<Operator value="OR"/>
<Condition  expsourcequant="exists"
        expsource="./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[2]/tid:partyReference"
        op="equal"
        dest_ref="pR"
        expdestquant="exists"
        expdest="./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[2]/tid:partyReference"/>
</ConsistencyRule>
```

Type 3: Existence of documents due to the existence of elements

This type of rule is related to the situation in which the existence of one or more elements requires the existence of a document.

Example: (Type 3.2) - For every exchange rate in a FX swap trade document F_1, there must exist a document F_2 with all the existing exchange rates.

```
<ConsistencyRule id=”R3.2”>
<Description> For every exchange rate in a FX swap trade document, there must exist
a document with a list of all the exchange rates.</Description>
<Source> <XPath> /fpml:FpML/fpml:Trade/fpml:product/decendant::fxs:exchangeRate
</XPath> </Source>
<Destination dest_id=”eR”><XPath>/DocumentType/Rate/r:FixedRate/
</XPath></Destination>
<Condition  expsourcequant=”forall”
              expsource=”.”
              op=”equal”
              dest_ref=”eR”
              expdestquant=”exists”
              expdest=”.”/>
</ConsistencyRule>
```

Type 4: Existence of related documents

This type of rule is related to the situation in which the existence of a document requires the existence of another document.

Example: (Type 4.2) - For every FX swap trade document F_1 referencing two parties pR_1 and pR_2, and produced by party pR$_1$, there must exist a FX swap trade document F_2, produced by party pR$_2$, with the same party reference name.

```
<ConsistencyRule id=”R4.2”>
<Description> For every FX swap trade document referencing two parties and produced by one
party, there must exist a FX swap trade document produced by the other party, with the same
party reference names </Description>
<Source> <XPath> /fpml:FpML/fpml:Trade </XPath> </Source>
<Destination dest_id=”ST”> <XPath> /fpml:FpML/fpml:Trade </XPath> </Destination>
<Condition  expsourcequant=”forall”
        expsource=”./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[1]/tid:partyReference”
        op=”equal”
        dest_ref=”ST”
        expdestquant=”exists”
        expdest=”./fpml:tradeIDs/fpml:TradeIDs/ tid:TradeID/tid:partyReference”/>
<Operatorvalue=”AND”/>
<Condition  expsourcequant=”forall”
        expsource=”./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[2]/tid:partyReference”
        op=”equal”
        dest_ref=”ST”
        expdestquant=”exists”
        expdest=”./fpml:tradeIDs/fpml:TradeIDs/tid:TradeID/tid:partyReference”/>
</ConsistencyRule>
```

With the proposed syntax it is also possible to represent consistency rules that check for the existence of unrelated documents or elements. Due to limitation of space we do not present here examples of these types of consistency rules.

4 Resolution Actions

In this section we present the syntax used to express the resolution actions. Similar to the consistency rules, and for the same reasons, the syntax to express the resolution action is also described in terms of XML [4] and XPath [7] syntax.

A resolution is composed of two parts. Part 1 is related to *fragment parts* of the participating documents that are inconsistent. Part 2 is concerned with *values* that should be replaced in the fragment parts of part 1, to convert the document to a consistent state.

Fig. 3 illustrates the Document Type Definition [4] for the resolution action syntax. It contains a root element called Action composed of two element contents. The two element contents are described below. Fig. 4 presents an example of a resolution action for consistency rule R4.2 (Type 4) shown in section 3

- DocumentFragment - it contains XPath expressions identifying the part of the document where inconsistency was detected;
- NewValue – it contains the 'new' value to which the fragment part has to be modified to restore the document to a consistent mode.

