SCOPE – A Generic Framework for XML based Publishing Processes

Uwe Müller, Manuel Klatt

Humboldt-Universität zu Berlin
Computer- und Medienservice
Unter den Linden 6
10099 Berlin
{U.Mueller, Manuel.Klatt}@cms.hu-berlin.de

Abstract. One of the objectives of the Open Access movement is to establish institutional repositories at universities and other research institutions in order to support self-archiving. Although a lot of software solutions have already been presented in recent years they lack a seamless integration of authoring tools, support for authors, and other technical publication tools.
This paper presents a formal approach to describe software components applied in publishing processes. Additionally it is depicted how this formal description leads to the technological basis for SCOPE (Service Core for Open Publishing Environments) – a publishing platform for XML based publishing models. SCOPE is a framework intended for the integration of different publication components into a single platform.

Introduction

Presently scholarly communication including scholarly publishing faces one of the most extensive revolutionary processes of its history so far [1]. Basically, this is determined by the fact that publication processes rest upon and are realized as well by electronic technologies and infrastructures to a growing extent. As a main condition, the spreading of the internet has facilitated a much faster dissemination and a wider reach of information and thus also serves as the basis for a new method of scholarly communication.

Obviously, the technological changing does not only concern the creation process of a publication. As a matter of fact this first phase of the publication chain has been nearly completely aided by computers and word processing systems for decades now. But electronic publishing involves more aspects than that such as the submission and review processes, metadata handling and retrieval mechanisms, conversion and transformation tools for different presentation and archival formats (including paper printing), and long term preservation. Nevertheless, the creation phase of a publication is crucial for the following steps and the technological quality of the overall electronic publication. For these reasons the technical qualification of authors as well as the development of appropriate authoring tools to support them became very important issues.

Coinstantaneously to the technological variation of publication processes, the organizational and economic perspective of publication models is changing as well. Traditionally it has been the duty of commercial publishing houses to organize and manage the publication process. Nowadays this task falls – at least partially – to the research institutions. Mainly driven by the Open Access movement, universities and other scientific institutes newly play the role of publishers and therefore have to establish appropriate infrastructures. Among others [2, 3, 4] the adoption of the *Berlin Declaration on Open Access to Knowledge in the Sciences and Humanities* [5] marks a significant milestone on the way to the introduction of institutional Open Access servers.

As the establishment of institutional repositories [6] is one of the main objectives of the Open Access movement several according software solutions have emerged during the last couple of years. The most important ones are *eprints.org*, developed at the University of Southampton and applied for more than one hundred repositories [7], and *DSpace*, originally designed to manage the digital collections at MIT [8]. While they implement upload and management functions as well as parts of a publishing workflow both systems lack the possibility to integrate technical workflow components, such as authoring and conversion tools, with the aim to model the technological part of the publication process. E.g., none of the established institutional repository software solutions is capable of handling XML documents in an appropriate way (management of document models, conversion from current word processing systems, and transformation to presentation formats).

Motivation

Since 1997, the Electronic Publishing Group, based at the Computer and Media Service and the University Library of Humboldt University, deals with problems of scholarly electronic publishing. Among other things, the group has established a certified document and publication server (*edoc* server)[1] for Humboldt University. Starting with electronic theses and dissertations, both, technical guidelines and a policy for the edoc server [9] have been developed, defining quality standards for scholarly electronic publishing.

One of the most distinctive insights of the first projects launched in conjunction with this subject[2] is the strong recommendation to use a structured and open document standard as the base file format for electronic publications. The main indicators for this decision are the desire to be company independent, to allow different presentation formats, to enable high quality retrieval and document browsing, and to accomplish the basis for long term preservation activities. Primarily, these considerations led to SGML as the central data format. Later, XML has emerged as the standard for structured document formats.

According to this understanding, various document models, template files, conversion programs, presentation styles and other tools have been developed for numerous

---

[1] edoc server of Humboldt University Berlin: http://edoc.hu-berlin.de/
[2] projects *Digitale Dissertationen* (1997-2000) and *Dissertationen Online* (1998-2000)

different publication types and individual requirements. Different publishing workflows have been implemented centering on SGML / XML.

