Journal of Theoretical and Applied Information Technology
30th November 2021. Vol.99. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

INNOVATIVE INTELLIGENT DATA MODEL FOR AN INTERNATIONAL LANGUAGE MORPHOLOGY SYSTEM

A. LECHHAB, Y. FAKHRI, S. ELHAKMI, N. RAFALIA, S. BENCHEHLA, M. ES-SOULI

RSD, Scientific Research and Doctoral Studies Vienne France

E-mail: s.bourekkadi@gmail.com

ABSTRACT

We offer a unique framework foclassr representing Persian-Arabic morphology in this paper. Because this was too complicated to model exhaustively using traditional methods, it was required to develop a suitable representation formalism. As a result, we created MorphoScript, a declarative object-oriented language that allowed us to best describe the entire morphological knowledge that we could identify. The goal of the research presented here is to develop a data model for natural language morphological components and composition rules. As a result, we shall provide the fundamental elements as well as the theoretical and technical underpinnings of a language capable of replicating and helping morphologies.

Keywords: *Innovation, Natural Language processing, Morphological Knowledge Base, Intelligent Data, Learning, Education System*

1. INTRODUCTION

In the automatic processing of morphology, a major topic is the modelization of morphological knowledge. It is a matter of articulating all morphological knowledge in a formalism that is reasonably well adapted for future utilization of this very specific type of knowledge.

The notion of relational databases has been adopted as the basic data model for storing morphological information in the majority of the work on morphology that we have consulted during years of research in this topic. However, the ensuing research methodologies, which are dependent on the capabilities of the DBMS in use, are insufficiently adapted to this specific challenge.[1][2] It would be condensed into SQL queries to obtain any data that was required. SQL queries don't take into account the type of data they're dealing with, and as a result, they're too broad to produce useful results.[3]

An extra popular option is to employ Artificial Intelligence models to represent morphological components and regulations. Indeed, a morphological analyzer may be thought of as an intelligent expert system that is built on a morphological knowledge base and includes an inference engine with intelligent capabilities for determining the morphological character of the text being analyzed. However, the choice of AI languages for their generality and sequential search for information does not persuade us. Because of the compiler (or interpreter) as well as the language capabilities that are not well suited to this type of data, the Lisp language (Ezzeldin Khaled & al. 2018) or Prolog language (Kadhem S. and Abd Almeer A., 2017) as a medium for representing morphological knowledge is probably not the best choice. For its part, the notion of semantic networks is more significant, but mastering the representation of morphological data requires a complicated and challenging modeling reflection.[4][7]

As a result, we decided on a specific form for the description of the morphological components and rules. The construction of an intelligent data model whose internal structure is natively based on a grammar that allows reasoning in morphology and morphological rules is undoubtedly the solution to this challenge. The compiler, on the other hand, will have to build an appropriate and optimal structure from the morphological data model, allowing direct research and determinism. We are confident that deterministic finite state automata are the best universal structure that meets all of these requirements (M. Gridach and N. Chenfour. 2011b).[11]

We have several approaches to dealing with Persian-Persian-Arabic morphology, each with its own set of

approaches and linguistic depths (Al-Sughaiyer and Al-Kharashi, 2004). The bulk of these systems, however, are based on relational databases that contain all of the Persian-Arabic morphs in tables (Dima Taji et al. 2018). Prefixes, suffixes, stems, and three compatibility tables: prefix-suffix, prefix-stem, and stem-suffix are the foundations of Dima Taji's method. The final three tables specify and ensure the compatibility of the different morphological components.[6][8]

