

ADEPT

Analysis and Development of Electronic Publishing Technologies Project
Unit for Scientific Information and Learning, KTH, Stockholm, and
Department of Computer and Information Science, Linköping University

Erik Sandewall

Demonstrating the Use of Author-Deposit Restrictions in Publication-Related Software Systems

This series contains technical reports from the joint KTH/Linköping project
Analysis and Development of Electronic Publishing Technologies (ADEPT).

The present report can persistently be accessed as follows:

Memo persistent URL: <http://piex.publ.kth.se/reports/adept/004/>

Date of manuscript: 2009-03-08

Related information can be obtained through the following www sites:

KRF website: <http://piex.publ.kth.se/krf/>

The author: <http://www.ida.liu.se/~erisa/>

Abstract

The present memo documents a system demonstration for the sponsoring agency, by first describing the goals of the project and its major design decisions, and then describing the demo setup. The project concerns the management of IPR information that determines whether, when and how a given research article can legally be posted on a public website, in particular in an institutional repository or archive. The challenge is to make this information available in structured form so that it can be applied and used in the automatic operations of, for example, an institutional repository, as well as in the autonomous operation of software agents for providing assistance to their users. The demo uses a configuration of agents representing the different interested parties in an article's lifecycle, and shows how the IPR information can be represented, made available, and put to effective use in such a network of software agents.

Introduction and Definition of Topic

The preparation and publication of a research article involves activities by a number of individuals and (increasingly) a number of IT systems, as well as interactions between these persons and IT systems. On the systems side this includes author support systems, systems for management of institutional repositories and of local systems for a research group or laboratory, and systems for the editors and the publishers of journals. It is increasingly the case that these systems have to communicate with each other and exchange information. Several of these systems have additional interfaces. Author support systems, for example, may need to access bibliographic databases for information retrieval or simply for help with preparing the list of citations.

The overall design of this system of software involves a number of nontrivial issues. There may be arguments in favor of monolithic designs, for example, a publisher-provided, web-based system that authors can use for preparing their articles and that journal editors and reviewers can use for peer review and editorial routines. As always in such situations, there can also be extensive arguments in favor of an open architecture whereby each of the stakeholders (authors, editors, reviewers, publishers, etc) can use the software system of their choice, and where the smooth communication between these systems is guaranteed by proper standardization of data formats and interfaces.

We have chosen to focus on the latter approach in our work since we believe that an open architecture is the one that best serves the needs of the research community both in the short term and in the long term. In order to study aspects of this emerging architecture we shall model the entire system as a network of *software agents* that can exchange messages. Some of these software agents interact directly with a user and shall be seen as a *software assistant* for its user. Other agents are only used for the services that they provide to other agents, including both the service of providing particular information and the service of performing certain tasks. The latter kind of agents will be called *server agents*.

The task of creating the software for such agents and getting them to communicate with each other is a standard task in contemporary software technology. More interesting problems come up as we address requirements that are specific to the preparation and publication of research articles, which we shall refer to as the *research publication domain*.

One specific problem of this kind is addressed in the present report, namely, the handling of IPR-related information that regulates to what extent an author can legitimately deposit his or her research article on a public server, either as a "preprint" before the article appears in a journal, or as a "post-print" after journal publication. Different journal publishers impose restrictions of different kinds, including embargo periods, choice of which version and which graphical appearance of the article may be posted, and requirements to cite the journal publication in particular ways and providing particular information of a bibliographic or other character. In addition, of course, there is still a considerable number of publishers that do not permit such parallel publication at all, and those that require additional payment in order to permit it.

Besides the restrictions imposed by the publishers, there are also guidelines,

policies and requirements that are imposed by funding agencies and by the research institutions themselves. This is an increasing trend. It is up to the individual researcher to reconcile the requirements from different parties, but one of the tasks of a software assistant as described above may be to facilitate this chore.

There are also other problems that have to be taken into account when designing the proposed architecture. Most publishers have a standard contract that they propose to authors, but groups of authors or their institutions are increasingly proposing their own contracts that they wish the publishers to accept. Individual authors may also negotiate their own amendments to the standard contract. Furthermore, the allowances for the authors are not always part of the formal copyright agreement; in some cases they are stated separately in a webpage or other document where the publisher "permits" the authors to post their articles in particular ways. It may not be entirely clear what happens to this permission when new situations arise, for example, if the publisher in question is acquired by another publisher, or when the author moves to another employer. Is it then the old or the new employer university that shall be able to post the article on its website?