While the overall amount of documents published at the edoc server has been manageable until a short time ago the advancing debate on Open Access causes sensibly increased demands for electronic publishing of scholarly documents on the institutional repository. Likewise, apart from university affiliations the established publishing services are now offered to external research institutions that are lacking the organizational and technological infrastructure to establish their own robust publication server.

Among other things, this results in a more complex variety of related and dependent tools and different workflow models. Moreover, for reasons of costs and complexity, there exists a need to assign some of the operations originally accomplished by the working group's staff back to the institutes, chairs and editors who are responsible for the publications. E.g., it is necessary to enable editors to autonomously manage their publication series. Certainly, this has to happen in a convenient and user friendly way.

Thus, the requirement to more formally manage and describe the different tools, their versions, properties and relations to each other has emerged. Furthermore, the workflows modeling the publishing processes have to be configurable more extensively and easily. The subsequently presented SCOPE architecture aims at satisfying these needs.

The motivating vision of SCOPE is to provide a service platform that meets user defined requirements and preconditions and is able to deliver all necessary tools and a workflow definition enabling a scientific editor to conveniently realize and manage his or her own XML based publication series.

Publication Components

In order to formally describe the technological parts of publishing processes SCOPE has defined *publication components*. A publication component (PC) is a distinguished software tool responsible for an atomic step within a publication chain. In most cases it is a publication component's abstract function to transform a document from one defined state to another. Besides, there are tools intended to validate certain document properties and components to generally support particular publication steps – especially the creation phase of publications.

Describing a publication process on the basis of PCs turns out to be an especially sensible part if the workflow is very technology centered. Typically, this is appropriate in case of XML based publication processes, where a multitude of validation, conversion, and transformation steps is necessary for each single document. As indicated in the preceding sections the application of XML as the core document format is considered to implicate many benefits in terms of high quality publishing.

Below, we will enumerate typical classes of publication components, each with some examples already developed and applied within SCOPE:

− *Document models*. They take the central position within the SCOPE framework. Up to now, a couple of different Document Type Definitions (DTD) is available –

among others the Dissertation Markup Language (DiML). Moreover, a DTD generation system has been developed, allowing individual compilation of DTDs and according example documents as well as reference manuals (see section *Workflow System*).

− *Authoring tools.* The best known representatives of this class are document styles and templates for common text processing systems. They enable authors to write their publications in a structured way and additionally keep ready some basic check facilities. The proper usage of template files allows an accurate conversion of the documents to XML. Appropriate document styles and templates for the existing document models are available within SCOPE for Microsoft Word and StarOffice / OpenOffice. Although the templates contain an own bibliographic management system SCOPE also supports current commercial products such as Endnote and Reference Manager. In order to transform the bibliographic information into valid XML segments, a variety of particular styles has been developed for these systems, integrating both, the layout and the structure of the references.

− *Validation and correction tools.* As a counterpart for the offered authoring tools, we have developed a variety of validation and correction tools. They are responsible to verify submitted documents in respect to formal correctness, and – automatically or user supported – to correct problems and formal errors. For example, there are macros to validate the proper usage of templates, the correct handling of images, etc. There also exists a script inspecting submitted PDF files, e.g. the proper embedding of used fonts. Partially, these tools have been integrated into the authoring tools to allow creators to independently validate their files. Other tools are integrated into the submission site of the document server.

− *Conversion tools.* In order to convert documents from one file format to another, a variety of conversion tools has been developed. Among others, scripts have been implemented to convert Microsoft Word or OpenOffice / StarOffice files into valid XML files according to the respective document models. On the other hand, presentation tools, such as XSLT scripts and XSL-FO styles have been developed to allow dynamic or static generation of HTML and PDF output based on the underlying XML sources.

− *Metadata tools.* Metadata plays a vital role in terms of indexing, detection and retrieval of electronic documents. Within the SCOPE framework metadata is managed with the aid of a metadata database and widely configurable input and search / browsing interfaces. These tools can be used by authors and staff as well as by researchers, using the publication server as an information source. Some of the metadata – especially technical metadata – can be extracted automatically from the received documents by according extraction tools.

− *Long term preservation tools.* To transparently ensure integrity and authenticity of the uploaded documents SCOPE uses electronic signatures. For this purpose, partially, tools by a commercial contractor are used. Other components have been self-developed. Development of further long term preservation components will be one of the upcoming businesses of SCOPE. Particularly, we intend to implement tools conforming to the OAIS reference model approach [10].