At last, we must point out that behind this outstanding work, there are two big flaws with the selected organization. The first problem is that there are a lot of units that need to be specified one by one in multiple tables. Due to the relational structure of the data and the size of the tables, the second issue, which should be a result of the first, is the time it takes for SQL queries to run.
wordnet is considered as a true reference to *Persian-Arabic* morphology as it is built in *Persian-Arabic* or simply "AWN", basically on complete relational structures The Open Multilingual Wordnet project (), based on the Princeton WordNet or "PWN" development method, includes AWN (Fellbaum Christiane, 2005, Princeton University, 2010). It uses the Suggested Upper Merged Ontology as an Interlingua to connect *Persian-Arabic* WordNet to previously created wordnets. A top-down strategy was used to build AWN (Mohamed Ali Batita and Mounir Zrigui, 2018). It entails translating the Princeton WordNet score (FellbaumChristiane, 2010) and boosting it with more particular Semitic concepts. The development process, which improves wordnet compatibility, is based on semitic cryptography of exact concepts. The core version V1 of beard comprises 21 813 words organized into 9 698 synsets, with six types of synset-to-synset relationships resembling 143 715 connections. The second version, released in 2008, has eleven 269 synsets, which correspond to twenty-three 481 words, and 161 705 linkages, which belong to twenty-two kinds; five of them are meant for PWN - beard interactions (Batita and Zrigui, 2018). As a result, we will only notice the large size of the online database as well as the non- *Persian-Arabic* style (root, models or schemes, etc.).[5][10][15][16]
The Linguistic Data Consortium supports and distributes the Buckwalter Persian-Arabic Morphological Analyzer (Buckwalter T., 2004), which is regarded one of the most referred solutions in the literature (Khaled Shaalan, 2011). (LDC). This is an open pool of universities, libraries, businesses, and government research institutes that was established in 1992 to address the shortage of linguistic resources. The Buckwalter morphological instrument is written in Perl, and the data is divided into three Persian-Arabic English lexicon files: prefixes, suffixes, and stems, totaling approximately 83 000 items. There are three morphological compatibility tables, each with around three 531 entries, that are utilized to handle the many possible combinations. yet again, we'll see how the morphological instrument must cope with a tremendous amount of information, owing to the storage type utilized, which only permitted simple files and a tight relative combination method.[12][13][17][18]

1.1 MorphoScript language

MorphoScript is a programming language based on the Java language syntax, but with a lexico-syntactic extension that allows it to simulate various morphological notions. It allows you to structure a script into classes and morphological rules that may be applied to the various morphological classes (hence the idea of choosing the Java language syntax as a base). An enumeration of what we called morphological components organizes the material of a morphological class. A collection of morphological and/or syntactic characteristics characterizes each component or object. The classes are linked via inheritance and reference links, which are used to link radicals to their original verbs, for example (A. Mahdaouy & al., 2019).[9][14][19][20]

As a result, a morphological script will be divided into three sections: morphological classes, morphological characteristics, and morphological rules. In the sections that follow, these aspects will be examined.

1.2 Morphological Components

When The class is the most fundamental component of a morphological script. Using the class idea, we may organize morphological components (denoted MCM) with the same fundamental features into a single building unit called morphological class (denoted MCL). Verbs, nouns, and particles must be grouped together into classes with similar morphological features and semantic activities.[21][22]
For instance, a category of all prefixes that may be added to a specific type of verb. As a result, combining verbs and nouns in the same class, or verbs with different morphological structures, makes no sense. Another clever aspect of MorphoScript is that it is not required to specify all potential physical

components inside a morphological class, simply the various schemes. The class of healthy verbs, for example, will allow all conceivable schemes of healthy verbs to be defined, rather than healthy verbs themselves. This results in a significant decrease in the number of morphological entries.[23][24][25] MorphoScript is a scripting and object-oriented language in one. The attributes of a MorphoScript class, unlike those of other object languages, are themselves objects of the class (such as enumerations in Java). As a result, instantiation is unnecessary, and the class serves a dual purpose: it defines the class and declares and characterizes the class's objects. We think of the class as a collection of things rather than a factory of objects. [26][27]

