Journal of Universal Computer Science, vol. 6, no. 3 (2000), 356-366
submitted: 15/9/99, accepted: 25/11/99, appeared: 28/3/00

Electronic Submission, Managing and Approval of Grant Proposals at the German Research Foundation based on Standard Internet and Office Tools

Dieter W. Fellner
(Braunschweig University of Technology, Germany
d.fellner@tu-bs.de)

Marco Zens
(Braunschweig University of Technology, Germany
m.zens@tu-bs.de)

Abstract: Today, grant proposals submitted to the German Research Foundation (DFG) are paper documents. They are received by ordinary mail, manually entered into a proprietary software system and, finally, information relevant to the specific task is extracted manually and sent to other departments involved in the reviewing/approval process. Of course, all these activities are purely hard-copy based.

This paper gives a first report on a research project "GoldenGate" which focuses on the development of a prototype system for a complete electronic workflow including submission, managing and approval of applications for research funding at the DFG. Typically one would use one of the available Information/Workflow Management Systems, but after careful consideration we made the decision to use a set of standard software tools and formats (i.e. *Hyperwave Information Server*, *MS Office'97*, *XML*) as the key components of our new system and combine them with minimal but flexible interfaces.

These ideas, the situation at the DFG, technical details of our present implementation and preliminary results are presented in this paper.
Category: H.4.1, H.5.3

1 Introduction

The German Research Foundation (*Deutsche Forschungsgemeinschaft, DFG*) — which is the largest provider of research funding in Germany — supports several thousand projects from every field of scientific research and education with an annual volume of about 1.2 billion $ US.

Up to now the DFG interacts with the outside world by paper documents only. Internally, electronic support is provided by an integrated, proprietary database system (called 'All-in-One') that can handle only ASCII-documents. This system is accessed with a VT220-type terminal emulation software, running on modern PCs with MS Windows NT which, by now, is available in almost every office.

The DFG has acknowledged the need for a new and modern system, that can handle multimedia documents, is open to external users (especially reviewers) and is flexible enough to be used by all departments. It also has to provide a modern graphical user interface (GUI), must be able to run on today's predominant office architecture (MS Windows NT) and should include the ongoing developments in the fields of workflow and document management.

This situation lead to our research project 'GoldenGate' presented in detail in [Section 3]. One main issue of this project is the new approach of using standard software components to model information and workflow management (see [Section 2]). [Section 4] gives some interesting technical details of the current prototype implementation (using *Hyperwave Information Server* and *MS Office'97*). In [Section 5], we describe some first experiences and outline the next level of functionality.

2 Information Management with Standard Software

One possible approach (which happens to be the one typically used) to replace paper documents by their electronic counterparts and to support conventional office processes with a computer system is to use one of the commercial Information Management Systems – or Workflow-/Document Management since one is most often part of the other – available today. The core of the functionality of these software packages can be described as

- access
- authentication and
- workflow.

These three features are the most important ones (besides storage) that Information Management Systems have to (and typically do) implement first, some more, some less. One cannot expect to convert traditional work processes into their digital counterparts without them. However, because people are used to the core features, they will recognize problems in these areas before getting to know or starting to appreciate the positive aspects of electronic workflows. Of course, these systems also include many of the advantages that an electronic environment can offer: the ability for several people at several places to access the same information simultaneously, the possibility to automatically build indexes or to compute results, issue queries (perhaps even full-text).

The point we make here is: any system that implements these three important core features can be used as an Information (or Workflow) Management System, even if this is not obvious or commonly agreed on by all of our colleagues.

Today's standard software components (meaning applications or systems that are widely available and also widely used, especially in office environments) are powerful enough to fulfill the task of implementing the core features of an Information Management System. A system based on these products has the advantage that their features and their flexibility can be exploited, even their file-formats can be used for data exchange. For example, there is no need to convert electronic documents from a workflow-specific format into application specific ones.

