CAISOR

This is a project memo from the CAISOR-ADEPT activity concerning

Analysis and Development of Electronic Publishing Techniques

Erik Sandewall, Daniel Bergström, Björn Husberg

Design of a Reactive, Database-Centered System for Electronic Publishing: The JARSS Experience

This project memo (PM adept-001) is disseminated through the CAISOR website for Cognitive Autonomy and Information Systems and has URL `http://www.ida.liu.se/ext/caisor/pm-archive/adept/001/`

Related information can be obtained via the following www sites:

CAISOR website:	`http://www.ida.liu.se/ext/caisor/`
ADEPT website:	`http://www.ida.liu.se/ext/adept/`
The author:	`http://www.ida.liu.se/~erisa/`

Date of manuscript: 2006-09-15

1 Introduction

Two major types of software systems for electronic publication are similar in some ways and different on one important point. We think of author support systems, on one hand, and editorial support systems on the other. Author support systems (reference) facilitate the work of a single author or a group of authors for preparing research articles and other documents, not only with respect to the writing as such, but also for the administration of underlying data, citations, and other sources, and for the administration of the submission and eventual publication of the article. Editorial support systems (reference), on the other hand, facilitate the work of journal editors, conference program chairs, and the reviewers that participate in the peer-review process.

Both these types of systems are database-oriented, in the sense that they rely on a comprehensive database of bibliographic information, full text of articles and auxiliary texts such as abstracts and reviews, and person-related information about authors and others that are involved in the respective process. However, they differ in an important aspect: author support systems are mostly proactive whereas editorial support systems are almost entirely reactive and workflow-oriented. The present article will discuss the design of the latter type of software system against the background of the experience with developing and using a particular editorial support system called JARSS, for Journal Author and Reviewer Support System. We conclude with a proposal for a perspective of 'major' and 'minor' workflow in the design of such systems.

2 The Editorial Reviewing Process

The character of the editorial reviewing process is probably well known to many readers, but it is summarized here for completeness. The object matter of the process is the individual research article, which starts as a manuscript written by one or more authors, and ends up as being sent to a publisher for inclusion in a particular journal. The article's trajectory begins when the authors send it to the editorial office of the journal. It is received by the editor-in-chief who selects reviewers, obtains their agreement to review the article, and sends the article to them. The reviewers return their recommendation, which may typically be 'accept', 'reject', 'accept after recommended changes', or 'recommend major changes to be made, and receive for renewed review after that'. The editor-in-chief integrates these recommendations and makes the decision. Some revisions are usually requested so the article goes back to the authors, they send in a revised article, and the revision may or may not go to the reviewers (usually the same ones as on the first round) for a second check.

This is the routine in principle. In practice there are a number of important modifications. First and foremost, obtaining the reviewers and the reviews is often the major difficulty from the point of view of the editor-in-chief. This is understandable since the peer review work is intrinsically not rewarded, but it is still a problem which needs to be alleviated. The work process is modified from this point of view. Large journals use associate editors that help the editor-in-chief, either by recommending suitable reviewers, or by entirely taking over the review coordination process for articles within their area.

One part of the problem is that some proposed reviewers decline the invitation to review. Another problem, and a bigger one, is that some accept to review, but still do not produce the review at all, or only after several reminders. A good editorial support system should have built-in facilities that provide support for dealing with these harsh facts of editorial life.

Yet another issue is that submitted manuscripts are not always in a form that is suitable for sending to reviewers. There may be technical formatting problems; manuscripts may be submitted in proprietary formats (e.g. Word .doc format) that are not accepted by all reviewers, and so forth. Furthermore, many journals require additional information to be sent together with the submission, for example statement of originality or of non-duplicate submission, and not all submissions comply with these requirements. Therefore, the complete workflow includes the possibility that an article has been submitted but the editorial office must contact the author for amendments. The article is then said to be in 'pre-received' state; it is 'received' when all parts of the submission are in correct order so that reviewing can begin.

There are also several other variants on the basic scheme, for example, the occurrence of special issues where there is one or a few guest editors for the special issue which act in the place of the editor-in-chief or of an associate editor for the special-issue submissions. Special types of contributions require special treatment, for example book reviews, public critique of submitted articles and authors' rebuttals to such critique, and so forth.

3 Design Considerations

An editorial support system must be able to support the process of submitted articles from submission, through peer review and revisions, to the point where they are delivered to the publisher. It must therefore be able to administrate the data and the documents that are involved in that process. It must also be able to administrate the data that some of the actors in this process need in order to make their decisions. In particular, in order to designate reviewers for a given article, one must know about (a) the research competence of possible reviewers, (b) their track record when they were asked to review before, (c) their present workload for the journal, (d) possible conflicts of interest. Some of this information is in the head of the editor-in-chief, or who else makes the choice of reviewer, but much of it can be provided by the editorial support system.

