# The Universal Author Identifier System (UAI_Sys)*

Dimitris A. Dervos[1], Nikolaos Samaras[2], Georgios Evangelidis[2], Jaakko P. Hyvärinen [3], Ypatios Asmanidis[1]

[1] Information Technology Dept., ATEI, P.O. BOX 141, 57400 Sindos, Greece, Tel: +30.2310791295, Email: {*dad*,*ypasm*}*@it.teithe.gr*

[2] Dept. Appl. Informatics, Un. of Macedonia, P.O. BOX 1591, 54006 Thessaloniki, Greece, Tel:+30.2310891844, Email: {*samaras*,*gevan*}*@uom.gr*

[3] Dept. of Computer Science and Information Systems, Un. of Jyväskylä, Finland, Tel: +358.142601211, Email: *japahyva@cc.jyu.fi*

**Abstract**

One common problem in the scientific research literature is that each one author cannot easily be identified uniquely. The problem arises when there are authors with identical names, authors who have changed their name(s) in the course of time, and authors whose names appear in alternative versions (for example: Jaakko Hyvärinen, and J. P. Hyvärinen) across the publications they have (co-) authored. The issue becomes more of a problem when data analysis utilizing author names is to be conducted, for example: in citation analysis.

In this paper we introduce the Universal Author Identifier system, codenamed UAI_Sys. The system is web based and publicly available, enabling each one author to register/update his/her own metadata, plus acquire a unique identifier (UAI code), ensuring name disambiguation. As soon as UAI_Sys becomes accepted and enjoys worldwide use, selected author metadata will become globally available to all interested parties. Care is taken so that UAI_Sys comprises more than just a database for storing and handling author identifiers. Provision is taken for the system to incorporate web services facilitating communication with third party applications, thus expanding the possibilities for web based co-functionality. Last but not least, the system supports role-based access and management (i.e. different user roles for authors, librarians, publishers, and administrators) for efficient and effective information dissemination and management, promoting research and collaboration.

UAI_Sys is being designed/developed along the lines of the Cascading Citations Analysis Project (C-CAP) which is co-funded by the Alexander Technology Educational Institute (ATEI), and the University of Macedonia (UoM).

* *Research conducted along the lines of the Cascading Citation Analysis Project (C-CAP, http://www.ccapnet.org), funded by the Research Committees of ATEI, and the University of Macedonia, Thessaloniki, Greece.*

## 1. Introduction

Today, developments like the evolving scholarly communication environment, the open access movement, and the globalization in academia and research advance with a rapid pace. As a result, more intense becomes the need for an improved scheme that quantifies the contribution research publications, authors, and scientific collection make in promoting science and technology. The current practice considers the number of citations received by each one research publication, and utilizes this information in the calculation of the journal impact factor metric [8, 9].

In the *Cascading Citations Analysis Project* (C-CAP), a somewhat different approach is taken: instead of refining the analysis at high level, an attempt is made to increase the granularity of the citation indexing paradigm at the data preparation stage, so that the information extraction phase that follows targets a richer data corpus. In this respect, citations are considered to target (*article*, *author*), rather than just *article* entities, plus it is not only the direct citations received that account for the calculation of the target's popularity measure, but also the citations received indirectly, by considering a finite number of levels in the corresponding citation graph [3,4]. In this respect, each one (co-)author of a research publication need be uniquely identifiable.

Research article authors are usually rated in accordance with the number of citations received by the articles they have (co-) authored, as well as on the basis of the citing article's host publication impact factor (journal, conference proceedings, book, etc.). In this respect, authors need be uniquely identified not just for the purpose of determining self-citations in the citation graph. Author popularity ranks calculated this way are then taken to comprise a critical parameter when it comes to making decisions on tenure, promotion, funding, and so on [26]. This is common practice, despite all the warnings issued with regard to the disadvantages of relying upon impact factor alone for journal evaluation [10].

Attempts have been made to uniquely identify each one author in citation databases, restricted in their scope to the citation dataset in question [11,28]. In direct relation is the fact that even the best automatic author name disambiguation system is bound to not be 100% foolproof [12], in any case: it is bound to fail when it comes to having to differentiate between homonyms [1]. In this respect, in C-CAP a strategic decision has been taken to face the challenge of developing a citation dataset supplier neutral Universal Author Identifier System (UAI_Sys).