Another important point is that 'standard packages' are already in use in most offices, which means that people know how to work with them. Thus, the reuse of already existing software is, particularly in this case, a matter of cost reduction and – even more important – a matter of significnatly reducing the time to introduce the system.

We mention two special software packages here, as these are used as basic components in the 'GoldenGate' project (see [Section 3] and [Section 4]): *Microsoft Office'97* and *Hyperwave Information Server* .

The applications of MS Office'97 do not need a detailed discussion, but we stress two important points. First, all major components (i.e. *Excel*, *Word*, *PowerPoint*) are fully programmable using *Visual Basic for Applications (VBA)* and, second, they are able to act as clients and servers for other software components with the help of *DCOM* [Microsoft Corp., 1999], the Microsoft *Distributed Component Object Model*. These abilities are exploited in our prototype implementation (see [Section 4]).

The *Hyperwave Information Server* [Kappe et al., 1993, Maurer, 1996] can be described as a second generation WWW-server because it is a combination of a database and a web-server. It is based on modern concepts of database design and information retrieval as well as multimedia storage and Internet access.

Some of its features critical for the ideas described in this paper) are:

- The integrated database stores any kind of (multimedia) data-objects in one or several hierarchies of objects and collections of objects (which can contain collections themselves). In any case, the data is stored only once, several instances are modeled by reference.

  These references or 'links' are again database objects and they are secure, meaning that the moving or renaming of an object, will not invalidate the associated hyperlinks.

- Any object can be described by arbitrary metadata ('attributes'). One can build indexes over these attributes to execute boolean or ranked queries.

- Every database object is access controlled with read, write and delete rights for single users, groups or everyone, similar to UNIX file-system rights.

- All these features can be used and even administrated via a conventional WWW-browser from anywhere on the world (given the proper access rights).

- Additionally, the full functionality can be used from other programs with the help of a TCP/IP based protocol (HG-CSP [Hyperwave GmbH, 1997], Hyper-G Client-Server-Protocol).

- A full-text engine of Verity, Inc. is integrated, that allows full-text queries not only for 'readable' data, but also for popular formats like Microsoft Word "doc", Adobe Portable Document Format (PDF) [Bienz and Cohn, 1993] or Postscript [Adobe Systems Inc., 1990].

Summing up, Hyperwave is a good solution for the central component of an Information and Workflow Management System, based on standard software packages.

3 Research Project 'GoldenGate'

GoldenGate [Fellner and Zens, 1999] is a cooperative research project of the German Research Foundation and the Digital Library Lab of the Insitute of Computer Graphics at the Braunschweig University of Technology. The goal of this project is the design and prototype development of a complete electronic workflow at the DFG, that allows proposals to be submitted in electronic form by researchers and to be stored and managed (e.g. made available to reviewers) in a database until the final approval and, of course, over the full life-cycle of each project including knowledge engineering related tasks executed well past the active period of individual research projects.

Requirements for the new system are, besides a modern GUI and the use of modern paradigms from the fields of Workflow and Document Management, the ability to handle multimedia documents and the efficient and cost effective customization for different departments (with different structures and needs) of the DFG. Another important feature is the controlled access to some of the material from the 'outside', be it other departments or external reviewers from any location, without special hardware or software. Finally, existing structures, processes and data have to be integrated and the platform is prescribed as MS Windows NT running on PCs.

One problem is that the current work processes in individual DFG departments are not formally defined in a complete and consistent way. Not uncommon to large institutions, we could not find a single person who was able to define all rules and formalisms. Thus, we had to develop our system using an iterative approach. On the other hand, the advantage of this situation was that we could freely choose an architecture that is powerful as well as flexible enough to be easily adapted to any new requirement in each iteration step (see an example for one Use-case in [Fig. 4]).