The proper management of this issue is of interest for the individual authors, but it is also of considerable interest for the operator of an institutional archive where scientific articles by the institution's researchers get to be posted. The operator of such an archive has to set up manual or automatic routines for verifying that the posting of articles is consistent with applicable rules and signed contracts.

From the author's point of view, besides the management of restrictions on, and conditions for postpublication, there is also the question of alerting the author that submission to a particular journal may not be possible if the manuscript has already been prepublished in a way that makes it ineligible for the journal.

If all these aspects are taken into account in a comprehensive system that is organized using software agents, it turns out that a certain level of cooperation and even of independent action will be required from those agents, for example:

- Assuming that each participating author has her or his own software agent, and that (as now) a journal editorial system will only communicate with one "corresponding author", the software agent of the corresponding author must inform the agents of the other authors about editorial decisions and assignments of bibliographic metadata concerning the article.
- The software agent that manages parallel publication for an author should be informed already when the author plans to submit an article to a particular journal. At that point it should raise objections if this is inconsistent with already effectuated prepublication.
- If a journal publisher requires withdrawal of a prepublished version of an article at the time of journal publication, or if it requires the prepublished version to be amended with bibliographic and other information, then the system should be designed so that this happens automatically. There are several ways how this can be designed into the system architecture, for example, by the software assistant of

the corresponding author automatically informing the prepublication server.

- Postpublication of an article shall only occur at a time that is acceptable according to the contract between the author(s) and the publisher, and with the graphical appearance and accompanying information that is required by that contract. Notice that this also requires that some of the participating agents must administrate individually made amendments to such contracts.

After considering these system design requirements and other similar ones, we propose that

- There is a need for an overall systems design (i.e., system architecture) for the set of various software systems that are engaged in the research publication domain.
- This overall systems design should be based on the concept that the individual participating systems are modelled as software agents with a capability to exchange messages with other agents, and to perform actions. Some of them will be software assistants for their respective users, with a capability of independent action on behalf of their users.
- The administration of IPR information for prepublication and post-publication is a good startingpoint and test example for this overall systems design, as well as for the implementation of the relevant software technology.

Agent Structure

For the purpose of the present study we define the following types of software agents in the overall systems design. Each agent type is described with the name that it has in the system.

- author-agent A software agent that is conceived as a software assistant for a particular person
- labmop A server agent for Management Of Publications on the level of LABoratory or research group
- instarch A server agent for management of an INSTitutional ARCHive
- editor-agent A software assistant agent for a journal editor that makes decisions about the acceptance of contributed articles
- jes A software agent for a journal. We model it as a server agent and not as a software assistant, and assume that it will communicate with editor-agents, author-agents and publisher-agents. The initiative for these communications will be variable
- publisher-agent A software agent for a publisher of scientific

journals. It is modelled as a server agent, like for journals. The officer in charge of a set of journals may require her or his assistant agents, but we leave this outside the model.

- jir A Journal Information Resource, that is, a server agent that provides information about publishers and journals, for example, their IPR conditions for pre- and postpublication.

Each agent maintains its own collection of information, for example as a database, and in terms of the model there is no common file storage or database for several agents. All information-sharing must therefore occur through message-passing which means that it is modelled explicitly.

Message-passing activities between these agents always start with a user command to one of the software assistant agents. The execution of the command may involve the sending of one or more messages. When an agent receives a message it executes a *script* that specifies what it is required to do. This may involve sending additional messages to other agents, or informing its user in the case of software assistants. It may also involve sending an acknowledging message to the sender, in which case messages appear in pairs that can be described in client-server terms. (¹)

This agent structure is of course merely one example of how things may be set up, and it is easy to think of alternative designs. However, our main concern shall be the mechanisms and rules for how IPR-related information is communicated, shared and used, and the described agent structure is sufficiently typical so that it is appropriate to use it for the study.