2. ALL CLASS OF SYSTEM

2.1 Morphological Properties

To morphologically express the exclusive morphological components, we wish to provide a list of morphological homes (which may also be syntactic characteristics), then assign them to the specific participants of the distinct classes. MCMs of diverse classes are commonly described using common features.[28][29]

A property type is described the usage of the equal syntax as a morphological type to which the «@property» annotation will be applied, as in the following example:

```
@property
class Number {
        SINGLE
DUAL
PLURAL
```

The morphological descriptor, like a morphological component, can include morphological parameters, most notably its graphemic value, as demonstrated in the example below:

```
@property
class Number {
        SINGLE("the translation of the word into the language studied")
DUAL("the translation of the word into the language studied")
PLURAL("the translation of the word into the language studied")
```

2.2 Properties

A list of morphological descriptors can be used to specify the components of a morphological class (the different schemes inside). In general, each MCM may have its own set of morphological descriptors. If we look at the "Gender" attribute, for example, we can see that some of the class's components are male and others are female. Component properties >> is how we refer to this type of property. We'll use the @uses >> annotation to enable a morphological class the ability to use the morphological descriptors of a property class to specify its components.

```
@uses(Number)
class ClassName {
        M1("name1", SINGLE)
        M2("name2", PLURAL)
        M3("name3", PLURAL)
        ...
}
```

The many morphological descriptors introduced by the @uses >> annotation become extremely accessible and can be utilized by all of the class's morphological components :

```
@property
class P1 {
        V1
V2
V3
@property
class P2 {
        V4
V5
}
@uses(P1, P2)
class C1 {
        M1("value1", V1)
        M2("value2", V2, V5)
        M3("value3", V3, V4)
}
```

When we utilize the morphological descriptors with their fully qualified names, the annotation @uses >> becomes unnecessary. Without the annotation @uses (P1, P2) >>, the class C1 >> in the previous example may be created as follows:

```
class C1 {
        M1("value", P1.V1)
        M2("value", P1.V2, P2.V5)
        M3("value", P1.V3, P2.V4)
}
```

there is another use case for properties is that some morphological classes must be globally defined by a collection of morphological descriptors that will be applied to all morphological components.For

Journal of Theoretical and Applied Information Technology
30th November 2021. Vol.99. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

instance, a morphological class's components are all single names. This sort of property is known as <<class properties>>, and we use the annotation <<@is >> to describe it:

@is(PropertyName**)**
class ClassName **{**

}

For Example :

@is(P1.V1, P2**)**
class C2 **{**
M1
M2
}

the morphological components M1 and M2 of the class C2 are both defined by the morphological descriptor V1 of the property P1 and all of the morphological descriptors of the property P2. This last feature requires that P2's morphological descriptors are additive rather than contradictory (like male and female).

It's worth noting that the same class can use both class and component properties at the same time. As a result, the general syntax is:

@is(P1, P2, ...)

@uses(Q1, Q2, ...)

class ClassName **{**
}

The Pi are morphological descriptors or property classes, but the Qi are always property classes**.**

Reference properties

Semantic linkages can exist between MCMs belonging to two different instructions. As a result, some morphological features stated in a classification are conjugate kinds of several genuine MCMs reported in distinct classes, Because the latter is described in a category and the other three characteristics in a distinct class, it is necessary to factor in this hyperlink reference between the components. In this example, we utilize the <<@ref >> annotation, which defines the reference class:

@ref(OriginClass)

class DerivedClass {...}

Any morphological component << Mi >> of the derived class, on the other hand, must be referenced to the base component << Rj >> using the reference property << Mi >>, which corresponds to the identification of the origin (or base) component Rj.

class Base {
...
Rj ("value", ...)
}

@ref(Base)

class Derived {
Mi("value", Rj, ...)
}

This signifies that the morphological component Mi has a base reference declared in the « Base » class with the identifier « Rj ».

