Intelligent Representation of Accounting Knowledge

Bogdan Patrut
Faculty of Computer Science, "Alexandru Ioan Cuza" University of Iași, Romania
Bulevardul Carol I, 11, Iași 700506
Phone: +40 234 542411
bogdan@edusoft.ro

Simona Elena Varlan
"Alexandru Ioan Cuza" University of Iași, Romania
Bulevardul Carol I, 11, Iași 700506
Phone: +40 234 542411
varlan_simona@yahoo.com

Abstract

In this paper we will describe some methods for representing different kind of accounting and financial knowledge. These methods are integrated in an original architecture modeling the complete accounting analysis, starting by describing an economic operation in natural language, and finishing by writing the financial reports.

1. Introduction

Representing accounting knowledge within an intelligent system implies, first of all, the establishing of the competences that an accountant should have. Then, we must represent the rules, reasoning, norms, value judgments, laws and principles that lie at the basis of accounting. At the end, we must represent the financial reports in an adequate model.

Thus, hybrid intelligent systems can be conceived starting from the classic artificial intelligence systems such as the expert systems or intelligent agents. The final aim, hard to attain but not impossible, is realizing some complex programs that could assist and help the (human) accounting expert by taking over his/her tasks or even by replacing him/her.

In this paper we will describe some methods for representing different kind of accounting and financial knowledge. These methods are integrated in an original architecture modeling the complete accounting analysis, starting by describing an economic operation in natural language, and finishing by writing the financial reports.

2. The General Architecture

The proposed architecture is presented in Figure 1. It consists of the following elements:

- Documents – these refers to bills, invoices, contracts, and other input information
- Economic operations – these represents all kind of economic operations that can be described in natural language sentences
- Semantic interpreter – which will parse the natural language sentences and will automatically generate accounting entries
- Norms – which represents all the rules, norms and financial and accounting lows, GAAP, IFRS
- Accounting entries – are formal equations for representing economic operations
- Rules – represents general accepted account functioning rules (i.e. technical rules)
- Reports – which represents financial reports of an enterprise at the end of the financial period (month, semester , year)

In the Figure we can observe different ways of representing accounting knowledge: regular expressions, first order predicate calculus, production rules, semantic nets, XBRL and SKOS.

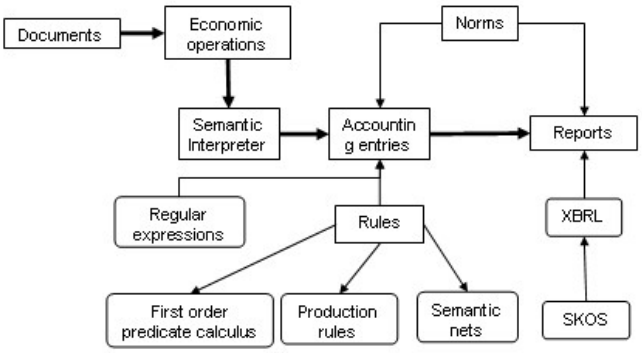

Figure 1. The general architecture

The bold arrows represent the flow of the process of complete accounting and financial operations, starting with reading or consulting the primary documents, and endind with the financial reports. It can be observed a special semantic interpreter, which can translate phrases describing economic operations into accounting entries. But the accounting entries are validated only if they respect the legal norms and the rules of functioning of the accounts. The regular expressions will be templates for accounting entries, as we will describe further. The rules of functioning of the accounts can be represented, as we will discuss further, by different ways: first order predicates calculus, production rules, semantic nets.

The financial reports, which represents the final stage of the accounting analysis process, will use the XBRL mark-up language for representing balance sheets or other kind of final reports. The SKOS language will be used to describe the taxonomy of the components of the financial reports.

3. Semantic Analysis for Economic Operations

This part of our paper deals with the "understanding" of the text afferent to an accounting entry. The procedure will parse the text, making the syntactical analysis, followed by an "interpretation" and "execution" of the analyzed text. Thus, we intend that by providing our system with the description of an accounting operation, to receive back the corresponding accounting entry.