## 2. System Overview

UAI_Sys is a Java based web application allowing each one author to register/update his/her own metadata content and request a unique identifier that s/he is going to retain and make use of for life. Apart from obtaining his/her unique author identifier (codenamed: UAI code), the author specifies the subset of his/her personal (meta)data that s/he wishes to become globally available to all interested parties. The system supports the industrial standard interface for other applications to connect to and co-function with, over the Internet.

## 2.1 Functional Requirements

When an author registers him/herself with UAI_sys, the system utilizes a timestamp-based random number generator facility to create an all-numeric, sixteen digit string, the UAI code, that will uniquely identify the author in question. The latter uses his/her UAI code as a username to login to UAI_Sys.

To prevent misuse in the form of numerous UAI registration requests originating from a single source fraudulent application, UAI_Sys comes bundled with a watermark protection facility. During the registration procedure, an image is generated involving a random mix of numeric and alphabetic characters. The user is required to type in the characters s/he is presented with for the system to proceed with the new author registration process. Upon completion of the latter, an email message is automatically compiled and sent to the just registered author, including his/her unique UAI code, plus a password required for accessing UAI_Sys. During the new author registration process, the user is prompted to also enter a 'secret phrase', plus his/her private response to it. The scheme comprises an alternative way of logging on to UAI_Sys in case the user forgets his/her password in the future. The password as well as the 'secret phrase'/response combination are user maintained and updatable entries during the regular UAI_Sys logon session(s).

Once registered with UAI_Sys, each one author is able to enter/update his/her own metadata. Every instance of the latter is updatable, except from the UAI string, of course. Trivial cases of author metadata that may be updated comprise, for example, the 'postal address', 'email address' fields. More involved cases involve, for example, the updating of the author's last name, or the insertion of author name aliases, i.e. different versions of the author's (*name*, *middle name(s), surname*) combination, all referring to the same individual.

UAI_Sys provides support for three types of user roles: a) the *administrator* who has full access/control over the system, b) the *operator* who can register new authors in cases where the latter either cannot access the Internet, or choose to have another authority (the library, for example) to act on their behalf, and c) the individual *author* who has access to and feels comfortable with the technology involved, utilizing it in order to keep his/her UAI_Sys entry up-to-date.

Libraries are expected to play a key role in UAI_Sys, for one reason: the system enjoying world-wide applicability, problematic cases calling for person to person communication between the central UAI_Sys management team and the end users are bound to arise. Such problems may only be dealt with by implementing decentralization, in the form of the local libraries acting as authorized UAI_Sys agents. This way, authors who seek for assistance in using/accessing the system will find a helping hand in their own language. For example, one may consider the most unlikely (however: possible) case whereby a UAI_Sys user has lost/forgotten his/her login password, and does not remember the response registered to comprise a valid one to the secret phrase associated with his/her UAI_Sys account. Also, it so happens that the email address registered with the UAI_Sys account is no longer valid, meaning that it is meaningless for the author in question to tag the '*I have forgotten my password*' radio button in order to have the system assign a new (automatically generated password) that is subsequently sent to the (obsolete) email address already registered with the author's UAI_Sys entry. The situation

calls for a person-to-person communication session, whereby the author will supply the necessary evidence that s/he is indeed the individual claimed to be. The communication will most likely be carried out in the author's own language, at the local library. The latter, provided that they act as an authorized UAI_Sys agent, will then make use of their privileged access to the system, initiate the procedure that generates a new password for the user in question, plus update the corresponding UAI_sys entry with the author's new email address where the just assigned (new) password is (automatically) emailed to.

It is important that UAI_Sys maintains a complete/detailed log of all update operations, with sufficient data to trace application critical moments whereby a UAI_Sys account updates the corresponding own data content, or that of another account's (say, in the case of privileged transactions initiated by accounts operated by libraries authorized to act as UAI_Sys agents to the application).

Last but not least, UAI_Sys need be searchable, both by the public user as well as by the registered one. The metadata fields and their content that are accessed by the former next to the latter may differ, at each one author's own discretion. In all cases, UAI_Sys is to associate every individual author with links to the corresponding own works that have been published electronically, available from dispersed resources across the Internet. For the latter to become possible, UAI_Sys needs to be coupled to the corresponding e-journals, institutional repositories [2,7], etc., and the author-user to be authorised to access their (full-text) content.