The design of the editorial support system must also make a choice about how the communication between the agents in this process is performed. There are in principle three possibilities:

- Outside the system. Communication proceeds by electronic mail (typically) and the person using the system operates the e-mail separately, enters information into the system when email messages are received and acted upon, and consults the system for making decisions.
- Within the system, operated as a website. Editors, reviewers, etc. visit the system website in order to obtain information and to communicate. The website provides capabilities e.g. for uploading reviews.

- Using email, but the email is integrated into the editorial support system. In particular, individual reviewers communicate with the journal editors using electronic mail, but the editors use the support system for receiving incoming email, for administrating it, and for generating outgoing email.

The third alternative is a 'hybrid' in the sense that it is intermediate between the first two. There is also a weaker 'hybrid' solution where the support system contains facilities for generating outgoing email, but not any facility for receiving and administrating incoming email.

These communication alternatives can in fact be applied for several of the communication lines: between editors and authors, between editors and reviewers, between editor-in-chief and associate editors, and so forth. Different solutions may be appropriate for different communication lines.

4 Overview of Design in JARSS

The JARSS system (Journal Author and Reviewer Support System) is accessed by all its types of users using a web interface. A user may enter the system in one of the recognized *roles*, namely author, reviewer, associate editor, or staff. The staff role includes editors-in-chief(s) as well as administrative editorial staff. Users enter by login and password, and each registered user may have access to one or more of these roles.

From a database point of view, the system is built around four major types of entities, namely *persons*, *submissions*, *messages*, and *events*. The persons in this sense are those that have occurred as author or as candidate or actual reviewer. Each submission of a new manuscript defines a submission entity in the system; revised versions of the same manuscript count as belonging to the same submission. Messages are mostly the incoming and outgoing email messages, but there are also internal messages entirely within JARSS. Events are typically the decisions about an article that result in, or follow from peer reviews.

There are a number of relationships between persons and submissions, in particular, the author-of and the reviewer-of relation. There are two major views of the information in the database, namely the view for a particular person entity, and the view for a particular submission.

The submission view is the most important one; it is obtained when the user looks at the information pertaining to a particular submission. This view shows (a) bibliographic metadata, such as author(s) and title; (b) information pertaining to the manuscript and the communication with the authors; (c) information pertaining to the reviewers and the reviews; (d) procedural information within the editorial office. For both the (b) and the (c) section there is structured data, document files, and email messages. This means that on one single page, the user of the system is able to obtain an overview of the communication that has occurred concerning the article as well as the versions of the article, the reviews of successive versions, and so forth.

The person view presents some of the same information, but as it reflects on the particular person. For example, while the submission view shows a list of the reviewers and their present communication status (date when asked, when reminded, when he/she promised to be ready, etc), the person view shows a list of which articles this person has submitted and whether they were accepted, as well as a condensed summary of his/her reviewing track record (number of articles reviewed, number of weeks to return review for each of them, but no additional details).

This means that JARSS uses the 'hybrid' approach to communication with users. Editors-in-chief and other staff use the system by direct interaction with the website, but reviewers, in particular, communicate using standard email. To be more precise: it is possible for a reviewer to make all communication via email; it is also possible to for the reviewer to use the website in specific situation, for example, for submitting the finished review and filling in a form for the 'formatted' part of the review. We have chosen this solution because it is certainly not possible to use the website exclusively (which researcher would visit a website periodically only in order to find out whether he had been invited to review a few more papers?), and therefore it was best to build the system so that it integrates the handling of both outgoing and incoming email. At the same time, offering an alternative and direct way of submitting specific types of information is a useful service that is not difficult to implement.