```
<!ELEMENT Action (DocumentFragment, NewValue)>
<!ELEMENT DocumentFragment (XPath)>
<!ELEMENT NewValue (#PCDATA)>
<!ELEMENT XPath (#PCDATA)>
```

Fig. 3. Resolution action syntax

```
<Action>
<DocumentFragment> <XPath>
   fpml:FpML/fpml:Trade/fpml:tradeIDs/fpml:TradeIDs/tid:TradeID[1]/
   tid:partyReference </Xpath> </DocumentFragment>
<NewValue>ABC Trust
</NewValue>
</Action>
```

Fig. 4. Example of a resolution action

Based on the action a stylesheet is created to support the restoration of an inconsistent document. Fig. 5 presents an example of a XSLT stylesheet related to the resolution action shown in Fig. 4. The stylesheet is composed of two template rules. The first template rule matches all the attributes and nodes in the original XML document (source tree), and creates a new XML document (result tree) by copying these attributes and nodes. The second template rule is related to the content of the resolution action. It matches the nodes related to the inconsistent part of the document, and replaces them with the 'new value' specified in the resolution action.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<xsl:stylesheet version=”1.0”  xmlns:xsl=http://WWW.w3.org/1999/XSL/Transform>
<xsl:output method=”xml”/>

<xsl:template match=”@*|node( )”/>
  <xsl:copy>
    <xsl:apply-templates select=”@*|node( )”/>
  </xsl:copy>
</xsl:template>

<xsl:template  match=”fpml:FpML/fpml:Trade/fpml:tradeIDs/tid:TradeIDs/tid:TradeID[1]/
          tid:partyReference”/>
  ABC Trust
</xsl:template>
</xsl:stylesheet>
```

Fig. 5. An example of the XSLT stylesheet

5 The XRule Tool

In order to evaluate our approach, we developed a prototype tool called *XRule,* which supports the consistency management process. The tool was implemented as proof of concept and developed in JDK 1.1.6. It uses Apache [1] Xerces Java Parser 1.1.2, for parsing XML documents, and Apache Xalan Java XSLT Processor 1.1, as the XSLT processor.

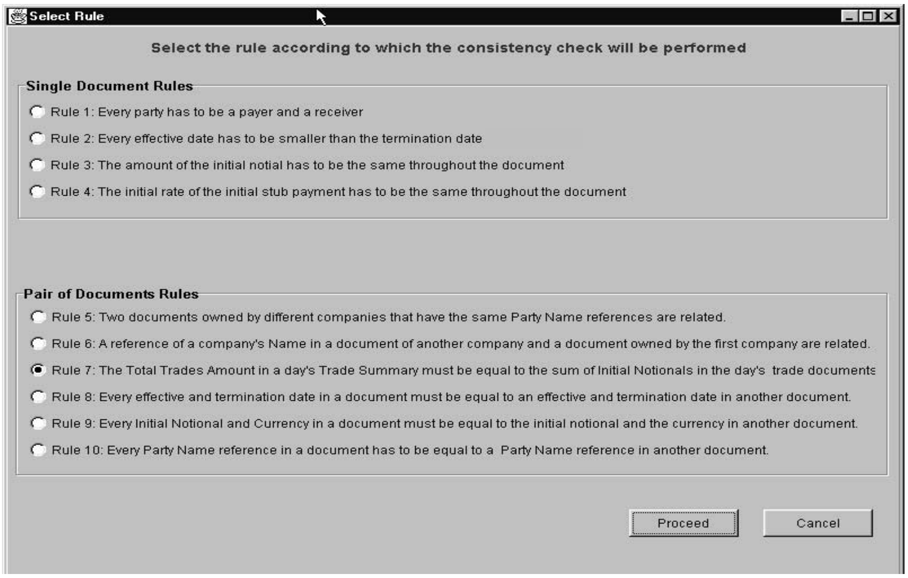
The prototype contains the implementation of essential features that enable us to evaluate and prove the feasibility of our approach. The main goals of our tool are to perform consistency checks and restore the documents to a consistent state. The document restoration is based on interaction with the user. This interaction is necessary to identify the correct instance value of the inconsistent document part.

Fig. 6 presents the initial screen of the XRule tool. It contains a list of all available consistency rules. For the prototype we have implemented ten different types of rules. We grouped these rules into two categories*: single document rules*, for the rules related to only one document, and *pair of document rules*, for the rules related to two or more documents.

After selecting a consistency rule, the documents to be checked for consistency are specified by the user. These documents can be located in the same machine where XRule tool is being used or accessed over the Web. The result of the consistency checking process is presented to the user, as shown in Fig. 7. For the example we consider the execution of only one trade on 26/05/1999. Therefore, there are only two documents checked for consistency rule 7 in Fig. 6. The first document (Source) is related to the trade summary; the second document (Destination) is related to the trade itself.