### 2.2 Non Functional Requirements

In parallel to supporting the functionality outlined in Section 2.1 above, the UAI_Sys application needs to also opt for and support/implement a number of (critical) non functional requirements, for example:

- Be secure in user authentication, plus in implementing user authorization policies.
- Be flexible and easy to upgrade, extend and maintain.
- Be durable, ensuring the integrity and the restoration of its content over soft- and hard- system crashes.

## 3. Pilot Implementation: Technology and Tools

As it is mentioned in Section 2 above, the UAI_Sys pilot implementation is a Java based web application that runs on top of an application server. The Java platform has been chosen for system implementation since it comprises the de facto world-wide standard for developing open source web-based applications, utilizing a large number of available tools and technologies. UAI_Sys is a Java2 Enterprise Edition (J2EE) application that utilizes open source Java tools and technologies provided by the JBoss community [19].

The application runs on the top of a JBoss 4.0.4 application server [17]. The latter is coupled to the PostgreSQL 8.1.2 object-relational database management system [27]. The UAI_Sys application has been developed using the JBoss SEAM [18], a

new application development framework for the Java Enterprise Edition 5 (Java EE 5) Platform, unifying the component models of Java Server Faces (JSF), and Enterprise Java Beans 3.0 (EJB 3.0) [22]. JSF comprises a User Interface (UI) framework for Java web applications [24], and EJB 3.0 is an extension to the Enterprise Java Beans that brings simplification and new functionality to the earlier EJB Application Programming Interfaces (APIs) [6].

The first version of the pilot implementation allows the client application to test-drive each one component of the proposed solution to ensure that the latter fulfils the set requirements specification. In the course of the pilot implementation phase new requirements emerge which are subsequently incorporated into the model under development. Figure 1 presents the UML diagram [30] of the data model for the pilot application, utilizing EJB 3.0 entity beans as persistent and plain old java objects (POJOs). JBoss implements EJB 3.0 persistency by means of the Hibernate 3 persistence engine [13]. It is noted that methods and attributes are not represented in Figure 1. The latter presents only classes corresponding to EJB 3.0 entity beans.

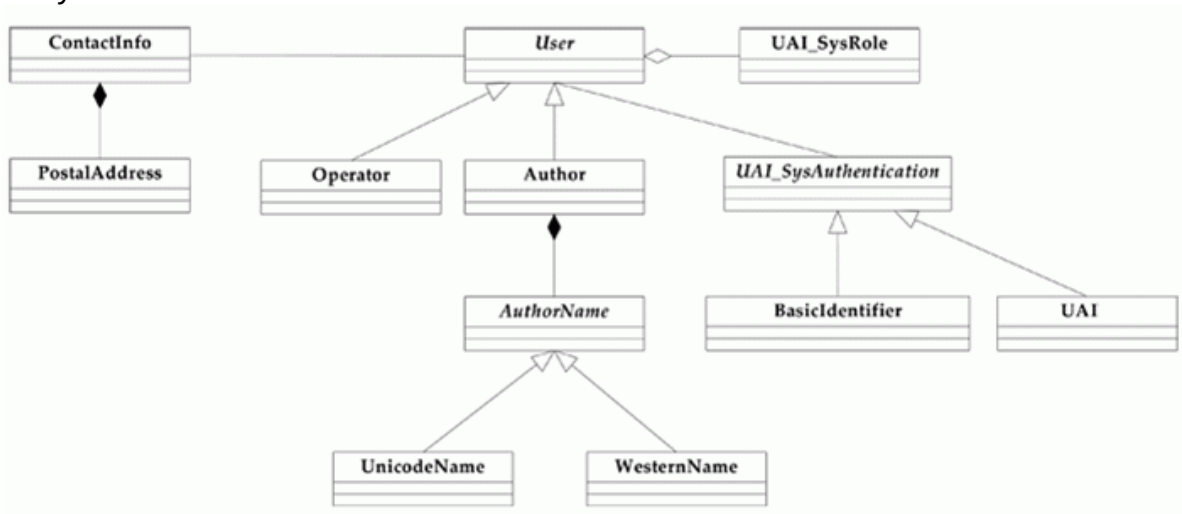


**Figure 1:** The pilot UAI_Sys data model

The Eclipse 3.1 Integrated Development Environment (IDE) [5] is used for application source code generation, utilizing the JBoss IDE plug-in [20]. Unit tests are created and run by using the JUnit [25] unit testing framework. To implement the three types of user roles (*administrator*, *operator*, and *author*), the Java Authentication and Authorization Service (JAAS) is used, namely a set of APIs that enable services to authenticate and enforce user access control [14]. Sensitive information like user passwords are channeled through an SSL tunnel, ensuring the safety of transactions during system operation over the Internet. Application packaging and deployment are done with the Apache Ant build tool [29].