For example[18], we would like that for phrases of the kind of those on the left to get answers like those on the right:

a) a social capital worth 3000 RON is subscribed => 456 = 1011, 3000 RON (RON = Romanian currency: sg. *leu*/pl. *lei*)

b) a land worth 2000 RON is deposited under the form of contribution in kind => 2111 = 456, 2000 RON

c) the social capital worth 3000 RON is entered as being paid => 1011 = 1012, 3000 RON

How could we make this possible, using the PROLOG language? First of all the phrases will be turned from a row of characters into lists of rows of characters, knowing that in the PROLOG language efficient predicates for working with lists can be implemented (the predicate "create_list" will deal with this). Then, from the respective list the "word" representing the numerical sum will be extracted (using the predicate "extract_sum").

In order to make things easier, words that are considered non-relevant for the phrases are taken out. Thus, neither prepositions nor other words that do not appear in a given dictionary, will not be taken into consideration. The predicate that simplifies a list goes through the list and analyses each word separately. This word is searched for in a dictionary (see the predicate "dictionary" and its clauses) and, if there is any, then it is replaced with its standard variant. The other variants are considered synonyms. In case the respective word does not exist in the dictionary, then it will no longer appear in the simplified list.

[18] the numerical codes or symbols for the accounts are specific to Romanian economy; in Romania, like in France, there is a General Accounts Plan, in which a numerical symbol represents an account; for example, "1011" represents "subscribed capital"

After eliminating the “unimportant” words, the remaining words will be alphabetically ordered (by the predicate "sort" that implements a procedure of sorting through insertion). Thus, sentences of the type "capital is deposited" or "deposited capital" will have the same meaning.

Finally, the predicate "translate" makes the desired translation is appealed to, thus reaching the required accounting entry.

To illustrate this, let us consider the phrase "a social capital worth 3000 RON is subscribed".

After the application of the first transformation ("make_list"), the following list is obtained ["is","subscribed","capital", "social", "worth", "3000", "RON"]. After that, the predicate ”extract_sum” will get the value of 3000 (RON) which will be finally displayed next to the accounting entry. By applying the predicate”simplify” to the obtained list of words, we are only left with the list [”subscribed”,”capital”], because the inarticulate standard variant “capital” has been chosen instead of “the capital”. Then, the list is sorted, resulting in ["capital","subscribed"] which, according to the translation given by the predicate "translate" leads to the accounting entry 456=1011.

4. The Representation of Accounting Knowledge

To exemplify the accounting knowledge representation models in a knowledge-based system, we will consider the four account functioning rules that can be expressed in natural language as follows:

1) If X is an assets account, X reflects Y and Y increases then X sells out.
2) If X is an assets account, X reflects Y and Y decreases then X credits.
3) If X is a liabilities account, X reflects Y and Y increases then X credits.
4) If X is a liabilities account, X reflects Y and Y decreases then X sells out.

For example, the first rule can apply to the account 512 that reflects the available in the current deposit on banks, in the case of payment of an invoice by a supplier, through payment order.

The first rule represents accounting knowledge and we can notice its generality as an essential feature. This knowledge can be represented in a computer in more ways and the encoding done by the programmer should take into consideration the conceptual models of knowledge representation studied by Artificial Intelligence. We will present a review of some of these models (Sowa, Course Technology) and we will exemplify the respective representations on rule 1.

a. First Order Predicates Calculus

In first order predicates calculus, predicates represent attributes of entities or relations among these and can be regarded as functions with the codomain of the lots of true and false truth values. Besides predicates the existential quantifier $\exists$ and the universal quantifier $\forall$ are used, as well as the operators of mathematical logics: conjunction $\wedge$, disjunction $\vee$, negation $\neg$, implication $\rightarrow$ etc.

Rule 1 above, expressed in natural language, can be represented in first order predicates calculus by the following formula:

$\forall x$ assets_account $(x) \wedge$ reflects$(x,y) \wedge$ increase$(y) \rightarrow$ sells_out(x)

The names of the predicates are chosen by us, and their semantic will be encoded in the system according to the needs.

b. Production Rules

Similar to first order predicates calculus, production rules are the most frequently used model of knowledge representation in intelligent systems, especially in the data bases of expert systems. Within such a system, knowledge is procedural and the entire system is built around production rules whose structure is the following:

<conditions> $\rightarrow$ <actions>

This means: IF <conditions> are true, THEN the <actions> can be executed.