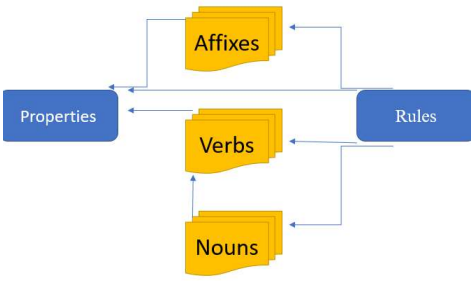
2.3 Extension

inheritance is a critical concept that allows for code optimization and reuses in the object-oriented programming domain, as well as the construction of a coherent set of classes arranged into a large tree structure known as an inheritance hierarchy. Because Persian-Arabic morphology may be represented as an extraordinarily complicated inheritance graph, the MorphoScript language needs to accommodate this concept. [30][31][32]

The inheritance connection, like the other aforementioned concepts, can be defined by an annotation for the sake of syntactic uniformity. As a result, we can use the following comparable syntax:

@extends(SuperClass)
class DerivedClass **{**
...
}

2.4 Properties



With the annotation << @is >>, we've previously introduced the concept of class properties and how they're implemented. This notion introduces a new sort of inheritance that allows a morphological class to be linked to a set of properties. In reality, numerous classes can share a single morphological

trait, allowing them to be described collectively using that property. [33][34]

If we consider a set of morphological classes "MCL1, MCL2,... MCLk," all of which have the attribute MPC, the classes MCLi are defined as follows:

@is(MPC)

class **MCL**$_i$ { $\forall i = 1 \grave{a} k$

...

}

2.5 Exception

The @exception annotation its possible to be used directly to morphological components or partitions of a morphological class .[35][36]

As seen in the example below, the morphological component MCM2 as well as all of the morphological components of the partition PID1 are exceptions to the morphological rules applied to the class MCL.

```
class E1 {
        Alias1(CME1, CME2, ...)
        Alias2(CME1, CME2, ...)
        ...
}
```

So you can see a specific classes that include subsets of irregular components of a morphological class MCL that do not follow the class's concatenation criteria. These irregular components will be given an alias so that they may be modified globally in any rule.

```
class MCL {
        MCM1(...)
        @exception MCM2(...)
        MCM3(...)
        ...
        @exception
class PID1 {
        ...
        MCMi(...)
        ...
}
class PID2 {
        ...
MCMj(...)
        ...
}
...
}
```

By using the @filter annotation, you may define a new filter class that gathers the only morphological components of a class MCL that accept the application of morphological rules.

```
@filter(MCL)
class F1 {
        Alias1(CME1, CME2, ... )
        Alias2(CME1, CME2, ...)
...
}
```

The **@filter** annotation can be used at the morphological component level or at the partition level, just like the exceptions.

1) 7.5 Object class

3. MODULES

A module will be implicitly derived based on the location of the morphological class, which corresponds to a directory like packages in python language. However, the @module annotation can be used to explicitly indicate a module name:

@module(technical.name.of.the.module, "Logical Name ")

The following annotation can be used to call modules:

@import(moduleA, moduleB, ...)

Above classes, modularity adds a greater level of grouping. Actually, the morphological database will be arranged into high-level objects called modules for the sake of structure. Each module comprises a collection of morphologically consistent classes. We might divide our database into six independent modules using this last MorphoScript technique, as indicated in the diagram below:[38][39]

It is crucial to have a strong understanding of morphological classes when creating a database using the MorphoScript language, which can be thought of as a conceptual notation of morphology. The language will thereafter be able to represent the morphological conceptual model symbolically. The following essential restrictions must be considered in such a model:

- When adhering to the first and second constraints is challenging, a class can be broken down into a set of subclasses using the aggregation idea. On the one hand, all of the class's components have some

common morphological features, and on the other hand, each subclass collects elements with specific concatenation properties.
- Another key limitation is that any inheritance link must be reported in order for the concatenation rules to take it into consideration.