When an inconsistency is found, the application presents the fragment parts that are inconsistent in both documents, with their respective values, as shown on the top of Fig. 7. The two participating documents are also displayed on the screen to allow the user to browse the documents, if necessary.

In the case where the user wants to execute the consistency handling process, s/he selects to “proceed”. The application presents to the user the screen shown in Fig. 8 with the condition in the related consistency rule that does not hold for the participating documents. After selecting the condition the application automatically displays the related inconsistent values in the fragment of the participating documents. The user specifies which document is inconsistent, by selecting either the *Source* or *Destination* value, and specifies the ‘new value’ that should replace the inconsistent one. For our example the user assumes that the Destination document is inconsistent, i.e. the document related to the trade executed on 26/05/1999.


Select Rule

Select the rule according to which the consistency check will be performed

Single Document Rules

- Rule 1: Every party has to be a payer and a receiver
- Rule 2: Every effective date has to be smaller than the termination date
- Rule 3: The amount of the initial notial has to be the same throughout the document
- Rule 4: The initial rate of the initial stub payment has to be the same throughout the document

Pair of Documents Rules

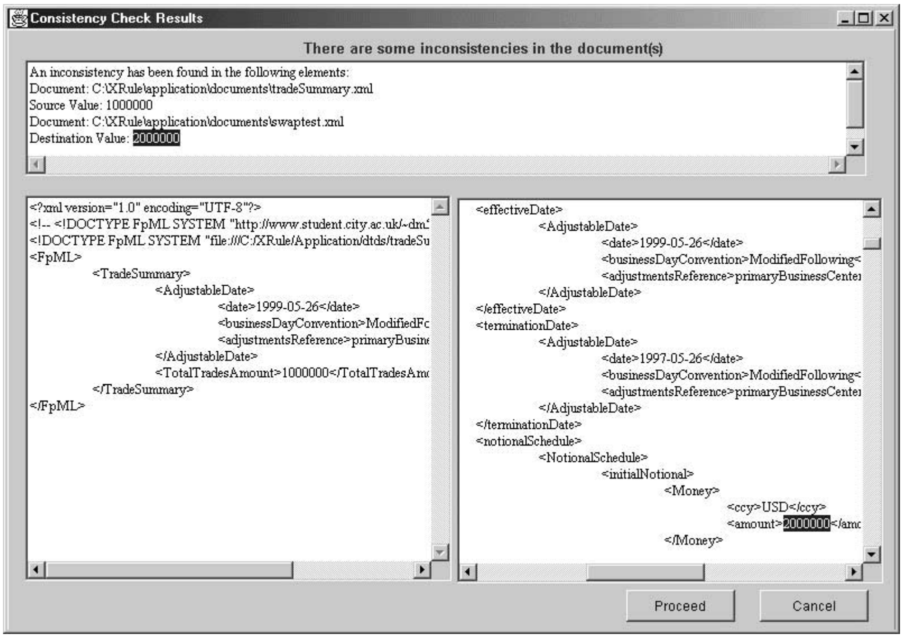
- Rule 5: Two documents owned by different companies that have the same Party Name references are related.
- Rule 6: A reference of a company's Name in a document of another company and a document owned by the first company are related.
- [✓ Rule 7: The Total Trades Amount in a day's Trade Summary must be equal to the sum of Initial Notionals in the day's trade documents]
- Rule 8: Every effective and termination date in a document must be equal to an effective and termination date in another document.
- Rule 9: Every Initial Notional and Currency in a document must be equal to the initial notional and the currency in another document.
- Rule 10: Every Party Name reference in a document has to be equal to a Party Name reference in another document.

Fig. 6. Consistency rules

Based on the information specified by the user, XRule generates the resolution action document and the XSLT stylesheet document. The tool executes the restoration process, and a consistent document is generated and presented to the user.

6 Related Work

Many approaches have been proposed for consistency management. In particular, approaches for software engineering documents and specifications. A complete and up to date survey can be found in [31].