Expert system generators like ExSys (http://www.exsys.com) or CLIPS (http://www.ghg.net/clips/CLIPS.html) are based on such production rules.

Researchers divide the production rules into two categories: deductive rules and inductive rules. Deductive rules are written under the form of the implication presented above, while the inductive ones under the form of a reverse implication. However, the difference between them is not syntactical. In the case of deductive rules we start from certain initial facts and, by applying the production rules we attempt to obtain new facts, finally reaching the intended aim, while in the case of inductive production rules we start with a purpose and, by applying the production rules "backwards" we attempt to demonstrate the purpose is attempted, on the basis of initial facts. Prolog is such an example of inductive system.

Deductive rules

In an expert deductive system generator the first functioning rule of accounts can be represented under the form

IF [X] is asset account and [X] reflects [Y] and [Y] increases THEN [X] debits.

Inductive - Prolog

The Prolog language has been designed particularly for artificial intelligence problems. It is based on first order predicates calculus and a Prolog programme defines a series of predicates presenting for each of them its clauses that is, the facts and rules that render true the predicate. A rule is seen as an implication $A1 \wedge A2 \wedge ... \wedge An \rightarrow B$, conversely written. In order to represent rule 1 of account functioning in Prolog we will have to specify some predicates for the quality of being an account (assets or liabilities), for increasing (decreasing) the accountable matter that some account or other would represent, for the relation of reflecting the accounting matter through a certain account and to express the fact that an account sells out (credits). Then we will write the mentioned accounting rule under the form of a clause, as bellow:

```
predicates
      accont(integer,symbol)   increases(string)
reflects(integer,string)    debits(integer)
      ...
clauses
      ...
      debits(X) if accont(X,a), reflects(X,Y), increases(Y).
```

c. Semantic Networks

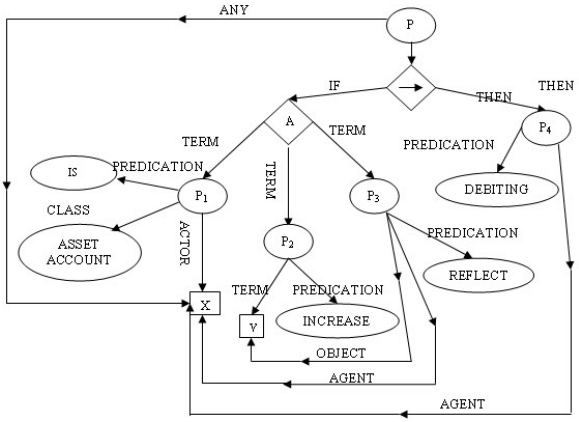

Figure 2. The debiting rules representation of asset accounts by a semantic network

Semantic networks constitute a much more abstract modality of knowledge representation, but a more general and efficient one also. Many researchers dealt with representing knowledge through semantic networks, but they are not so frequently used as the other models due to the difficulty of encoding them into a programme. We will not go into representation details but we can affirm that a semantic network is a graph whose knots contain sentence identifiers, logic connectives, entities etc., and the arc relations among these (class affiliation, subset, features, agent, predication

etc.). for example, the 1st functioning rule of accounts can be represented within a multi-sorted semantic network with the universal quantification of the variable x, as in figure 2.

5. Regular Expressions as Templates for Accounting Entries

Regular expressions represent a meta-language for describing words in a language (formal or natural), in a very simple and elegant formalised way. There are different variants for noting regular expressions, we will choose one that is sufficient for describing accounting formulas (Andone & Pătruț, 2007).

Regular expressions are rows of characters that represent patterns or models. They are built on the basis of a grammar, just like a programming language. These patterns are used to "recognise" and manipulate some rows of characters. Regular expressions are particularly used in text processing of any kind: programme writing, web pages generation.

For some chapters in financial accounting, the accounting formulas are simple. However, there are some extremely complex cases for which there is a wide diversity of formulas. All accounting handbooks present the various chapters of financial accounting with more or less ambiguous explanations in Romanian, with more or less relevant practical examples. For an accuracy of presentation the use of a formal and clear representation mechanism is required, and we consider fit the mechanism of regular expressions. Also, the sums that are involved in an accounting formula should respect certain relations and these have to be accurately presented.