One question with respect to this model is whether it is actually realistic to maintain one separate instance of the software for each author. Is it not more realistic to assume that one journal editorial system, for example, communicates with all the authors of articles that were submitted to this journal, both by allowing the author to visit the system's website, and by having the system send e-mail to authors? This is admittedly the common arrangement today. However, the solution of having one support system for each author makes it possible to add more automatic facilities and to provide better facilities for the author, so it is a more forward-looking design. Having separate service websites for each journal, each institutional repository, and so forth puts a larger burden on the users since they then have to coordinate the information between these different services.

It is very unlikely that a single party shall be able to implement the software for all these agents and put them into operation, since journal publishers will certainly want to have their own systems, and since authors will want to be in full control of their articles during the authoring stage. The agent structure shown above must therefore be thought of as a *model* for the entire system, and not in its entirety as a specification for software to be implemented. Some of the participating agents may be implemented based on the model, but not all of them.

¹The question whether this will be required for all messages is a technical systems-design issue.

Demonstration Scenario for the PARPUB Project

The development program `OpenAccess.se` gives partial funding to our project *Domänmodellering av rättigheter och bivillkor vid parallellpublicering av vetenskapliga artiklar (Domain modelling of rights and conditions for parallel publishing of scientific articles)*. See reference 1 for the initial project proposal, and reference 2 for a milestone report (both of these are in Swedish language, however). The project consists of two main parts, one concerning the acquisition of parallel publication IPR information (i.e. SHERPA/Romeo-type information), and the other concerning the development of software techniques for making such information effectively available.

The present report concerns the latter subproject, focussing on the system demonstration that took place as scheduled on February 12, 2009. Reference 3 is a report from the other subproject. The purpose of the demo was to show the capabilities of the currently available parallel publication IPR information and to illustrate how it can be used in a systems context. This is a step towards the integration of such information in operational software systems in the research publication domain.

Agent Configuration for the Demonstration

The demonstration scenario defines one software agent for each one of the agent types shown above, except that three author agents are defined, each one serving one particular author. The following sequence of steps are used in the standard case where an article eventually gets published. Each step is effectuated by a command from the user to her software assistant agent, unless otherwise noted. It is assumed that this agent already contains the article and its initial metadata (authors, title, abstract, etc) when the sequence starts.

- *Register the article* with the laboratory-level agent, providing it with the basic information about authors and title.
- *Request prepublication* to be made by the laboratory-level agent (optional).
- *Register intention to submit* the article to a particular journal. This registration is again made with the laboratory-level agent. The author receives advise about submission procedure and gets access to help services for preparing the manuscript in the way required by the journal in question.
- *Submit* the article to the journal that was selected in the previous step. The software assistant of the corresponding author sends the manuscript and the required accompanying information to the journal agent, and receives a confirmation in return. It then informs the laboratory-level agent that this has happened.
- *Editor in charge* of the article specifies the acceptance decision to his or her assistant agent, which sends this information to the journal agent. This agent in turn informs the assistant agent of the corresponding author, which gives the information to its user and informs the laboratory-level publication manager.

- *Corresponding author proposes postpublication* of the article to the institutional archive agent. This agent inquires the laboratory-level agent about publication details, and then makes its decision about whether, when and in which appearance the article is to be posted on its public server.

Each step in this sequence requires that certain basic data about the article shall be transmitted, and several of the steps also require the transmission of full-text files. The message-passing shown above is merely a skeleton for the flow of events, and many of the steps require a number of things to be done, such as reformatting the article with additional information required by the publisher, selecting a web address (URL) or other data item that can serve as a persistent identifier of the parallel-published article and generating appropriate initial contents for it, and so forth.

Likewise, what is shown as one step in this sequence may actually consist of several interactions, for example, for the choice of journal. Interactions between co-authors of an article are not modelled in this sequence but they should be an important part of any author-support system that is used when co-authors work at different locations.

In general, a system architecture that is based on message-passing between agents will only work properly if there are one or a few *proactive agents* that check that each item in the flow moves forward (each article, in our case) and is not left behind. This role belongs to the author and the journal's executive editor, with the assistance of their respective software agents.