- Morphological classes must be built in such a way that components of the same class accept all of the same prefixes and suffixes, or they are suffixes or prefixes of the same classes. The building rules would otherwise be defined for each morphological component rather than for the entire class.
- A special example of the previous requirement is that components belonging to the same morphological class must either accept or reject concatenation.[40][41][42]

4. RESULTS

We were able to describe the entire morphological knowledge base in a legible and incredibly efficient data format using the MorphoScript language. This result is shown in Table through the evaluation of the numerous components and morphological principles.

There is a significant improvement in modeling when comparing MorphoScript data model to our old (

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="morpho.verbs.origin" >
<morphological_class name="OriginScheme">
   <properties>
      <modifier>final</modifier>
      <is> ---- </is>
      <is> ---- </is>
      <is> ---- </is>
      <is> ---- </is>
   </properties>
   <component name="XXX0" id="Y1" />
   <component name="XXX1" id="Y2" />
   <component name="XXX2" id="Y3" />
   <component name="XXX3" id="Y4" />
   <component name="XXX4" id="Y5" />
   <component name="XXX5" id="Y6" />
   <component name="XXX6" id="Y7" />
   <component name="XXX7" id="Y8" />
   ...
</morphological_class>
</package>
```

, this is just an example (for more information on the system structure, we instruct the researchers to contact the authors for a detail on the deaffferents system class, and the rest of the code)

)XML solution XMODEL .

B. Modules	*C. MCL*	MPC	MCM
Suffixes	10	5	40
Verbs	70	5	300
Particles	10	5	30
Prefixes	10	3	30
Nouns	10	11	250

We are confident that we have defined a very acceptable language for modeling Persian-Arabic morphology utilizing a set of key features: script-oriented or declarative syntax, object-oriented modeling, pattern-based morphology, and extendable semantics using the annotations idea as a result of this study. This enabled us to master morphological complexity, model it completely concisely, and decouple it from the morphological controller. With a huge reduction in complexity, the latter becomes quite optimal.
Finally, the morphological treatment becomes straightforward and deterministic thanks to the morphological automata generated automatically from the MorphoScript database.

5. CONCLUSION AND FUTURE WORK

in the future work, we will try to integrate this system at the level of learning and education, at the base of a data warehouse dedicated to the guidance of the pupils to understand these international languages, our system is based on a set of innovative research in this field, we cannot confirm that these are the first, but it is a continuation of other research work.
using Big data in the future work, can give good results, while our future research work will be based on data sciences.

REFERENCES:

[1] Arabic Finite-State Morphological Analysis and Generation. Proceedings Retrieved from http://www.ikprress.org/index.php/JOBARI/article/view/3306

[2] Darwish K., 2002. Building a Shallow Morphological Analyzer in One Day. Proceedings of the workshop on Computational Approaches to Semitic Languages in the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02). Philadelphia, PA, USA.

[3] Dima Taji, Salam Khalifa, Ossama Obeid, Fadhl Eryani, and Nizar Habash, 2018. An Arabic Morphological Analyzer and Generator with

Copious Features. Proceedings of the 15th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 140–150 Brussels, Belgium, October 31, 2018. https://doi.org/10.18653/v1/P17