Most of the PCs available in the existing system have been developed or deployed within SCOPE. But basically, the introduced framework is entirely open to existing or externally developed tools and components.
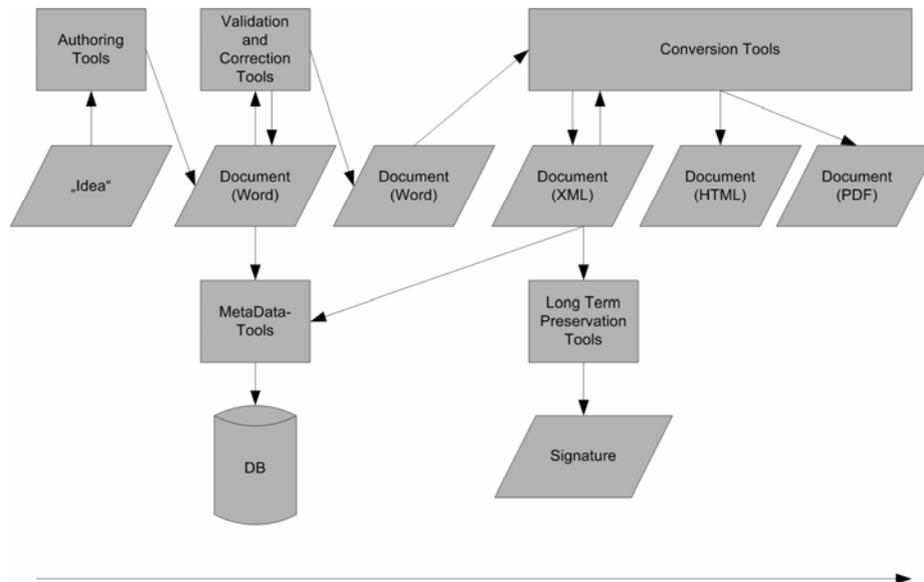
Fig. 1: Example document workflow using publication components

Fig. 1 provides a simple example process showing the way of a scholarly publication from the idea to the final presentation version. In this case the Word document comes into existence with the aid of an *authoring tool*, e.g. a template. Subsequently, it is transformed by a variety of validation and correction tools before it is converted to XML and afterwards to HTML. The PDF version could be generated directly from the Word file or from the XML source using XML-FO. Moreover, metadata and digital signatures are created by the respective tools.

As will be shown in the remaining sections the PCs' approach is suitable for abstract specification of publishing workflows. It represents the basis for both, the component management system (see next section) and the overall workflow system (see section on *Workflow System*).

PC Management System

The publication components necessary to convert documents from one format to another do not exist as context free modules or tools. They depend on each other and may be related to each other as the following examples show: A certain style file is designed for a peculiar DTD A; it could also be used in conjunction with DTD B and Schema A´ but is not applicable for documents structured with DTD C. Since they may have been derived from another component and inherited some properties, they may exist in different versions applicable under different conditions and the like. In order to provide a publication service to editors and authors on the basis of the aforementioned PCs a management system has been developed.