The Use of Parallel Publication IPR Information

The parallel publication IPR information that we consider in the PARPUB project consists of two parts: rules that have been extracted from the standard publication agreement of each publisher or (when applicable) each particular journal, and alternative rules that originate from the actual agreement between the author and the publisher. The former is the kind of information that is provided by the SHERPA/Romeo website, and it includes at least the following:

- whether the publisher will consider an article that has already been republished
- whether the publisher will accept postpublication or not
- whether the publisher requires separate payment in order to accept postpublication
- publisher requirements on embargo time, i.e., the required delay between publication of the article in the journal and its postpublication
- restrictions on the type of server (private, institutional, discipline-oriented, etc) that may be used for the above
- whether special rules with respect to the above aspects may apply for articles reporting work that has been funded by particular agencies. This includes both special rules made by the publisher, and special rules made by the funding agency

- which version of the article may be used, or may not be used in the case of postpublication, according to the publisher
- publisher requirements on what information must be included in the postpublished version of an article, or in web pages that serve as "wrappers" for such articles, or in web pages containing a list of such articles
- which graphical appearance of the article may be used, or may not be used, according to the publisher
- publisher requirements on what must be done to a prepublished version of an article at the time when the final article is published in the journal

All of these are actual restrictions that are being made by some publishers. This rule base has a considerable complexity, therefore. Notice in particular the potential interaction between the requirements of funding agencies and those of particular publishers, and the need for advise to the authors about how to handle such conflicts. Notice also the complexity of rules that arises when different sets of rules apply depending on the funding agency, or depending on whether the author agrees to pay a parallel publication charge.

Once the types of IPR information has been identified, it follows fairly directly how it has to be used in the publication-domain architecture. There are basically three kinds of uses:

- *early warning* in order to inform or remind the author of restrictions that may be relevant for his or her decisions about an article. For example, if the author registers the intention to submit a previously prepublished article to a journal that does not accept such submissions, the author shall be given a warning. The same applies if the author has submitted the article to such a journal and then proposes republication during the reviewing period
- *impose restrictions on actions*, in particular, prevent the laboratory-level or institution-level archive from posting an article in cases where this is inconsistent with known agreements between author and publisher
- *information formatting actions*, including in particular the reformatting of an article in order to comply with publisher requirements (as well as institutional requirements, in fact), and the generation of web pages that present an article or a set of articles

It is fairly straight-forward how to incorporate these kinds of actions in the publication agent configuration described above. The plan for the demo event is to show as many as possible of these facilities in the context of this demo configuration.

The Resource for Parallel Publication IPR

A key resource for the demonstration is of course that there must be a server agent that is used as an information resource concerning the parallel publication rules of the publishers that may be of interest for the system's user

community. The SHERPA/Romeo database is the most obvious choice for this purpose. However, at the present time it has the following limitations.

- Incomplete coverage of publishers
- For some publishers, different journals have different standard contracts; this is not represented in the available database
- Much of the information described above is only available as free text in ordinary language; it needs to be rendered in structured form in order to be of use for the purposes described here
- Some of the contractual and parallel-publishing information is not written into the standard copyright release contract, but is provided by separate statements e.g. on the publisher's website. This makes it difficult to collect the information and to assess its legal strength.

The task of making the database more complete has several additional difficulties: not only the sheer size of the undertaking, but also the fact that some publishers are unwilling to unveil their standard publication agreements.

As already mentioned, a complementary part of our present project aims at making the database of parallel publication rules more comprehensive, but the results of this work will not be available for the demonstration that is now being prepared. We shall therefore use a resource that makes indirect use of the present contents of the SHERPA/Romeo database. Two approaches are being explored in parallel in this respect.

The *SHERPA/Romeo Interpreter* is an agent that inquires the SHERPA/Romeo API each time it receives requests from some of the agents described above. It interprets the response from the API and applies it to the case at hand.

The *Parallel Publication Advisor* is a software agent that contains a knowledgebase of information about publisher's restrictions and requirements for parallel publication. This knowledgebase is used for answering queries of the same kind as those that are directed to the SHERPA/Romeo Interpreter. The present knowledgebase consists mostly of information that has been extracted from the SHERPA/Romeo website, but it is intended that it will be gradually extended with information obtained from the other part of the present project.

Each of these approaches has its advantages and drawbacks with respect to continued work. The advantage of the SHERPA/Romeo interpreter is that it automatically provides access to additional information that is added to the SHERPA/Romeo site. The advantage of using the Parallel Publication Advisor, on the other hand, is that we are able to move ahead and implement new types of information and new methods for extracting information directly from publisher sources.