Following the ideas of [Section 2] and knowing that the requirement of world-wide access can only be fulfilled with Internet technologies, the GoldenGate system consists of

- a Hyperwave Information Server as a central (though distributed) database and access point,
- MS Office applications as well as
- ordinary WWW browsers (e.g. *MS Internet Explorer*, *Netscape Navigator*,...), used as access and administration tools.

The final system will allow applicants to download (via a Web browser) a template for the favorite word-processor (LaTeX, *Adobe FrameMaker*, *Microsoft Word*, ...) which includes support and help (by 'wizards' or selection lists) when creating the grant application. The resulting (electronic) document is uploaded (again, via a Web-browser) to the GoldenGate server at the DFG, where it is converted into an internal format (i.e. XML [W3C XML WG, 1997]) and stored in a collection at a special location of the database hierarchy.

Upon arrival of the document at the DFG's server, an officer in charge is informed that a new grant application has arrived. From this application, a 'dynamic view' (a Microsoft Excel sheet) is generated automatically and also stored in the database. It is referenced from other locations within the hierarchy, e.g. according to one or more area-specific thesauri, and it is supplemented by metadata that has been automatically extracted from the application. These dynamic documents can then be checked, manipulated or passed on with Excel'97. Many features of Excel (like pop-up menus, forms or COM) do support the user in the subsequent handling of the grant proposal. Typical examples are the access of the address database (consistently maintained with the same Hyperwave server) or the selection of standard salary amounts from various lists. During the whole process, so called 'static views' (currently Adobe PDF documents, but pure text or anything else is possible) can be generated at any time to freeze a special state or to act as read-only documents for other departments or reviewers.

All the time, the document remains stored in the database hierarchy of Hyperwave. 'Free' browsing or non-standard queries to the database such as access to the DB which is not part of a dedicated module which is directly accessible through the user interface, can be done with each user's favorite Web browser. For example, looking at all proposals by applicant 'X' or at all accepted applications in research field 'Y' is possible – given the proper access rights, of course. In the case where someone someone clicks on a dynamic view to work with it, the document is checked out by assigning the read/write access to this user exclusively until it is checked in again. Hyperwave's programmable templates, its internal locking mechanisms and CGI scripts are used to implement this behavior.

A prototype of the system described above is already in use. Technical details of the current implementation will be given in the next Section.

4 Technical Details

The main components of GoldenGate are a Hyperwave Information Server, MS Excel'97 and MS Word'97 (see [Section 2]). They are combined with and accompanied by several tools and interfaces, programmed in the scripting language Python [Rossum, 1995]. Additionally the technologies HTTP, TCP/IP and COM are used. [Fig. 1] gives an overview of the current package.
As shown, everything is centered around a Hyperwave server holding the GoldenGate database. The hierarchical organization of this DB is schematically displayed in [Fig. 2]. Using Hyperwave's features (see [Section 2]), other 'virtual' hierarchies are added according to the demands of the individual DFG departments, for example, an alphabetical hierarchy of persons or a hierarchy for several research programs. As Hyperwave offers reference links to objects and collections, no data entity has to be duplicated (see, for example, the dashed arrows in [Fig. 2]).

The system is initialized with an address database and all documents relevant for currently active research grants. Both are extracted from the current system ('All-in-One'), parsed and inserted by Python scripts, using Hyperwave's native Client-Server protocol (HG-CSP). Following the concept of modularization, we implemented this task by creating an API for the HG-CSP as a C++-extension to Python that is embedded via dynamic runtime linking (a DLL).