To illustrate, will consider some accounting problems and we will present the representation modality of the adequate accounting recording by using regular expressions.

1. The constitution of the share capital

It is recorded the constitution of the share capital in a value s. Is also recorded the paid of share capital and its transformation in deposited subscribed and paid share capital.

456 = 1011 s

((5311|5121|5124|2xx|3xx|4xx) = 456)* [s]

2. The accountancy of the operations concerning financial assets

For the acquisition of some shares from another entity in amount a, an entity will do the next accounting recording:

((261|262|263)=(5121|269))* [a]

then

269 = 5121 [a_1] with a_1 <= a.

At the ending of the year there are recorded dividends afferent of the owned financial assets in amount d..

(5121 = (7611|7612|7613|7614|7615|7616|7617))* [d]

There is recorded the sale of the bought shares at a price p

461 = 7641 [p]

There is recorded the issue of sold shares from the administration

(6641 = (261|262|263))* [a]

3. An entity buys raw materials in amount [x] lei, the invoice including also TVA [x*0.19] RON. This is recorded in accounting, knowing that the entity uses the continuous inventory, like below:

301 = 401 [x]

4426 = 401 [x*0.19]

4. An entity realizes an informatics program whose production cost is [x] RON, for circulation accounting of the good purchased for resale, and also it buys another one for "The wages calculus and accounting" whose cost of purchase is [y] RON, with TVA [y*0.19] RON. The amortization term is for 3 years, for both the programs, so the monthly amortization is [(x+y)/36] RON.

a) It is recorded the realized informatics program by entity
208 = 721 [x]
b) It is recorded the invoice of bought informatics program.
208 = 404 [y]
4426 = 404 [x*0.19]
c) It is recorded the amortization for the first month of both the programs.
6811 = 2808 # [(x+y)/36]
d) At the ending of the amortization time, is recorded the issue of both the programs.
2808 = 208 # [x+y]
667 = 4111 # [x*(1-y/100)*(1-z/100)*t/100]

Even if our examples use numerical symbols from the Romanian General Accounts Plan, the bellow examples shows that the regular expressions are adecquate for writing templates for accounting entries.

6. XBRL – A New International Standard for Representing Financial Reports

The XBRL XBRL stands for eXtensible Business Reporting Language. It is one of a family of "XML" languages which is becoming a standard means of communicating information between businesses and on the internet. XBRL is being developed by an international non-profit consortium of approximately 450 major companies, organizations and government agencies. It is an open standard, free of license fees. It is already being put to practical use in a number of countries and implementations of XBRL are growing rapidly around the world (www.xbrl.org).

In this section we will present an examples of business report created using XBRL.

In this example we will use International Financial Reporting Standard-General Purpose (IFRS-GP) taxonomy adopted by the International Accounting Standards Board (IASB). The example expresses a balance sheet, statements about assets and equity and liabilities for two distinct periods for a virtual European company. The namespace prefix used here for the taxonomy is a recommended one: ifrs-gp. The human readable version of the financial report is the following:

Table 1. Two Balance Sheets

	2006	**2007**
Assets		
Investment property	400,000	500,000
Inventories	140,000	300,000
Construction in Progress	100,000	120,000
Current Tax Receivables	350,000	530,000
Prepayments	10,000	10,000
Cash and Cash Equivalents	650,000	900,000
Other Assets	60,000	70,000
Total Assets	1,710,000	2,430,000
Equity and Liabilities		
Issued capital	330,000	500,000
Reserves	100,000	230,000
Accumulated profits	1,000,000	1,500,000
Trade and other payables	200,000	200,000
Other current liabilities	80,000	0
Total equity and liabilities	1,710,000	2,430,000