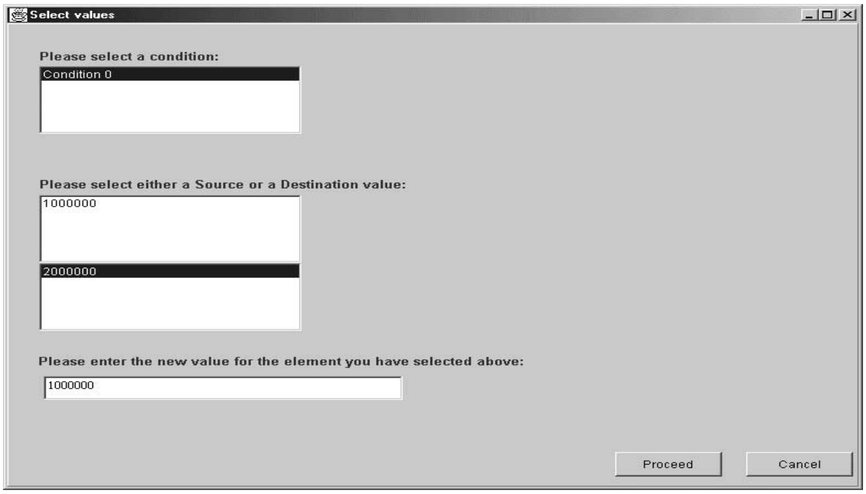
There are some inconsistencies in the document(s)

```
An inconsistency has been found in the following elements:
Document: C:\XRule\application\documents\tradeSummary.xml
Source Value: 1000000
Document: C:\XRule\application\documents\swaptest.xml
Destination Value: 2000000
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- <!DOCTYPE FpML SYSTEM "http://www.student.city.ac.uk/~dm
<!DOCTYPE FpML SYSTEM "file:///C:/XRule/Application/dtds/tradeSu
<FpML>
        <TradeSummary>
                <AdjustableDate>
                        <date>1999-05-26</date>
                        <businessDayConvention>ModifiedFc
                        <adjustmentsReference>primaryBusine
                </AdjustableDate>
                <TotalTradesAmount>1000000</TotalTradesAm
        </TradeSummary>
</FpML>
```

```
<effectiveDate>
        <AdjustableDate>
                <date>1999-05-26</date>
                <businessDayConvention>ModifiedFollowing<
                <adjustmentsReference>primaryBusinessCente
        </AdjustableDate>
</effectiveDate>
<terminationDate>
        <AdjustableDate>
                <date>1997-05-26</date>
                <businessDayConvention>ModifiedFollowing<
                <adjustmentsReference>primaryBusinessCente
        </AdjustableDate>
</terminationDate>
<notionalSchedule>
        <NotionalSchedule>
                <initialNotional>
                        <Money>
                                <ccy>USD</ccy>
                                <amount>2000000</amc
                        </Money>
```

Fig. 7. Result of the consistency checking process



Please select a condition:

Condition 0

Please select either a Source or a Destination value:

1000000

2000000

Please enter the new value for the element you have selected above:

1000000

Fig. 8. Consistency handling process

In [8][9][13] inconsistency is seen as a logical concept and the authors proposed a first-order logic-based approach to consistency management in the ViewPoints framework. However, this approach has not been implemented in a distributed setting.

Spanoudakis and Finkelstein [27][28] suggested a method called *reconciliation* to allow detection and verification of overlaps and certain types of inconsistencies between specifications expressed in an object-oriented framework. When managing inconsistency, overlap detection is an activity that precedes consistency rule construction [14]. We are investigating the use of reconciliation method to check consistency of meta-level XML documents, i.e. XML Schema documents [12].

In [8][13][18][29][33] the authors proposed logic-based approaches for consistency checking, where some formal inference technique is used to detect inconsistencies in software models expressed in formal modeling languages. Our work complements the work in [10], where the authors developed a technique for detecting inconsistencies in distributed documents with overlapping content, based on XML and related technologies.

The strategies proposed for consistency handling can be divided into two groups. One group is related to the approaches that use actions, which modify the documents by repairing or ameliorating the inconsistencies [8][9][20][23][33]. The other group is concerned to approaches that notify the stakeholders about inconsistencies and perform analysis that would safe further reasoning from documents [3][13][18].