As indicated in the preceding section, a PC is basically characterized by the sort of files it is able to process – more precisely – by the properties of its source and target files.
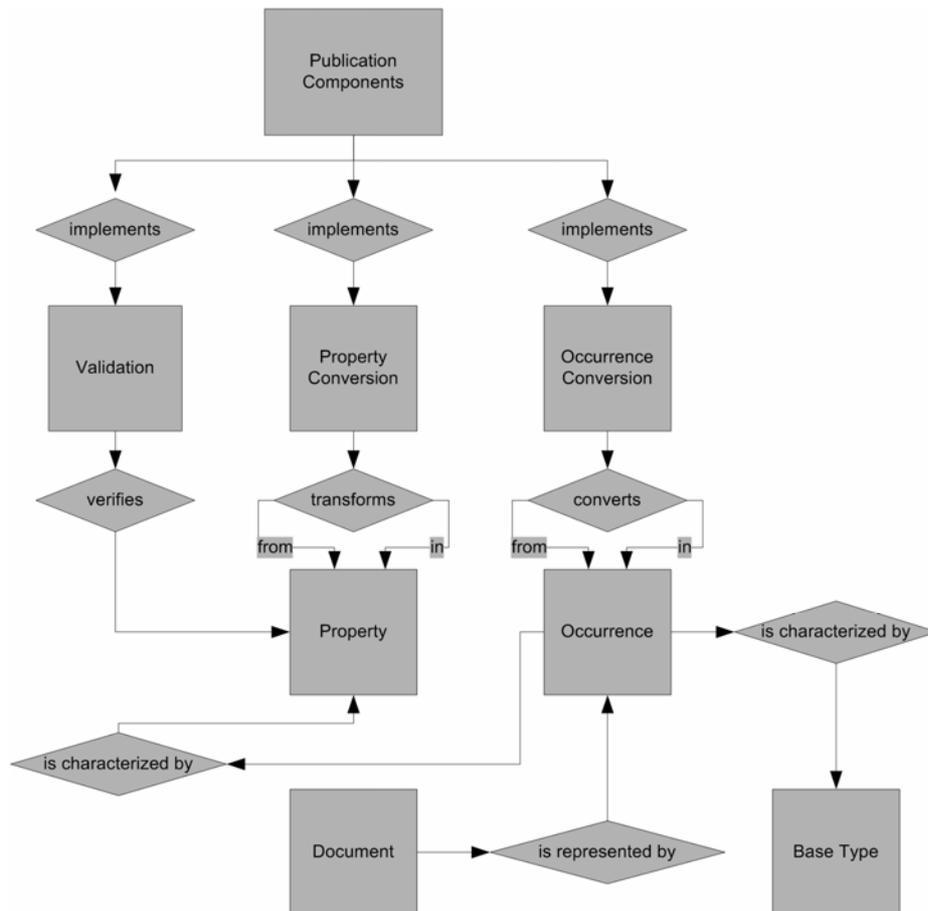
Fig. 2: Correlation between PCs and documents, illustrated in an entity relationship diagram

Fig. 2 depicts the simplified formal relationship between PCs and documents using an entity relationship (ER) diagram. While a *document* is interpreted as a differentiated entity of intellectual output only defined by its content and logical structure, it can be represented by different *occurrences*, e.g. a Word or XML file with certain properties or a paper copy. An occurrence of a document is characterized by a *base type*, i.e. the file format (e.g., MS Word, PDF), and a set of attributes or *properties*, e.g. the page setup, the used fonts and templates, and the type of the embedded pictures. A *publication component* (see previous section) is able to transform certain properties into each other or to convert one occurrence into another one. There are also PCs, e.g. validation tools, which only verify properties and do not change them.

The ER diagram shown in Fig. 2 depicts this correlation by the entities *property transformation*, *occurrence conversion*, and *validation*.

Based on the formal description approach for PCs indicated in Fig. 2 a database model and an according *PC management database* (PC-DB) has been developed mapping the aforementioned interrelations between publication components and documents. Basically, the database serves to contain the formal characteristics of PCs, which are substantially more complex than explained in the preceding paragraphs. I.e., each PC developed or used within SCOPE has been described on the basis of the presented formal approach and stored in the PC-DB.

Among other things, this step provides a big advantage for both, the developers of PCs and the staff being involved in advisory services for publishers: Using simple database requests up-to-date PCs can be very easily and conveniently retrieved and detected according to the respective requirements in case they already exist. Otherwise, the unsuccessful request to the PC-DB simultaneously provides a formal basic description of the PC to be developed.

Thus, the most evident purpose of the PC-DB and the underlying description approach is to efficiently manage a growing and increasingly complex variety of publication components.

Clearly, there are a lot of interrelations and dependencies between the respective considered PCs. Apart from the different versions representing the varying status of development, PCs are primarily tied together by the specific properties and occurrences they are able to process. E.g., a PC implementing a property conversion with certain source and target properties can be substituted by another PC with equal characteristics.

However, the most interesting application of the presented system and the dependencies of PCs is the implicit modeling of whole technological publishing processes. Interpreting the target format of a PC as the source format of another, it is eventually possible for the management system to adequately respond to requests such as *"Give me all PCs necessary to electronically publish MS Word files in a certain HTML layout."* Basically, this request is translated into recursively arranged simple database requests as mentioned before and leads, if available, to a chain of PCs forming the publishing process starting from the originating MS Word file and ending up with the specified HTML layout.