The XBRL document for this report is:

```
<?xml version="1.0" encoding="utf-8"?>
<xbrli:xbrl
```

```
xmlns:xbrli=”http://www.xbrl.org/2003/instance”xmlns:link=”http://www.xbrl.org/2
003/linkbase”
      xmlns:xlink =
“http://www.w3.org/1999/xlink”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xmlns:iso4217=”http://www.xbrl.org/2003/iso4217”
 xmlns:ifrs-gp=http://xbrl.iasb.org/int/fr/
ifrs/gp/2005-05-15”
xsi:schemaLocation=”http://xbrl.iasb.org/int/fr/ifrs/gp/2005-05-15/sample   ifrs-
gp-2005-05-15.xsd”>
  <link:schemaRef xlink:type=”simple”
xlink:href=”ifrs-gp-2005-05-15.xsd”/>
  <xbrli:context id=”DATE_2006”>
     <xbrli:entity>
      <xbrli:identifier scheme=
 “http://www.iasb.org/sample”>Samp
      </xbrli:identifier>
     </xbrli:entity>
     <xbrli:period>
      <xbrli:instant>2006-12-31</xbrli:instant>
     </xbrli:period>
  </xbrli:context>
  <xbrli:context id=”DATE-2007”>
     <xbrli:entity>
      <xbrli:identifier scheme=“http://www.iasb.org/sample”>Samp
      </xbrli:identifier>
     </xbrli:entity>
     <xbrli:period>
      <xbrli:instant>2007-12-31</xbrli:instant>
     </xbrli:period>
  </xbrli:context>
<xbrli:unit id=”U-Monetary”>
     <xbrli:measure>iso4217:EUR</xbrli:measure>
  </xbrli:unit>
<!--Balance Sheet-->
      <!-- For year 2006 -->
  <ifrs-gp:InvestmentProperty     contextRef=”DATE_2006”     unitRef=”U-Monetary”
decimals=”0”> 400,000</ifrs-gp:InvestmentProperty>
      <ifrs-gp:    Inventories    contextRef=”DATE_2006”    unitRef=”U-Monetary”
decimals=”0”> 140,000
</ifrs-gp:Inventories>
      <ifrs-gp: ConstructionInProgress
contextRef=”DATE_2006” unitRef=”U-Monetary” decimals=”0”> 100,000</ifrs-gp:
ConstructionInProgress >
<ifrs-gp: CurrentTaxReceivables
contextRef=”DATE_2006” unitRef=”U-Monetary” decimals=”0”> 350,000</ifrs-gp:
CurrentTaxReceivables>
  <ifrs-gp:Prepayments contextRef=”DATE_2006”
unitRef=”U-Monetary” decimals=”0”> 10,000</ifrs-gp: Prepayments>
<ifrs-gp:CashAndCashEquivalents contextRef=”DATE_2006” unitRef=”U-Monetary”
decimals=”0”> 650,000</ifrs-gp:
CashAndCashEquivalents>
…
<ifrs-gp:EquityTotal contextRef=”DATE_2006”
unitRef=”U-Monetary” decimals=”0”> 1,710,000</ifrs-gp: EquityTotal >
<!-- For year 2007 -->
      <ifrs-gp:InvestmentProperty contextRef=”DATE_2007” unitRef=”U-Monetary”
decimals=”0”> 400,000</ifrs-gp:InvestmentProperty>
      <ifrs-gp:Inventories contextRef=”DATE_2007”
unitRef=”U-Monetary” decimals=”0”> 140,000</ifrs-gp:Inventories>
      <ifrs-gp: ConstructionInProgress
contextRef=”DATE_2007” unitRef=”U-Monetary” decimals=”0”> 100,000</ifrs-gp:
```

```
ConstructionInProgress >
      <ifrs-gp: CurrentTaxReceivables
contextRef=”DATE_2007” unitRef=”U-Monetary”
decimals=”0”> 350,000</ifrs-gp:
CurrentTaxReceivables> …
<ifrs-gp:EquityTotal contextRef=”DATE_2007”
unitRef=”U-Monetary” decimals=”0”> 1,710,000
</ifrs-gp: EquityTotal >
</xbrli:xbrl>
```