This API is also used by other Python tools like the address dialog (see [Fig. 3]) that acts as a graphical frontend to search, edit, or update the address data. Other examples are several hierarchy browsers that are used by MS Excel and MS Word to

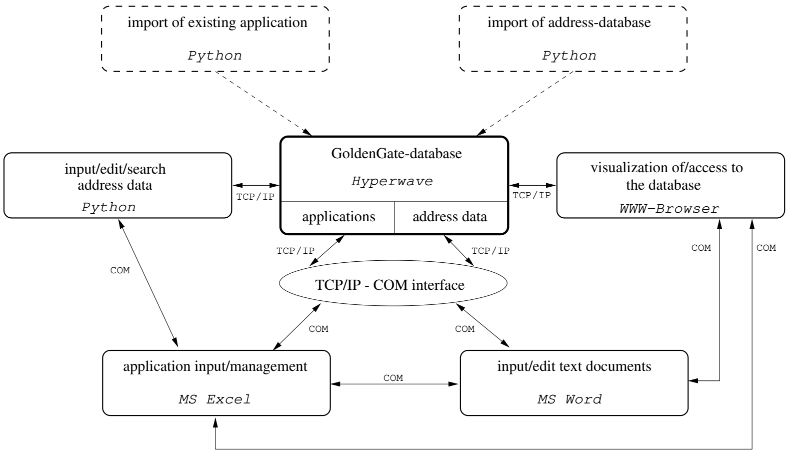

Figure 1: *The structure of the current prototype implementation of GoldenGate.*

directly access objects on the server. These tools are implemented with Mark Hammond's Win32-extensions to Python [Hammond, 1999], that, for example, allow the use of Windows/Microsoft GUI elements and system calls.

A visualization of database structures and objects is presented by Hyperwave. Thus, one can simply use any Web browser to access database objects and information (see [Fig. 3]).

Utilizing the customizable Web interface of Hyperwave, we modified Hyperwave's presentation templates (using a combination of *JavaScript* [Netscape Inc., 1999] and Hyperwave's own template language *PLACE* [Hyperwave GmbH, 1999]) to suit our needs. For example, to generate the Web-page in [Fig. 3] metadata is extracted from database objects and displayed, when the user looks at the contents of a collection.

The so called 'TCP/IP – COM Interface' plays an important role in our system. The Win32-extensions allow a Python script to act as a COM client, i.e. a software component that uses functionality (methods, data) which other components (libraries, processes, applications – so called COM servers, which can also be implemented easily with Python) offer via the COM protocol. One Python COM server is registered for the address dialog (to cater for convenient access by MS Excel) and several other servers form the 'TCP/IP – COM Interface' mentioned above. This interface offers methods, objects, and GUI dialogs to access Hyperwave. In a way, COM invocations are translated into the HG-CSP. This setting allows MS Word and MS Excel to access, load, store, and even organize documents and collections in the Hyperwave database.

After a new grant proposal has been received and converted (see below), the user creates a new Excel sheet from a special template. This template – besides including an empty default sheet and all initial structures – activates buttons and menu entries that can initiate the execution of several VBA (*Visual Basic for Applications*) modules. Upon execution of the 'import' module, the user decides which new grant proposal to process. The corresponding XML document is parsed and the newly created sheet is

