Interest Rate Financial Instruments - XML Representation and Calculation of Cash Flow Streams

Sevdalina Trifonova
Svetoslav Kyuchukov

Abstract: *This paper presents a methodology based on the XML technology, for describing financial products and their cash flow streams. Nowadays financial instruments become more complex and our purpose is to find a general approach as to describe their conditions. Interest rate and capital cash flows are defined in XML. XML structure is dependent on grammar rules that are defined in assigned DTD. DOM validates source XML file and construct evaluation tree. Java package calculates interest and capital payments.*

Key words: *Computer Systems and Technologies, eXtensible Markup Language (XML), Financial Products, DTD (Document type definition), DOM (Document Object Model), Context- free grammar*

INTRODUCTION

Financial institutions offer a wide range of financial products to their clients. Increasingly, data about these products are electronically exchanged in ordering, trading, and confirmation processes among financial institutions, supervisors and institutional clients.

One of the most popular financial products is credit. Generally, credit is a over-the - counter (OTC) financial product, which is not traded on recognized stock exchanges (e. g London stock exchange or Sofia stock exchange) and has traditionally been less standardized. Credit is a contract between a financial institution and a client; a bank borrows money to the client and the latter is obligated to pay an interest rate and return the money until the maturity date.

There are three standardized methods of amortization – Annuity (the sum of interest and amortization payments is constant for each period), Bullet (amortization at maturity) and Regular (equal amortization payments in all periods).

The interest rate convention is fixed or floater. Fixed interest rate means constant interest rate from issue date to maturity. Floater interest rate is fixed on interbanking rate (e.g. LIBOR, FIBOR, SOFIBID, and SOFIBOR). Amortization and interest rate payment dates are determined according to standard frequency conventions (Monthly, Quart, Annual etc.)

Nowadays financial institutions are competing as to attract clients using for the purpose promotions and special conditions. Financial institutions offer corporate clients non-standardized amortization plans that match their work-cycle (dull and active seasons). Complicated definition of interest rates becomes more popular. Additionally, financial institutions use optional rights to achieve maximal profit and to limit loss (cap option defines maximal interest rate, which is payable, floor – minimal interest rate)

Hard coded systems are not adequate for describing this variety of conditions because the appearance of a new condition leads to changes in procedures and even in GUI.

Thus, we have focused on finding a flexible way to describe interest rate financial instruments. This paper describes a flexible system based on XML technology and Java.

CONSTRUCTING CASH FLOWS OF INTEREST RATE FINANCIAL INSTRUMENTS

We have selected XML because it is a widespread technology for sharing data between applications and many of the newest financial systems support different XML standards [3]

XML presents contract's conditions as divided into the following several main groups:

- common data of credit (position id, issue date, maturity, currency, notional amount, paid value, interest basis etc.)
- customer data – customer id , customer name
- conventions determining actual payment and fixing (city, dateroll, mode). The conventions adjust calendar date to financial date for the selected financial center
- cash flow streams. The formula and the list of payment dates define rate and capital payments.

The structure sets up on XML tags and their attributes as shown below:

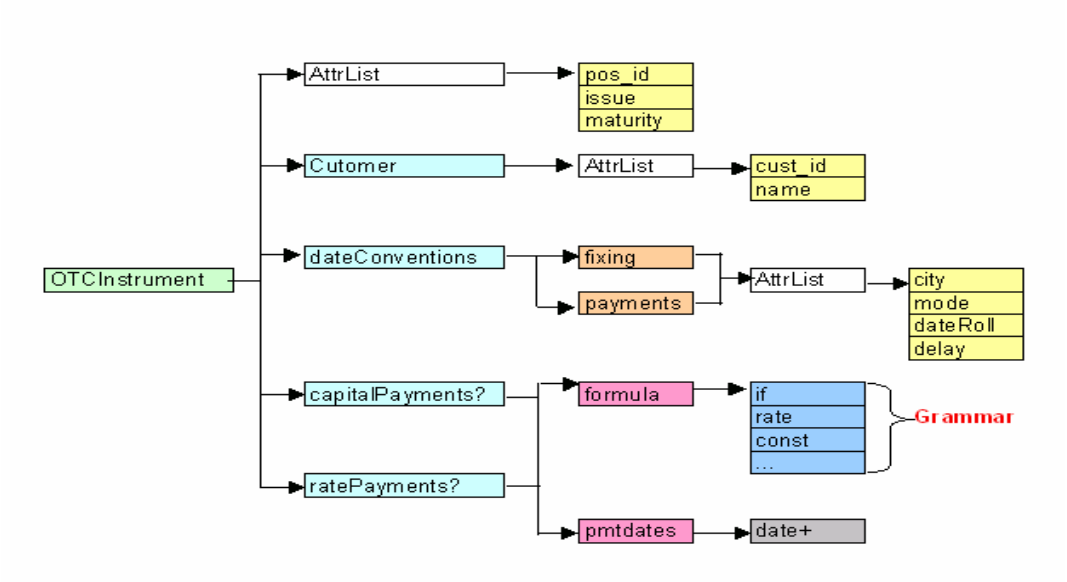

Figure 1 XML financial product structure

DTD (Document Type Definition) schema language is used to match XML source file to appropriate structure. Also, DTD is a grammar–based language and is used to describe grammar rules as indicated below.

The most important feature of this implementation is the ability to freely define conditions for interest rate and amortization payments. Transformation from formula to instrument cash flow streams is based on grammar rules that are validated by DOM parser and transformed to evaluation trees that are applied in Java package.

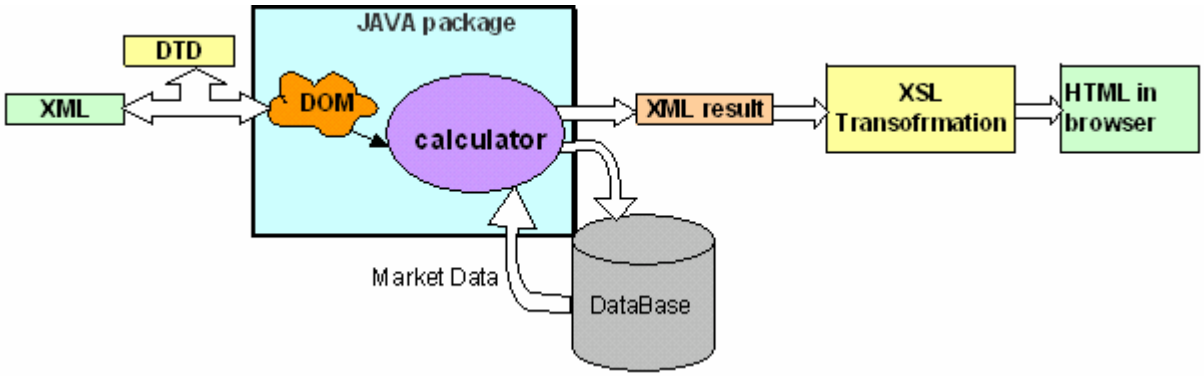

Figure 2 Structure of the system

The computing module (Java calculator) runs methods corresponding to the evaluation tree and extracts actual market data (interbanking rates, discount interest rates, foreign exchange rates etc) from the database.

There are two possibilities for keeping cash flow streams built:

- export to XML where data could be shown in WEB browser after XSL transformation to HTML page. Also, XML file could be sent to other financial institution, a supervisor or a client
- store to database where data could be used by another applications for analysis, calculation of risks, etc.

GRAMMAR RULES

We have researched many different complex contracts and developed a context-free grammar, which covers definition of complex capital and interest rate payments formulas.