The corresponding taxonomy for the XBRL document is an XSD (XML Schema Definition) file representing the XML Schema instance of the vocabulary needed to express assets, equitable and liabilities.

```
<?xml version="1.0" encoding="utf-8"?>
<schema
   xmlns=”http://www.w3.org/2001/XMLSchema”
   xmlns:xbrli=”http://www.xbrl.org/2003/instance”
   xmlns:link=”http://www.xbrl.org/2003/linkbase”
   xmlns:xlink=”http://www.w3.org/1999/xlink”
   xmlns:ifrs-gp=”http://xbrl.iasb.org/int/fr/ifrs/gp/2005-05-15”
   xmlns:sample =”http://xbrl.iasb.org/int/fr/ifrs/gp/2005-05-15/sample”
   targetNamespace=”http://xbrl.iasb.org/int/fr/ifrs/gp/2005-05-15/sample”
   elementFormDefault=”qualified”
   attributeFormDefault=”unqualified”>
     <annotation> <appinfo>
      <link:linkbaseRef xlink:type='simple'
      xlink:href='ifrs-gp-pre-bs-liquidity-2005-05-15.xml'
      xlink:role='http://www.xbrl.org/2003/role/
presentationLinkbaseRef'            xlink:arcrole='http://www.w3.org/1999/xlink/
properties/linkbase'/>
      <link:linkbaseRef xlink:type='simple'
      xlink:href='ifrs-gp-pre-bs-netAssets-2005-05-15.xml'
      xlink:role='http://www.xbrl.org/2003/role/
presentationLinkbaseRef'            xlink:arcrole='http://www.w3.org/1999/xlink/
properties/linkbase'/>
      <link:linkbaseRef xlink:type='simple'
      xlink:href='ifrs-gp-cal-bs-liquidity-2005-05-15.xml'
      xlink:role='http://www.xbrl.org/2003/role/
calculationLinkbaseRef'             xlink:arcrole='http://www.w3.org/1999/xlink/
properties/linkbase'/>
      <link:linkbaseRef xlink:type='simple'
      xlink:href='ifrs-gp-cal-bs-netAssets-2005-05-15.xml'
      xlink:role='http://www.xbrl.org/2003/role/
calculationLinkbaseRef'       xlink:arcrole='http://www.w3.org/1999/xlink/
properties/linkbase'/>
   </appinfo> </annotation>
<import namespace=”http://www.xbrl.org/2003/instance”
schemaLocation=”http://www.xbrl.org/2003/xbrl-instance-2003-12-31.xsd” />
<importnamespace=”http://xbrl.iasb.org/int/fr/ifrs/gp/2005-05-15”
schemaLocation=”ifrs-gp-2005-05-15.xsd” />
      </schema>
```

7. SKOS – Language for Describing the Taxonomy of the Accounts

When we create a metadata for representing financial reports, we need to refer to shared group of accounts, reflecting some patrimonial elements that are common. This can be done by creating a selection of standardized terms with fixed meanings used to help metadata creators to describe a resource. In this way we can avoid ambiguity, control synonyms. This set of standard terms represent controlled vocabulary which can be a simple list of defined terms from which a metadata creator chooses a suitable one or might be a complex thesaurus made up of hierarchical relationships and synonyms. Controlled vocabularies are useful because they help metadata creators

and searchers to use common meanings. To create controlled vocabularies we need a simple language to describe concept and concept schemes. Semantic Web already offers languages like RDS and OWL. The Resource Description Framework (RDF) is a language for making statements about resources but provides only the low level semantics required for metadata statements (Klyne & Carroll, 2004). The OWL language by the other hand, provides the necessary semantic level to describe resources but it demands effort, expertise therefore costs because it is a class-oriented language and requires a precise logically modeling. Therefore there is a need for a language simple enough like RDF that does not required so much effort and expertise, bur power enough to define complex conceptual structures and to support semantically search like OWL does. For this purpose it was created a new language SKOS.

SKOS is an extensible RDF language for describing concept and content of concept schemes (taxonomies, glossaries, classification schemes and thesauri) that include semantic relationships between these concepts. SKOS Core represents the core model for expressing the basic structure and content of a concept scheme (Miles & Brickley, 2005). SKOS Core Vocabulary is a set of RDF properties and RDFS classes that can be used to express the content and structure of a concept scheme as an RDF graph [4]. The SKOS Core Guide and the SKOS Core Vocabulary Specification are currently Working Drafts for W3C Working Group Notes. They present the basic metamodel consisting of an RDF/OWL schema, an explanation of the features that the properties and classes of the schema represent. Currently they are at the proposal stage within W3C.