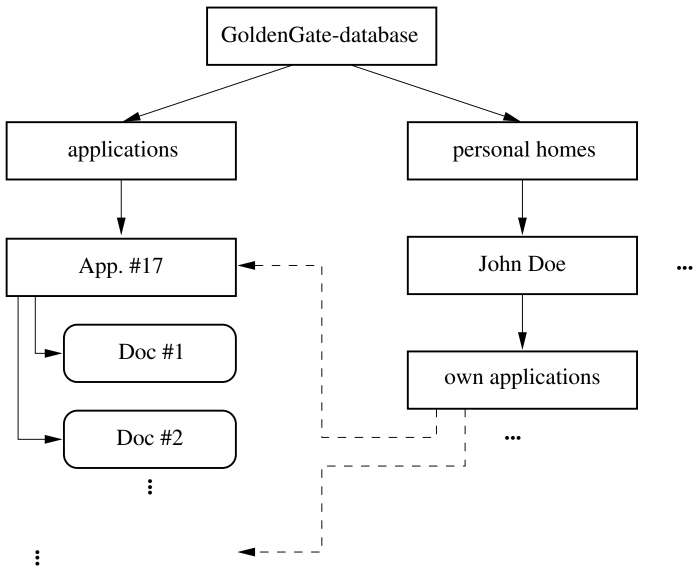

Figure 2: *A schematic model of the initial GoldenGate database hierarchy.*

filled out automatically. The user only needs to assign a new grant ID (used by DFG as an internal reference), check the plausibility and store this new 'dynamic document', as we call it, into the database. [Fig. 3] contains examples for an application and the resulting Excel sheet. The 'storage' module also builds a collection for the whole new process, references it from all respective reference collections and creates (with the help of *Adobe PDFWriter*) a 'static document' in PDF, which acts as a snapshot of the application's state. For documentation purposes this last step is repeated every time the dynamic document is saved.

Other VBA modules access Python tools via COM to access the integrated address database (see above), offer support while editing, implement a simple document history and care for the locking of documents. The current locking mechanism acts as a proof of concept how revision control and workflow management can be implemented with the features of Hyperwave like shifting of access rights and document write locks. While one user works with a dynamic document, it is locked in the database and an attribute marks it as 'in use by X'. This is also visible in the WWW-browser and is implemented by an CGI script and a VBA module, as you can open a document from Excel or the browser. Using this mechanism, we effectively prevent two people from editing the same file at the same time, which would otherwise be possible as it is downloaded from the server and uploaded later. Remember, read-only access is always possible by access to the latest 'static' document.

An important feature of the document templates for grant applications is the hidden markup (i.e. invisible 'hints' or 'tags' embedded into the document) which makes it possible to automatically and safely extract relevant data. In the MS Word'97 version