Van Lamsweerde et al. [32][33] proposed a formal framework for various types of inconsistencies that can occur during requirements engineering process. The idea is to manage conflicts at the goal level in the context of the KAOS requirements engineering methodology. This is achieved by introducing new goals or by transforming specifications of goals into new specifications free of conflicts.

In addition, methods for consistency tracking have been proposed in [11][13]. On the other hand, specification and application of consistency management policies are presented in [11][13][24][28].

Identification and resolution of semantic and syntactic conflicts are also issues in the multidatabase system domain. Many approaches have been proposed in the literature [5][17][25][26]. A survey of different approaches to detect and resolve conflicts can be found in [2].

Although the existing approaches have contributed to a better understanding of the consistency management problem, an approach which deals with consistency management of distributed financial documents have not yet been proposed.

7 Conclusion and Future Work

In this paper we presented an approach to consistency management of XML financial documents. The approach supports the activities of consistency checking and consistency handling. It uses XML and related technologies to allow Internet-scale distribution, standardisation of the consistency management process, and access to XML documents.

We proposed the use of *consistency rules* and *resolution actions* to support the management process. We developed a prototype tool as proof of concept to evaluate the ideas of the work and demonstrate the feasibility and applicability of the approach.

Although the approach has been proposed for dealing with inconsistencies in the financial domain, it can be deployed in other settings where consistency management is necessary and important. Examples are found in the health care domain, scientific domain, and business domains, among others.

Before large-scale experimentation and use, we are expanding the prototype to allow consistency management of heterogeneous financial documents such as FIXML [16], FpML [15], and OTP [22], for meta-level (XML Schema) and instance documents. We are also extending our work to allow different types of consistency rules and resolution actions. In particular, rules and actions involving complex financial calculations like derivative models, and semantic aspects of the data. In [31] we proposed an approach for monitoring financial information where we present a syntax to describe complex financial calculations. We also plan to expand the approach to support other important activities of the consistency management process, such as consistency tracking, consistency diagnosis, and consistency policy [31].

References

[1] Apache. http://xml.apache.org/index.html.

[2] C. Batini, M. Lenzerini, and S.B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computer Surveys*, 18(4), pages 323-364, December 1986.

[3] B. Boehm and H. In. Identifying Quality Requirements Conflicts. *IEEE Software*, pp. 25-35, March 1996.

[4] T. Bray, J. paoli, C.M. Sperberg-McQueen, E. Maler. Extensible Markup Language (XML) 1.0. W3C Recommendation, http://www.w3.org/TR/2000/REC-xml-20001006, World Wide Web Consortium.

[5] M.W. Bright, A.R. Hurson, and S. Pakzard. Automated Resolution of Semantic Heterogeneity in Multidatabases. *ACM Transaction on Database Systems*, 19(12), pages 212-253, June 1994.

[6] J. Clark. XSL Transformations (XSLT) Version 1.0. Recommendation http://www.w3.org/TR/1999/REC-xslt-19991116, World Wide Web Consortium.

[7] J. Clark and S. DeRose. XML Path Language (XPath). Recommendation http://www.w3.org/TR/1999/REC-xpath-19991116, World Wide Web Consortium.

[8] S. Easterbrook, A. Finkelstein, J. Kramer, and B. Nuseibeh. Co- ordinating Distributed ViewPoints: the anatonomy of a consistency check. In *Concurrent Engineering Research & Applications,* CERA Institute, USA 1994.

[9] S. Easterbrook and B. Nuseibeh. Using ViewPoints for Inconsistency Management. *IEE Software Engineering Journal*, November 1995.

[10] E.Ellmer, W. Emmerich, A. Finkelstein, D. Smolko, and A. Zisman. Consistency Management of Distributed Documents using XML and Related Technologies. UCL-CS Research Note 99/94, 1999. Submitted for publication.

[11] W. Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, and S. Armitage. Managing Standards Compliance. *IEEE Transactions on Software Engineering*, 25(6), 1999.

[12] D.C. Fallside. XML Schema Part 0: Primer. Working Draft http://www.w3.org/TR/2000/WD-xmlschema-0-20000407, World Wide Web Consortium.