### 3.1 Web Service Support

The World Wide Web Consortium (W3C, [33]) who manage the evolution of the SOAP protocol [31] and the Web Service Description Language (WSDL) specifications [32], define the concept of the Web service as follows:

> *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standard.*

The UAI_Sys application implements support for Web service interface(s), via the JBossWS web service implementation [21]. The latter comprises a standard implementation of J2EE compliant web services (WS4EE, [16]), also supporting Web Service Metadata (JSR-182, [15]) and EJB3 Stateless Session endpoints [23]. This leads to a more comprehensible and time saving development process than, say, XML descriptors based web service coding (WS4EE). In the case of a class needing to provide a web service interface, all which is required is for the @WebService and @WebMethod annotations to be included and the Web service is generated automatically at application deployment time.

### 3.2 Lessons Learned and Experience Gained

To meet the UAI_Sys requirements, the core of the pilot application was developed by utilizing the J2EE and JAVA EE 5 technologies. In order to simplify system development, and facilitate (stepwise) test driven application code generation/writing, EJB 3.0 has been found to comprise a successful strategic decision choice: the focus was on writing POJOs using annotations, rather than on coding complex EJB APIs. The JBoss SEAM platform has also been found to comprise a winner; nowadays it is hard for one to consider developing Java web applications without it. Although the UAI_Sys application does not involve complex workflows or user interaction sessions, both comprising cases where SEAM demonstrates its strength, it has been still possible to obtain the feeling of SEAM's eliminating the need for normal JSF-relating glue code, as well as of its concept of bijection [18].

## 4. Conclusion

In this paper we report on the pilot version of the Universal Author Identifier system, codenamed UAI_Sys. The system is web based and is meant to be publicly available, enabling each one author to register/update his/her own metadata, plus acquire a unique identifier (UAI code), ensuring name disambiguation. As soon as UAI_Sys becomes accepted and enjoys worldwide use, selected author metadata will become globally available to all interested parties. Care is taken so that UAI_Sys comprises more than just a database for storing and handling author identifiers. Provision is taken for the system to incorporate web services in order to provide communication facilities to third party applications expanding the

possibilities for web based co-functionality. Beginning with the pilot version of UAI_Sys, the system supports role-based access and management (i.e. different roles for authors, librarians, publishers, and administrators) in a way that it facilitates efficient and effective information dissemination and management, promoting research and collaboration.

## Acknowledgements

The authors are grateful to Richard Hartley and Anita Coleman, members of the C-CAP Extended Advisory Board Committee, for their generous assistance and active participation in UAI_Sys relating discussions, carried out either during live sessions as well as over the Internet. Special thanks are due to ISI-Thomson Scientific (http://www.isinet.com/) for making their citation database available to C-CAP.