Taxonomies, or taxonomic schemes, are composed of taxonomic units or kinds of things that are arranged frequently in a hierarchical structure, typically related by subtype-supertype relationships, also called parent-child relationships.

The example presented here is about some concepts from the balance sheet presented in Table 1. The properties used in this example are: skos:broader and skos:narrower used to define relationships of meaning between concepts. Broader is the inverse of narrower.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"

  <skos:Concept rdf:about="http://myTaxonomyScheme/concept/#1">
    <skos:prefLabel>Assets</skos:prefLabel>
    <skos:narrower rdf:resource="
http://myTaxonomyScheme/concept/#2"/>
    <skos:narrower rdf:resource="
http://myTaxonomyScheme/concept/#3"/>
  </skos:Concept>

  <skos:Concept rdf:about="
http://myTaxonomyScheme/concept/#2">
    <skos:prefLabel> Investment property
</skos:prefLabel>
    <skos:broader rdf:resource="
http://myTaxonomyScheme/concept/#1"/>
    <skos:narrower rdf:resource="
http://myTaxonomyScheme/concept/#3"/>
  </skos:Concept>

  <skos:Concept rdf:about="
http://myTaxonomyScheme/concept/#3">
    <skos:prefLabel>Inventories</skos:prefLabel>
    <skos:broader rdf:resource="
 http://myTaxonomyScheme/concept/#1"/>
    <skos:narrower rdf:resource="
 http://myTaxonomyScheme/concept/#4"/>
  </skos:Concept>
   </rdf:RDF>
```

8. Conclusions

Our research is to identify a complex architecture for the complete financial and accounting analysis, using the new information technologies and knoledge representation model. This architecture will be used to develop an intelligent information system, able to realize the accounting analyze and to make financial reports, augmented with semantic.

REFERENCES

Andone, I., Pătruţ, B., *ContTest – a Multi-agent System for accounting education*, SIGEF 2007, Poiana Brasov XIV Congress of Fuzzy Systems in Economy and Management

Klyne, G., Carroll, J. J., *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C Recommendation 10 February 2004. World Wide Web Consortium, 2004.

Miles, A., Brickley, D., *SKOS Core Guide, W3C Working Draft*, February 2005. World Wide Web Consortium, 2005.

Miles, A., Brickley, D., *SKOS Core Vocabulary Specification*, W3C Working Draft, World Wide Web Consortium, 2005.

Sowa, J., *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Course Technology; 1 edition ISBN 978-0534949655

* * * Exsys Ltd., http://www.exsys.com

* * * CLIPS, http://www.ghg.net/clips/CLIPS.html

* * * www.iasb.com – the official site of the IASB organization

* * * www.xbrl.org – the official site of the International XBRL community

Bogdan Pătruţ, PhD in Accounting and Computer Science, is a senior lecturer at the Faculty of Computer Science, ”Alexandru Ioan Cuza” University of Iași, Romania. His domains of interest/research are computer science applied to social and political sciences and multi-agent systems applied to accounting education. He is also interested in social media challenges in the new academic environment. He published or edited more than 25 books on programming, algorithms, artificial intelligence, interactive education, and social media.

Simona-Elena Vârlan is an Assistant Professor at "Alexandru Ioan Cuza" University of Iași, Romania. **Education:** PhD in Cybernetics and Statistics at the Faculty of Economics and Business Administration at “Alexandru Ioan Cuza” University of Iași; BSc. in Computer Science from “Alexandru Ioan Cuza” University of Iași; she has the ability to establish software requirements, to design, implement, and test using different programming languages and paradigms, data bases modelling, knowledge management and representation, to process and to interpret experimental data by statistical methods proved in *2 research & development grants* where she was *IT expert*. Currently, her *subjects of research* are *Recommender Systems, GIS programming, Semantic Web and Internet of Things. Publication activity*: She published over 30 research papers in journals indexed in international data bases from which *10 are indexed WOS*. The international visibility of the scientific work is revealed by 19 citations (GS), h index 3.