[13] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency Handling in Multi-Perspective Specifications. *IEEE Transactions on Software Engineering,* 20(8), pages 569-578, August 1994.

[14] A. Finkelstein, G. Spanoudakis, and D. Till. Managing Interference. Joint Proceedings of the SIGSOFT'96 Workshops – Viewpoints'96: An International Workshop on Multiple Perspectives on Software Development, San Francisco, ACM Press, pages 172-174, October 1996.

[15] FpML. Financial product Markup Language. http://www.fpml.org.

[16] FIXML. Financial International Exchange Markup Language. http://www. fix.org.

[17] J. Hammer and D. McLeod. An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(1), pages 51-83, 1993.

[18] A. Hunter and B. Nuseibeh. Managing Inconsistent Specifications: Reasoning, Analysis and Action. *ACM Transactions on Software Engineering and Methodology*, 7(4), pp. 335-367, 1998.

[19] Infinity. Infinity Network Trade Model. http://www.infinity.com/ntm.

[20] B. Nuseibeh and A. Russo. Using Abduction to Evolve Inconsistent Requirements Specifications. *Australian Journal of Information Systems*, 7(1), Special Issue on Requirements Engineering, ISSN: 1039-7841, 1999.

[21] OMG (1998). XML Metadata Interchange (XMI) - Proposal to the OMG OA&DTF RFP 3: Stream-based Model Interchange Format (SMIF). Technical Report AD Document AD/98-10-05, Object Management Group,m 492 Old Connecticut Path, Framingham, MA 01701, USA.

[22] OTP. Open Trading protocol. http://www.otp.org.

[23] W. Robinson and S. Fickas. Supporting Multiple Perspective Requirements Engineering. *In Proceedings of the 1st International Conference on Requirements Engineering (ICRE 94)*, IEEE CS Press, pp. 206-215, 1994.

[24] W. Robinson and S. Pawlowski. Managing Requirements Inconsistency with Development Goal Monitors. *IEEE Transactions on Software Engineer*, 25(6), 1999.

[25] E.Sciore, M. Siegel, and A. Rosenthal. Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems. *ACM Transactions on Database Systems*, 19(2), pages 254-290, June 1994.

[26] M. Siegel and S.E. Madnick. A Metadata Approach to Resolving Semantic Conflicts. *In proceedings of the 17th International Conference on Very Large DataBases,* pages 133-145, Barcelona, Spain, 1991.

[27] G. Spanoudakis and A. Finkelstein. Reconciliation: Managing Interference in Software Development. *In Proceedings of the ECAI '96 Workshop on Modelling Conlicts in AI*, Budapest, Hungary, 1996.

[28] G. Spanoudakis and A. Finkelstein. A Semi-automatic Process of Identifying Overlaps and Inconsistencies between Requirements Specifications. *In Proceedings of the 5th International Conference on Object-Oriented Information Systems* (OOIS 98), pages 405-425, 1998.

[29] G. Spanoudakis, A. Finkelstein, and D. Till. Overlaps in Requirements Engineering. *Automated Software Engineering Journal*, vol. 6, pp. 171-198, 1999.

[30] G. Spanoudakis and A. Zisman. Information Monitors: An Architecture Based on XML. *In Proceedings of 6th International Conference on Object Oriented Information Systems – OOIS 2000*, London, December 2000.

[31] G. Spanoudakis and A. Zisman. Inconsistency Management in Software Engineering: Survey and Open Research Issues. *Handbook of Software Engineering and Knowledge Engineering*, 2000. (To appear).

[32] A. van Lamsweerde. Divergent Views in Goal-Driven Requirements Engineering. *In Proceedings of the ACM SIGSOFT Workshop on Viewpoints in Software Development,* San Francisco, pages 252-256. October 1996.

[33] A. van Lamsweerde, R. Darimont, and E. Letier. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transaction on Software Engineering.* November 1998.

[34] A. Zisman, W. Emmerich, and A. Finkelstein. Using XML to Specify Consistency Rules for Distributed Documents, *In Proceedings of the 10th International Workshop on Software Specification and Design (IWWSD-10),* Shelter Island, San Diego, California, November, 2000.