[4] Ezzeldin Khaled, Siddiqui Sanjeera and Shaalan Khaled, 2018. Evaluating Arabic Parser and Recommending Improvements. pp. 417-428. Proceedings of the International Conference on Advanced Intelligent Systems and Informatics. DOI : 10.1007/978-3-319-64861-3_39.

[5] Fellbaum Christiane, 2005. WordNet and wordnets. In: Brown, Keith et al. (eds.), Encyclopedia of Language and Linguistics, Second Edition, Oxford: Elsevier, 665-670.

[6] Fellbaum Christiane, 2010. WordNet. In Theory and applications of ontology: computer applications, pages 231–243. Springer.

[7] Kadhem Suhad and Abd Almeer A., 2017. Arabic Texts Classification Based on Keywords Extraction Technique. Engineering and Technology Journal (مجلة الهندسة والتكنولوجيا). ISSN: 16816900 24120758. Volume: 35 Issue: 2 Part (B) Scientific Pages: 96-104. Publisher: University of Technology (الجامعة التكنولوجية).

[8] Khaled Shaalan, Younes Samih, Mohammed Attia, Pavel Pecina, and Josef van Genabith, 2011. Improved spelling error detection and correction for arabic. Council for Science Engineering and Technology, page 7.

[9] Mahdaouy Abdelkader, Ouatik El Alaoui Said and Gaussier Eric, 2019. Word-embedding-based pseudo-relevance feedback for Arabic information retrieval. Journal of Information Science (JIS). Vol. 45. pp. 429–442. DOI : https://doi.org/10.1177/0165551518792210.

[10] Mohamed Ali Batita and Mounir Zrigui, 2018. Derivational Relations in Arabic WordNet. Proceedings of the 9th Global WordNet Conference (GWC 2018). January 8 - 12, pp 137–145, 2018 Singapore. ISBN 978-981-11-7087-4

[11] Mourad Gridach and Noureddine Chenfour. (2011c). A Computational Feature-Based Morphological Analysis and Generation of Modern Standard Arabic. International Journal of Computational Linguistics Research (IJCLR), ISSN 0976-4178, Volume (2), Issue (1), pp. 24 – 36, March 2011.

[12] M. Ehrig and S. Staab, "QOM - Quick Ontology Mapping", In Proceedings of International Semantic Web Conference, 2004, pp.683-697.

[13] J. Euzenat and P. Shvaiko, "Ontology Matching", Springer-Verlag, Heidelberg (DE), 2007.

[14] J. Euzenat and P. Valtchev, “Similarity-Based Ontology Alignment in OWL-Lite", In Proceedings of ECAI, 2004, pp.333-337.

[15] D. Fensel, F.V. Harmelen, I. Horrocks, D.L. McGuinness, and P.F. Patel-Schneider, "OIL: An Ontology Infrastructure for the Semantic Web", Presented at IEEE Intelligent Systems, 2001, pp.38- 45.

[16] D. Fensel, "Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce", Springer, 2001.

[17] V. Haarslev and R. Moller, "RACER System Description", In Proceedings of IJCAR, 2001, pp.701-706.

[18] F. van Harmelen, P.F. Patel-Schneider, and I. Horrocks (Editors), "Reference Description of the DAML+OIL Ontology Markup Language", http://www.daml.org/2000/12/reference.html, 2000.

[19] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "Semantic Schema Matching", In Proceedings of OTM Conferences (1), 2005, pp.347-365.

[20] C. Ghidini and F. Giunchiglia, "A Semantics for Abstraction", In Proceedings of ECAI, 2004, pp.343-347.

[21] O. Gotoh, "An Improved Algorithm for Matching Biological Sequences", Presented at Journal of Molecular Biology, 162:705-708, 1982.

[22] Y. Kalfoglou and W.M. Schorlemmer, "IF-Map: An Ontology-Mapping Method Based on Information-Flow Theory", Presented at Journal Data Semantics, 2003, pp.98-127.

[23] M.C.A. Klein and D. Fensel, "Ontology Versioning on the Semantic Web", In Proceedings of SWWS, 2001, pp.75-91.

[24] P. Lambrix and H. Tan, "A Tool for Evaluating Ontology Alignment Strategies", Presented at Journal Data Semantics, 2007, pp.182-202.

[25] M. Li, M. Baker, "The Grid: Core Technologies", John Willey & Sons England (2005).

[26] G.A. Miller, “WordNet: A Lexical Database for English", presented at Commun. ACM, 1995, pp.39-41.