In general, using this system it is possible to determine, which tools are necessary to convert a document with certain properties to a document with other properties, occurrence or presentation formats. The management system has been implemented prototypically to study its functionality. The formal specification of PCs and the PC-DB is also used for the SCOPE workflow system presented in the next-but-one section.

DTD Management System

The aforementioned PC management system's primary focus is on tools and word-processing properties. As discussed in the first two sections, documents are converted into XML by default, for reasons of high quality retrieval and long-term preservation. As there can quite easily be understood, different XML tags may be necessary to fully describe different documents.

A simple example is a document with mathematical formulas. To express and tag formulas correctly the underlying DTD must be adapted. Other documents may not consist of formulas at all. For these documents a simpler DTD without formula support is sufficient. One solution to accomplish the various requirements is to provide a universal DTD to accommodate all different documents. However, this would lead to a high complexity and difficult maintenance.

For this reason we have chosen a different approach and have developed a DTD generation system, *DTDSys*, allowing for the compilation of individually assembled DTDs. Therefore, the elements of the virtual *"universal"* DTD have been grouped into modular units. These modules which are XML files themselves are stored in the *DTDBase*.

Using DTDSys – a transformation system on the basis of XSLT and Java – the modules can be combined to an individual DTD – e.g. the xDiML DTD used for Theses and Dissertations. Due to its modularity the system can easily be used to supply new publication series with appropriate DTDs, which can contain special elements and which are as slim as possible and therefore more easily applicable than a universal DTD. On the other hand, due to the re-usable modules, the generated individual DTDs are still interoperable, to a certain extend.

Fig. 3 presents the basic structure of DTDSys. DTDBase, the underlying database of the system contains the different module files including information and description on the single XML elements and attributes. DTDBase delivers an HTML file (*dependencies.html*) enclosing the necessary information of all available modules including dependency information. Using a current web browser, a user can compose an individual DTD selecting the required modules which subsequently are stored in the *selection.xml* file. On the basis of this XML file and the original DTD modules the *full-dtd.xml* is generated, containing complete information on all selected modules. Out of this file both, the actual DTD – in this case *xdiml.dtd* – and the element reference websites are generated which consist of one PHP file per XML element (e.g. *chapter.php* for the chapter element). Thus, at the push of a button, an individually adapted DTD file and its corresponding web based reference can be created.
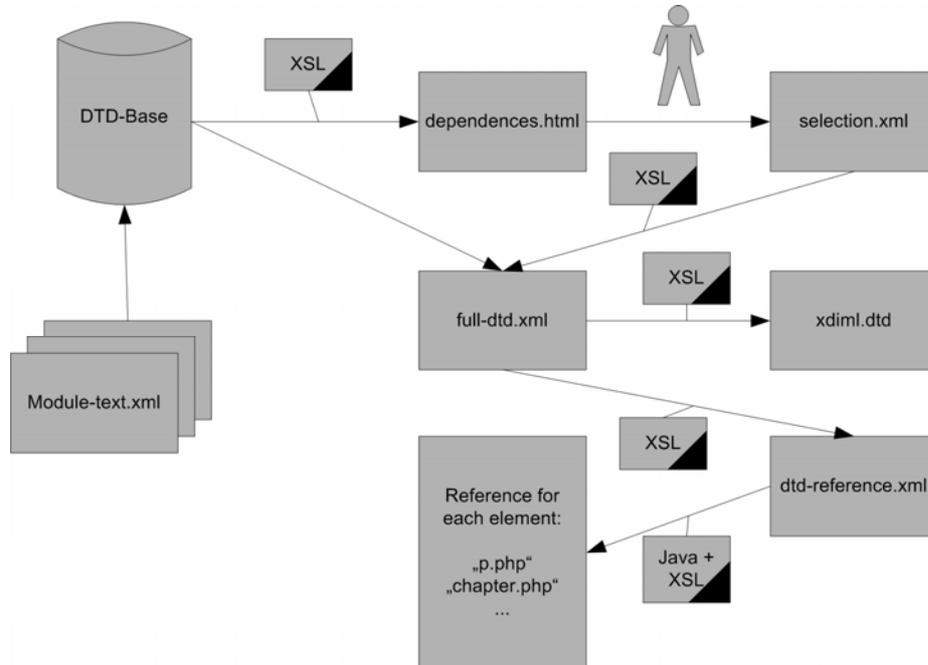
Fig. 3: DTD generation process using DTDSys