Table 1 Grammar Rules

Grammar Rules	Description
Factor=>Expression | Const | Fixing | IRPay | IF | Parameter|IRR…	
Const => <const> NumberConst | Parameter </const>	Constant value
Fixing =><rate> Const <datefrom>From</datefrom> <dateto>To</dateto> < curve>Curve</curve> <ibrate>IBRate</ibrate> </rate>	Rate is fixed from explicit curve or interbanking index.
IRPay=><irpay> Const < datefrom>From </datefrom> <dateto>To</dateto > </irpay>	Total Rate payment for period
From, To => Date Sign PerExpr | Date	Date expression
PerExpr => Period <mult/> *NumberConst* | Period	Period expression
Date =><date> *DateConst* | issue| maturitypend | damo|dpay….. </date>	Date refers some special dates of instrument and instrument's period
Period => <period> pfreq | afreq |dfreq| NumberConst (y |m)… </period>	Definition of period length.

There are the instrument parameters that refer to:

- specific dates (issue, maturity, period start (pstart), period end (pend), amortization date)
- frequency and period length (interest rate frequency (pfreq), amortisation frequency (afreq))
- common parameters (paidvalue, notamount, remaining debt, short, and long.)

We use Document Object Model (DOM) to validate XML source file according to grammar rules presented by DTD

Table 2 XML data and DTD representation of grammar rules

XML data	Document Type Definition
<CapitalPayments count="7"> <formula> <if> <condition> <date>damo</date> <equal/> <date>maturity</date> </condition> <if-true> <const>remainingdebt</const>	<!ELEMENT CapitalPayments (formula,pmtdates?)> <!ATTLIST CapitalPayments count CDATA #REQUIRED> <!ELEMENT RatePayments (formula,pmtdates?)> <!ATTLIST RatePayments count CDATA #REQUIRED> <!ELEMENT pmtdates (pdate*)> <!ELEMENT pdate (#PCDATA) > <!ATTLIST pdate id CDATA #REQUIRED> <!ELEMENT formula ((plus?, minus?,if)*,(plus?, minus?,const)*,(plus?, minus?,rate)*,(plus?, minus?,irpay)*)> <!ELEMENT const (#PCDATA) >

</if-true> <if-false> <const>100</const> <minus/> <irpay> <datefrom> <date>damo[-1]</date> </datefrom> <dateto> <date>damo</date> </dateto> </irpay> </if-false> </if> </formula> <pmtdates> <pdate id="1">1.04.2003</pdate> <pdate id="2">1.07.2003</pdate> .. <pdate id="7">1.04.2005</pdate> </pmtdates> </CapitalPayments>	<!ELEMENT if (condition,if-true,if-false?)> <!ELEMENT if-true (((plus?, minus?,if)*,(plus?, minus?,const)*,(plus?, minus?,rate)*,(plus?, minus?,irpay)*))> <!ELEMENT if-false (((plus?, minus?,if)*,(plus?, minus?,const)*,(plus?, minus?,rate)*,(plus?, minus?,irpay)*))> <!ELEMENT condition (date?,(equal?,less?,greater?,date)?)> <!ELEMENT irpay (const?,datefrom,dateto)> <!ELEMENT datefrom (date?,(plus?, minus?,period)?)> <!ELEMENT dateto (date?,(plus?, minus?,period)?)> <!ELEMENT date (#PCDATA)> <!ELEMENT plus EMPTY> <!ELEMENT minus EMPTY> <!ELEMENT equal EMPTY>

The table above shows XML definition of capital payments. Payment dates are explicitly listed in a *pmtdates* element. The *formula* sub-tree describes the evaluation tree of amortisation.

In this case, the formula indicates that

If amortisation date is equal to maturity then

capital payment is equal to remaining debt

else

100 EUR – (sum of rate payments for period from previous amortisation date to current amortisation date)

The second column shows DTD representation of grammar rules, e.g. *if* structure must consist *of condition, if-true, if-false* XML elements in appropriate order and also *if-false* could be missing because *?* declares one or none occurrences of the element.

<!ELEMENT if (condition,if-true,if-false?)>

ALGHORITHM IMPLEMENTED IN THE CALCULATOR

The main functionality implemented in the computing module is the construction of evaluation tree and cash flow streams.

The first step of cash flow calculation is determining the time structure based on the following dates - of issue, maturity, capital and interest rate payment.