From the point of view of the demo configuration and in a short-range perspective, it will actually be an advantage to have two server agents that provide similar services, since this is quite realistic in a practical situation.

Other Software Agents in the Demo Configuration IPR

The following additional software systems are being used for the demo configuration.

The LISTA Design Tool for Software Agents

The *Leonardo Implementation of State Transfer Agents* (LISTA) is a simple framework for the implementation of software agents such as those described above. It is based on the Leonardo computation system and the embedded web server in Allegro Common Lisp (ACL). Each agent is therefore implemented as a web server, and agents communicate in a client-server fashion by sending and answering http requests, using a REST-like (Representational State Transfer) philosophy. Besides setting up the agents themselves, the LISTA package also provides a framework for defining the structure of the states that are being transferred in messages, and for defining the scripts according to which agents process incoming requests.

The choice of this approach was motivated by two reasons: we wish to make it as simple as possible to implement agents and to equip them with their scripts, and we believe that this approach will make it straightforward to interface existing, operational software systems to the configuration.

The MADMAN Author Support System

The *Manager for Articles, Data, Manuscripts And Notes* (MADMAN) is an author-support software system that has been developed in our group since a number of years. It is described in a separate memorandum (reference 4). MADMAN software modules are loaded into the the software agents for author assistance and for laboratory-level management of articles in order to provide them with capabilities for formatting of articles and for generation of web pages which are needed for their proper operation.

Outcome of the System Demonstration

In the demonstration we showed the successive steps that would be taken in the lifecycle of one particular article, where successive variants of the article with appropriate cover layouts were generated at successive points. In particular, the journal where the demo article was virtually published requires a relatively complex preset paragraph containing bibliographic information on the first page of a parallel published article; this was generated correctly. It is shown in reference 5.

In general, the overall configuration and all the participating subsystems worked correctly and as intended during the demonstration.

Future Plans for System Integration

The following additional software systems are being considered for inclusion in the system at a later time.

The JARSS Editorial Support System

The *Journal Editor and Reviewer Support System* (JARSS) has also been developed in our group and is in actual use by a number of journals. It is a mature system in the sense that it has been in use since year 2003 and since it has gone through several rounds of extensive rewrite. We are planning to interface JARSS to the demo configuration at some point, but it is not being planned before the upcoming demo event.

Swedish Institutional Archive Systems

Plans for the future include making contacts with the developers and users of institutional archive systems at Swedish universities and to offer them access to the parallel publishing IPR resources that have been developed in the present project. As far as we can tell at the present time this ought to be a fairly simple interface and it should not have any major effects on the structure of the archive system software.

Author Support Systems

We similarly wish to investigate the possibility of including one or more existing author support systems in the present architecture. By comparison with the case of institutional archives, this is however likely to require more substantial changes in the author system software in order to implement IPR-related services.

References

1. Erik Sandewall: OA-publicerade domänmodeller avseende vetenskaplig publicering och gruppstruktur. Organizational Memo number 4, Division of Publication Infrastructure, KTH, Stockholm. (Initial project proposal for the present project). Available at <http://piex.publ.kth.se/reports/rapp/004/>.
2. Erik Sandewall: Delrapport för OpenAccess.se-projektet Domänmodellering av rättigheter och bivillkor vid parallellpublicering av vetenskapliga artiklar. (Milestone report, January 2009). Available at <http://piex.publ.kth.se/reports/rapp/007/>.
3. Preben Hansen, Gunnar Eriksson and Oscar Täckström: Steps towards automatic acquisition and recognition of IPR conditions for parallel publishing. Project report, Swedish Institute for Computer Science (SICS), 2009.
4. Erik Sandewall: Support for Managing IPR and Parallel Publishing in the MADMAN Research Author Support System. Available at <http://www.ida.liu.se/ext/caisor/pm-archive/adept/002/>.

Demonstration Data

5. Lars Larsson: Seeing Farther than Others while Standing on the Shoulders of Giants. Mockup article. Front page for postpublication purposes of an article that has supposedly been published in the *Journal of Biomedical Optics*. Available on the webpage of the present report, <http://piex.publ.kth.se/reports/adept/004/>.