The DTDSys also facilitates the integration of externally managed (standard) DTDs such as SVG, SMIL, MathML, or MusicML and thereby allows the generation of DTDs with multimedia extensions. The use of a controlled and centrally managed set of modules provides the advantages of shared semantics beyond the borderline of different DTDs – a feature which is used e.g. for qualified full text retrieval. The XML based publishing approach is currently applied for dissertations and master theses, university series publications, as well as a few electronic journals and conference proceedings. Using *DTDSys* various other document models can be realized rather easily.

Workflow System

One of the main developments of SCOPE is the workflow system. It is – in conjunction with the PC management system mentioned two sections before – the centerpiece of the conversion and long term preservation process. Fig. 4 shows the principal functionality of the workflow system. This graphic is based on standard workflow ideas published by the Workflow Management Coalition [11].
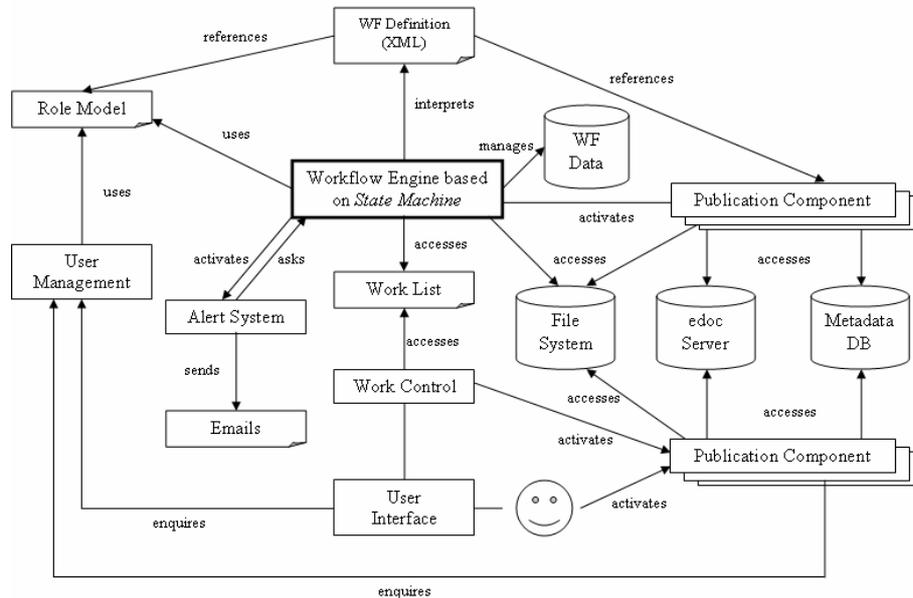
Fig. 4: Overview of the Workflow System

The workflow system itself is composed of a *state machine*, a *role model*, and a *work list*. In addition it uses the *file system* and a relational database for storing its *workflow data*. Thereby, the workflow states can be permanently saved. Storing the workflow information other than in the main memory is especially essential to handle long running processes as they are common in publishing environments.

External applications, the *publication components*, that are controlled and whose actions are initiated by the workflow system are found on the right side of Fig. 4. There are PCs which are automatically activated by the workflow engine, and there are PCs partly or completely controlled by human users of the system. The publication components are integrated into the workflow system according to their formal specification. These applications may have access to the *metadata database* and the *edoc server* or can handle user interactions.

The workflow engine is realized by an external state machine which is based on an open source project called Lucille, a project by the Austrian company XiCrypt. It is completely written in Java and allows full state and transition evaluation and control. As it is necessary in heterogeneous publishing environments the workflows can be easily modeled and altered using an XML based *workflow definition* language. This feature allows quick re-configuration of the workflow using standard XML editors. Additionally there exists a graphical editor for basic configuration. Using the workflow definition language, it is also possible to quickly incorporate new versions of PCs which is necessary for the continuous development of these tools.