Capital and interest rate payment dates could be calculated according to frequency conventions,

$$PaymentDate_i = PaymentDate_{i-1} + Frequency \quad (1)$$

$$PaymentDate_0 = IssueDate \quad (2)$$

or they could be explicitly defined in XML when they are not regular

<pmtdates>
<pdate id="1">1.04.2003</pdate>
<pdate id="2">1.07.2003</pdate> ..
<pdate id="7">1.04.2005</pdate>
</pmtdates>

In the second case the algorithm extracts fixed dates from the source XML file and builds the time structure of the contract.

Two important restrictions must be always applied:

- the bank pays the notional amount or the first part of the notional amount to the client at issue date and the contract begins;
- the maturity date is the last rate and capital payment date of the contract.

For the purpose of the algorithm describing we have divided the calculating module into two calculators-one for the capital payments and one for the rate payments. But in fact calculators are implemented in one Java package as classes.

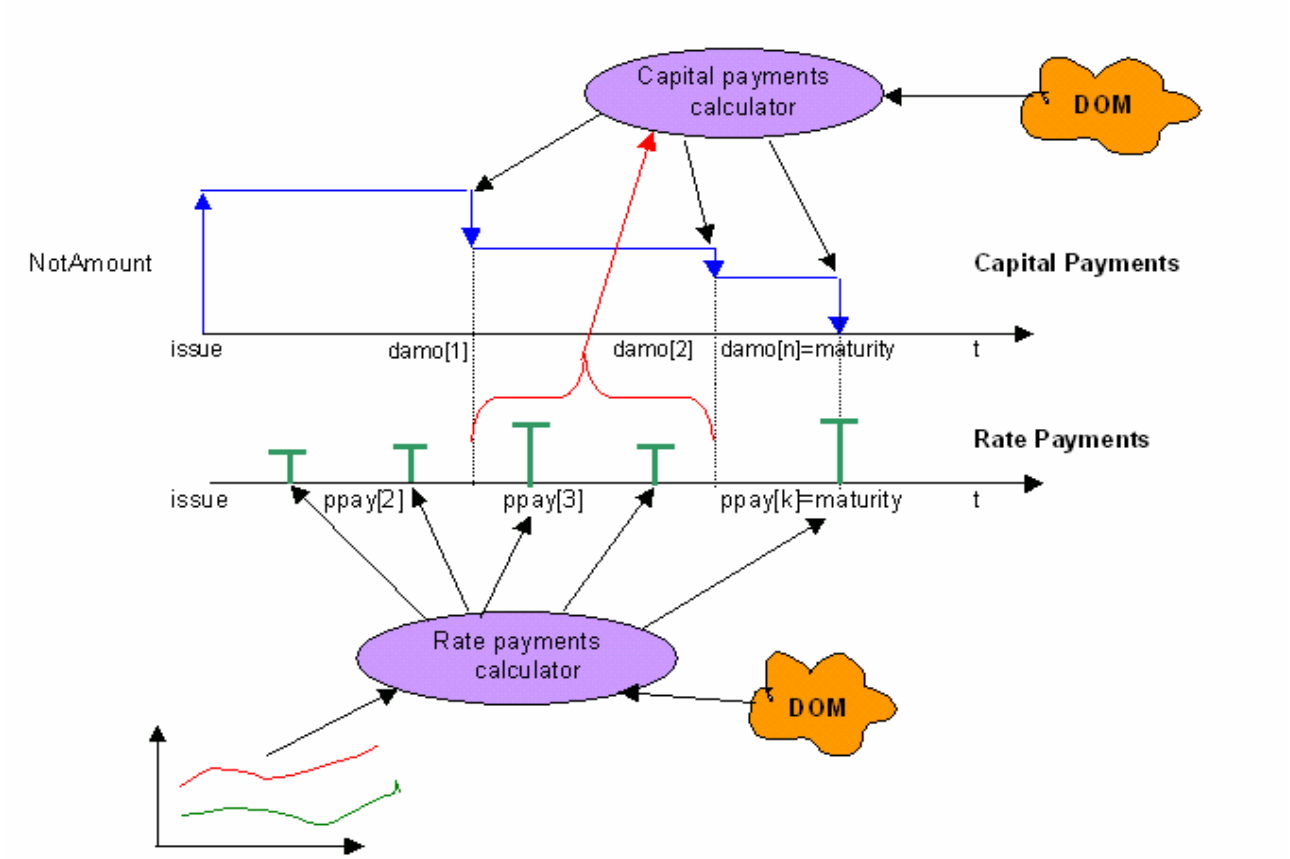

Figure 3 Time structure and cash flow streams of a contract

Rate payment calculator uses a formula loaded in DOM. In the common case, the formula refers to one or more methods implemented in calculator. Methods load market data from the database and calculate the interest rate of the period as percentage.

The next step is estimating the period length according to the convention interest base (e.g. ACT/360 - length of period in years is calculated as actual length in days divided in 360 days per year)

$$\Delta t(pstart, pend) = (pend - pstart)/360 \quad (3)$$

and the rate payment is

$$RatePayment_i = fixing(pstart, pend) * DebtSquare(pstart, pend) \quad (4)$$

where *fixing* is the period interest rate as percentage

Debt Square shows dependence of the rate payments on amortisation.

$$DebtSquare(pstart, pend) = \sum \mathrm{Re}\, mainingDebt_j * \Delta t(d_j, d_{j+1}), d_j \in [pstart, pend] \quad (5)$$

The remaining debt is a function of amortisation payments

$$\mathrm{Re}\, mainingDebt_k = \mathrm{Re}\, mainingDebt_{k-1} - AmortizationPayment_{k-1} \quad (6)$$

Capital payment formula describes the evaluation tree of amortisation payments. Another dependence exists in the case shown in the previous section – the current amortisation payment depends on the previous rate payments run into a period between the previous and the current amortisation payment.