For the editor-in-chief and other staff, who are then the major users of the system, the system is therefore a tool for responding to messages, and in particular to e-mail messages. This has a number of direct consequences for the design:

- The system user is confronted with a large message flow, and it is important to provide as much automatic support as possible for administrating those messages. In particular, the JARSS system has a facility where incoming messages are automatically sorted into the mail folder pertaining to the particular submission that it addresses. Also, outgoing messages are of course always placed in the correct folder. (Typically the user generates an outgoing message when he/she is working with the webpage describing one particular submission).
- Although the system shall make sure that messages are related automatically to the right submission, it must also present those incoming messages that require attention, and the presentation must be in a reasonable and logical order. For example, the system maintains an 'alert list' of overdue reviews. If the user enters the system with the idea of first attending to the alert list and to do something about these overdue reviews, he/she must be able to see at a glance whether an email has just come in for any of these, so that he does not send out a reminder when a review has in fact already arrived.
- Since a considerable part of the work with the system consists of reading e-mail messages and acting on them, it is important that the presentation of an e-mail message shall be accompanied by a number of *effect buttons* that can be clicked in order to implement a particular action or decision. For example, if the user has brought up an incoming email message containing a submission, it shall of course be possible to create an answer email immediately, either by writing it manually, or by generating it from a script. However, it shall also be possible to initiate a new submission entity, obtain its registration number so that it can be placed in the Subject field of the return

> message to the author, lift the attachment files in that email message into the manuscript folder for the new submission entity, and confirm that the status of the submission has advanced from 'pre-received' to 'received'. Another effect button implements the alternative course of action where the author receives a message requesting additional information, and keeping the submission in pre-received status.

The point with the effect buttons is that they shall allow the user to get the work done without having to navigate around the website. Without the effect buttons, the user would first have to be in one part of the system in order to read the mail, and then go to another view in order to generate an outgoing email, and a third view in order to reset the status of the article.

5 Conclusion I: Major and Minor Workflow

The discussion in the previous section can be summarized and expressed in more general form as follows: In the design of a reactive software application, like the editorial support system, it is important to recognize both the major and the minor workflow. The major workflow is the well-known one, and in the case of an editorial system it is the editorial routine for submitted articles that was described by way of introduction in this article. Major workflow is important because the individual decisions, in particular by editorial staff, are often represented as transitions in the workflow for that particular article.

Minor workflow, on the other hand, is the elementary operations that the user performs when he or she uses the software system, for example for going from an overview of pending incoming submissions, to looking at one particular submission in that list, and from there to looking at candidate reviewers, and selecting one of them, and adding one more person to the database since she should be asked to review this submission but was not yet in the database. In the detailed system design it is important that the minor workflow is as efficient and effective as possible.

Major and minor workflow are related: major workflow specifies what has to be done, and minor workflow specifies how to do it. Therefore, the ultimate test of the correctness of the minor workflow, is that it performs major workflow correctly. Everything beyond that point has to do with convenience and performance, and in particular with the convenience and performance of the user.

6 Conclusion II: How Much Configurability?

Configurability is an important aspect of many applications, and also for editorial support systems. There is the configurability per journal - every journal may have its own variation of the review process - and the configurability from user to user, corresponding to different user work situations and preferences.

One major means of configurability is to allow major workflow to be table-driven, so that there is a data structure defining the successive 'states' that a submission may be in, and there are operations for changing the state of a submission. The JARSS system provides configuration in that sense, and it is quite important both for facilitating the gradual development of the system for one particular customer/journal, and for adapting the system to the needs of different journals

At the same time, it is our experience that this is not enough for obtaining the best possible quality of service. A direct implementation of table-driven major workflow can not easily take the considerations of minor workflow into account, such as the need for effect buttons of the kind described above. The work must therefore be done on several fronts. Besides the model of major workflow, there must be active involvement with qualified users in order to understand the real-world requirements on the system on both major and minor levels. The actual workflow model in the JARSS system for the AI Journal is considerably more complex than the simplified account that was described in the first section, for example.

Also, if one assumes that it is worthwhile to use explicit models of workflow as the backbone of the implementation, then it becomes necessary to build, implement, and maintain models of minor workflow as well. This is certainly within the reach of technology, but the effort is not negligible, and at some point one must ask whether it is cost-effective compared to the alternative of direct implementation on the programming-language level.

7 Acknowledgements

The development of the JARSS system was financed by a contribution by the Artificial Intelligence Journal Division (AIJD) of the International Joint Conferences on Artificial Intelligence (IJCAI). The development work and the design has benefitted very much from the ideas and advise by the journal's co-editor-in-chief, Ray Perrault.

8 Annex: Implementation History and Status

Our systems development in this area started with an earlier system, called JESS, that was implemented quickly in 2002 in order to meet an immediate need. The first JARSS system was implemented by Björn Husberg in 2003, with updates during 2004. Daniel Bergstöm made many additional, gradual improvements during 2005, and rewrote major parts of the system during the first part of 2006. Erik Sandewall wrote the initial JESS system and directed the continued work on JARSS.

The system is implemented in Allegro Common Lisp and runs as a web service under https access. The implementation is very stable. We are presently preparing to open up the use of the system to additional journals besides the AI Journal.