## References

1. Braun, T. (2003). The reliability of total citation rankings. *J. Chem. Inf. Comput. Sci* (43), p.45-46.
2. CDSware (2006). Retrieved 15.05.2006: http://cdsware.cern.ch/
3. Dervos, D.A. and Kalkanis, T. (2005). cc-IFF: A Cascading Citations Impact Factor Framework for the Automatic Ranking of Research Publications. Proceedings of the 3rd IEEE International Workshop on Intelligent Data Acquisition and Advanced Computer Systems: Technology and Applications (IDAACS), p. 668-673, Sofia, Bulgaria, 5-7 September, 2005. Postprint version from DLIST, retrieved 15.05.2006: http://dlist.sir.arizona.edu/1105/
4. Dervos, D.A., Samaras N., Evangelidis G., and Folias T. (2006). A New Framework for the Citation Indexing Paradigm. Proceedings of the Annual Meeting of the American Society for Information Science and Technology (ASIS&T), Austin, Texas, November 2006: to appear
5. Eclipse (2006): Eclipse Integrated Developing Environment. Retrieved 5.05.2006: http://www.eclipse.org/
6. EJB 3.0 Expert Group (2006): JSR 220: Enterprise JavaBeansTM Version 3.0. Retrieved 15.05.2006: http://jcp.org/aboutJava/communityprocess/pfd/jsr220/index.htm
7. Fedora (2006). Retrieved 15.05.2006: http://www.fedora.info/
8. Garfield, E. and Sher, I.H. (1963). New factors in the evaluation of scientific literature through citation indexing. *American Documentation* 14(3): 195-201.
9. Garfield, E. (1972). Citation Analysis as a tool in journal evaluation. *Science* 178: 471-479.
10. Garfield E., (1994). The Impact Factor. Retrieved 15.05.2006: http://scientofic.thomson.com/knowtrend/essays/journalcitationreports/impactfactor/
11. Giles C.L., Bollacker K., Lawrence S. (1998). CiteSeer: An Automatic Citation Indexing System, *Digital Libraries 98-The Third ACM Conference on Digital Libraries Proceedings*, p. 89-98
12. Han, H. Giles, L. Zha, H. Li, C. and Tsioutsiouliklis K. (2004). Two supervised learning approaches for name disambiguation in author citations, Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries, p. 296-305, 2004.

13. Hibernate (2006): Hibernate. Retrieved 15.05.2006: http://www.hibernate.org/
14. Java Authentication and Authorization Service, JAAS (2006). Retrieved 15.05.2006: http://java.sun.com/products/jaas/
15. Java Community Process (2006): JSR 181: Web Services Metadata for the JavaTM Platform. Retrieved 15.05.2006: http://www.jcp.org/en/jsr/detail?id=181
16. Java Community Process (2006): JSR 921: Implementing Enterprise Web Services 1.1. Retrieved 15.05.2006: http://www.jcp.org/en/jsr/detail?id=921
17. JBoss (2006): JBoss application server. Retrieved 15.05.2006: http://labs.jboss.com/portal/index.html?ctrl:id=page.default.info&project=jbossas
18. JBoss (2006): JBoss SEAM, Retrieved 15.05.2006: http://www.jboss.com/products/seam/
19. JBoss (2006): JBoss. Retrieved 15.05.2006: http://www.jboss.org
20. JBoss (2006): JBossIDE. Retrieved 15.05.2006: http://www.jboss.org/products/jbosside
21. JBoss (2006): JBossWS. Retrieved 15.05.2006: http://labs.jboss.com/portal/index.html?ctrl:id=page.default.info&project=jbossws
22. JBoss (2006): SEAM - Contextual Components A Framework for Java EE 5 Version: 1.0.CR2. Retrieved 15.05.2006: http://docs.jboss.com/seam/reference/en/pdf/seam_reference.pdf
23. JBoss (2006): Supported Web Service Stacks. Retrieved 16.05.2006: http://wiki.jboss.org/wiki/Wiki.jsp?page=WebServiceStacks
24. JSR-127 expert group (2004): JavaServer™ Faces Specification Version 1.1. Retrieved 15.05.2006: http://java.sun.com/j2ee/javaserverfaces/download.html
25. JUnit org (2006): JUnit unit testing tool. Retrieved 15.05.2006: http://www.junit.org/index.htm
26. Kleijnen J.P.C. and Van Groenendaal, W. (2000). Measuring the quality of publications: new methodology and case study. *Information Processing and Management* 36: 551-570.
27. PostgreSQL (2006): PostgreSQL database. Retrieved 15.05.2006: http://www.postgresql.org//
28. SCOPUS (2006). Retrieved 15.05.2006: *http://www.info.scopus.com*
29. The apache ant project (2006): Ant build tool. Retrieved 15.05.2006: http://ant.apache.org/
30. Unified Modelling Language (UML) Documentation. Object Management Group (2005). Retrieved 31-05-06 : http://www.omg.org/cgi-bin/doc?formal/05-07-04
31. World Wide Web Consortium (2006): SOAP-protocol specifications. Retrieved 15.05.2006: http://www.w3.org/TR/soap/
32. World Wide Web Consortium (2006): Web Services Description Language specifications. Retrieved 15.05.2006: http://www.w3.org/TR/wsdl
33. World Wide Web Consortium (2006): World Wide Web Consortium (W3C) Home page. Retrieved 15.05.2006: http://www.w3.org/