$$AmortizationPayment_k = \mathrm{Re}\, mainingDebt_{k-1} - \sum_{l=1}^{n} RatePayment_l \quad (7),$$

where n is the number of rate payments between k-1 and k-th capital payment dates.

The payment dates must be read in consecutive order because the current interest payment depends on the previous amortisation payment and the opposite one. It is not possible to separately calculate an interest rate and amortisation cash flow streams.

We manage dependencies providing communication between capital payments and interest rate payment calculators. The calculators are able to access each other's structure and to extract payable amounts.

CONCLUSIONS AND FUTURE WORK

The Java package is the first part of the entire software system that not only supports cash flow streams construction, but also calculates credit, market and operational risks of instruments and portfolios.

Several XML standards for describing financial deals exist (e.g. FpML puts accent on trading of derivatives; SWIFT is a standard for money transfer). But no one focuses on describing intricate instruments and especially credits.

One of our goals is to keep the system open for adapting to XML standards and using the same methodology to take to other financial products (e.g. credit, interest rate derivatives, options, futures in FpML format).

REFERENCES

[1] Assoc. Prof. Dr. A. Antonov, J. Janakieva, Bank and Financial Systems, 2004

[2] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 (Third Edition), http://www.w3.org/TR/2004/REC-xml-20040204/, 2004

[3] The XML Standard for Swaps, Derivatives and Structured Products, http://fpml.org/

[4] Johnny Stenback, Philippe Le Hégaret, Arnaud Le Hors. Document Object Model (DOM) Level 2 HTML Specification, http://www.w3.org/TR/DOM-Level-2-HTML/, 2003

[5] PriceWaterHouseCoopers, http://pwc.com/gx/eng/main/home/index.html

ABOUT THE AUTHOR

Sevdalina Trifonova, PhD Student, Department of Computer Science and Technology, Technical University of Varna, Phone +359 52 612 637,
E-mail: sevdraganova@eurorisksystems.com

Svetoslav Kyuchukov,"EuroRisk Systems" Ltd Varna, Phone +359 52 612 637,
E-mail: svet@eurorisksystems.com