[27] H. Mihoubi, A. Simonet, and M. Simonet, "An Ontology Driven Approach to Ontology Translation",

[28] A. Borgida, R. Brachman, D. McGuinness, and L. Resnick.

[29] CLASSIC: Astructural data model for objects. In Proceedings SIGMOD-89, 59-67. ACM, 1989.

[30] P. Courtot. A New Personal and Enterprise Application: SEARCH '97 White Paper, In Verity web site http : //www. verity, com /eorp /whi~epapers /".

[31] Fensel: Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. SpringerVerlag, Berlin, D, to appear (2000).

[32] P. Borst, J.M. Akkermans, and J.L. Top: Engineering Ontologies, International Journal of HumanComputer Studies 46 (1997) 365-406.

[33] A.Th. Schreiber, J.M. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van De Velde, and B. Wielinga: Knowledge Engineering and Management. The MIT Press, Cambridge, MA, 2000.

[34] U. Reimer (Ed.): Proc. 2nd Int. Conf. on Practical Aspects of Knowledge Management (PAKM'98), Basel, Switzerland, October 1998. URL: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-13/.

[35] D. Fensel, S. Decker, M. Erdmann, H.-P. Schnurr, R. Studer, and A. Witt: Lessons learnt from Applying AI to the Web. Journal of Cooperative Information Systems, to appear (2000).

[36] F. van Harmelen and J. van der Meer: WebMaster: Knowledge-based Verification of Web-pages. In Proceedings 2nd Int. Conf. on The Practical Applications of Knowledge Management (PAKeM99, London, UK, April 1999), pp. 147-166. The Practical Applications Company, Blackpool, UK, 1999.

[37] J. Davies, S. Stewart, and R. Weeks: Knowledge Sharing over the World Wide Web, WebNet '98, Florida,USA, November 1998. (also at http://www.bt.com/innovation/exhibition/knowledge_management/).

[38] J. Davies: Supporting Virtual Communities of Practice, in R. Roy (Ed.): Industrial Knowledge Management, Springer-Verlag, London, forthcoming (2000).

[39] B. Bremdal, F. Johansen, C. Spaggiari, R. Engels, R. Jones: Creating a Learning Organisation Through Content-Based Document Management, OECD HRP Meeting, Loen, NO. CognIT Report, Oslo, May 1999.

[40] A. Maedche, H.-P.Schnurr, S. Staab, and R. Studer: Representation Language-Neutral Modeling of Ontologies. In: Frank (Ed.), Proceedings German Workshop Modellierung 2000. Koblenz, D, April 2000.

[41] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S.Staab, and R. Studer: The Ontology Inference Layer OIL, On-To-Knowledge EU-IST-10132 Project Deliverable No. OTK-D1, Free University Amsterdam, Division of Mathematics and Informatics, Amsterdam, NL, 2000. Available from http://www.ontoknowledge.org/oil.

[42] F. Ygge and J.M. Akkermans: Decentralized Markets versus Central Control - A Comparative Study, Journal of Artificial Intelligence Research 11 (1999) 301-333. (Also available from http://www.jair.org/).

[43] J. Angele, D. Fensel, and R. Studer: Developing Knowledge-Based Systems with MIKE, Journal of Automated Software Engineering 5 (1998) 389-418.

[44] Y. Ding, G. Chowdhury, and S. Foo: Bibliometric Information Retrieval System (BIRS): A Web search interface utilizing bibliometric research results. Journal Am. Soc. for Information Science, to appear (2000).

[45] Bourekkadi, S., Khoulji, S., Mabrouk, A., Larbi, K. M., Laaziri, M., & Omari, O. (2016). Psychologie Informatique et son impact sur le comportement humain [Computers psychology and its impact on human behavior]. *International Journal of Innovation and Applied Studies*, *14*(2), 543.

[46] El Imrani, O., Layti, M. B. M., Bourekkadi, S., Boulaksili, A., & Kabbassi, I. (2021). Optimization of the International Trade Activities in the Period of COVID-19 by Proposing an Algorithm. In *Artificial Intelligence for COVID-19* (pp. 187-193). Springer, Cham.