The state machine controls the flow and execution of the workflow. It supports the full spectrum of possible branching and parallelism. Every step in a workflow needs to be implemented using a specific java class. If the performed actions of several steps

are alike, they can be implemented by the same java class. Each step is afterwards handled by its own class instance. These classes can implement automatic action or a user interface necessary for human interaction. As mentioned above, many steps in the process of converting somehow need human interaction, due to the fact of the manifold possible errors. We are continuously improving the tools, and some automatic error handling is already possible, but certain aspects as the correct layout of pictures and graphics cannot easily be decided by computers.

As SCOPE is mainly intended for publishers the workflow system needs to incorporate a review process. We have decided to use *GAPWorks*, the workflow system developed in the GAP project [12]. Reusing GAPWorks enables us to use a proven technology. To integrate this external workflow system it is completely encapsulated into a single step of the state machine. This step then initializes the GAPWorks workflow engine and only communicates using specific file handles with the storage system underlying the state machine. The publisher can then use GAPWorks for his review process. If the review process finishes, the GAPWorks workflow ends in a final state, terminates and writes out the result of its reviewing to the file system. Then the state machine step is reactivated and can handle the returned data.

Conclusion

With this paper we have presented a generic framework for XML based publishing processes. It is mainly based on a formal approach to describe publication components as technological parts of electronic publishing processes.

The SCOPE architecture consisting of a management system for publication components and a configurable workflow system rests upon this abstract data model. Moreover, a DTD management system has been developed to generate XML DTDs out of individually selected modules.

The seamless integration of authoring tools, transformation and conversion tools, and other technical publication components such as long term preservation and digital signature tools distinguishes the SCOPE architecture from other systems in the market, namely eprints and DSpace. The main difference to these systems is on the one hand formed by the focus on supporting the authors in the publishing and writing process through templates and other support tools. On the other hand the long-term preservation of the documents distinguishes this system from other systems. The use of XML as data format and the publication component management system support the long term preservation efforts. These technologies enable the controlled conversion to new presentation formats and styles. Additionally the usage of the GAPWorks review component with its flexibility and easy configuration allows peer-reviewing and helps to guarantee the publication of high quality works.

While the management system for publication components has been realized prototypically the workflow system is just being implemented.

References

[1] Shi, Y; Sosteric, M.; Wenker, O. (2001): The Upcoming Revolution in the Scholarly Communication System. *The Journal of Electronic Publishing*. 7 (2), December 2001. Accessed from http://www.press.umich.edu/jep/07-02/sosteric.html on February 24, 2005

[2] Budapest Open Access Initiative (BOAI): http://www.soros.org/openaccess/

[3] Crow, R. (2002): The case for institutional repositories: A SPARC position paper. Accessed from http://www.arl.org/sparc/IR/ir.html on February 24, 2005

[4] Harnad, S. (2001): The self-archiving initiative. *Nature*, 410, 1024-1025. Accessed from http://www.nature.com/nature/debates/e-access/Articles/harnad.html on February 21, 2005

[5] Berlin Declaration on Open Access to Knowledge in the Sciences and Humanities, October 2003. Accessed from http://www.zim.mpg.de/openaccess-berlin/berlindeclaration.html on February 24, 2005

[6] Chan, L. (2004): Supporting and Enhancing Scholarship in the Digital Age: The Role of Open Access Institutional Repositories. *Canadian Journal of Communication*, 29:pp. 277-300. Accessed from http://eprints.rclis.org/archive/00002590/ on February 22, 2005

[7] eprints,org, developed at the University of Southampton. Accessed from http://www.eprints.org/ on February 28, 2005

[8] DSpace. Accessed from http://dspace.org/index.html on February 28, 2005

[9] Document and Publication Server of Humboldt University Berlin – policy. Accessed from http://edoc.hu-berlin.de/e_info_en/policy.php on February 28, 2005

[10] Reference Model for an Open Archival Information System (OAIS). Blue Book. Issue 1 (January 1st, 2002). Accessed from http://ssdoo.gsfc.nasa.gov/nost/wwwclassic/documents/pdf/CCSDS-650.0-B-1.pdf on February 28, 2005

[11] Workflow Management Coalition (1995): The Workflow Reference Model. Accessed from http://www.wfmc.org/standards/docs/tc003v11.pdf on February 28, 2005

[12] GAPWorks, developed by ISN Oldenburg. Accessed from http://www.gapworks.de on February 28, 2005