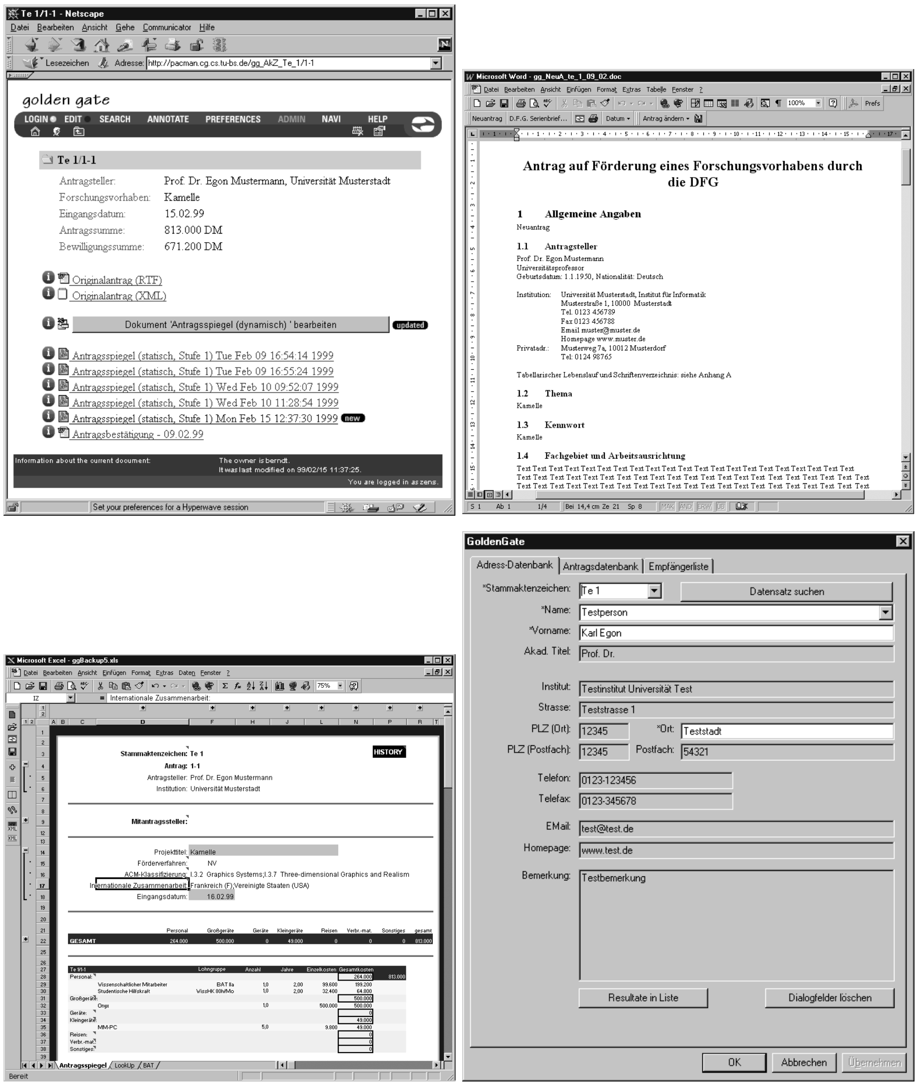

Figure 3: *Some screen shots showing components of GoldenGate at work: on the top left a view with a browser on a collection holding a grant proposal; below a dynamic view with MS Excel of the same 'document'. The top right displays the MS Word document with the grant proposal and below a (Python) tool to directly access the address database.*

of the application template – because of the lack of more adequate tools – this markup is implemented with paragraph and character styles. For example, when the applicant's name is entered, it is marked with style `ggName`. After the document is stored as RTF, the resulting file is parsed, relevant information is extracted and an XML-file is generated, which will, for example, include the text with style `ggName` as content of the tag `<ggName>`. We can then use one of several available XML-parsers to check the new application and – possibly – automatically reject it if something is missing or the structure is incorrect.

5 Preliminary Results and Future Work

A prototype of the GoldenGate system as described in the previous two sections has been implemented and is currently in use. It allows a researcher, after registering for electronic submission once, to identify himself and to download a template document for MS Word'97. This document, when opened, guides the author through the completion of the essential parts of an application form, via a sequence of dialog boxes (today often called a 'wizard'). When printed, the document has to form required by the application process for DFG funding. As an alternative, the document can be saved in RTF (*Rich Text Format*) and uploaded to the GoldenGate-server with the help of a web-form ([Fig. 4] presents a use-case diagram).

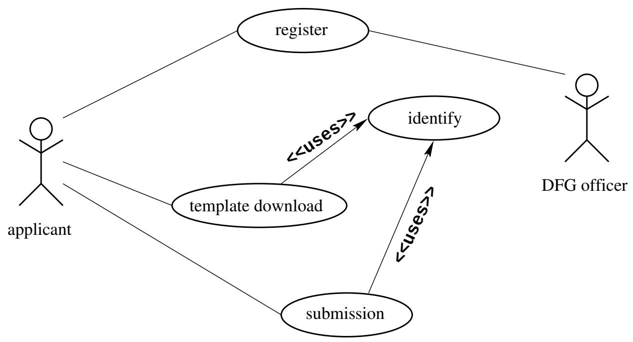

Figure 4: *An UML use-case diagram for 'electronic submission'*

On the server side, a dynamic document is created (see [Section 4]) that allows the officers in a DFG department to fulfill all typical tasks. See [Fig. 3] for screenshots of an application and the resulting Excel sheet.

During first demonstrations and iterations we learned quickly that we had to increase our system's functionality: support for standard letters (acknowledging the receipt of a proposal, asking for missing details, . . .) and mail merges, storage for internal and external documents (for example letters to the applicant) and statistical analyses of elements in the database became necessary. Particularly in this situation our ideas did prove their power and their flexibility, as we did not have to reimplement/reinvent anything. We just integrated more of the existing features of those tools we already used and only had to update some of the interface dialogues. Currently, Microsoft Word'97 is used to generate mail merges (by accessing the address database on the Hyperwave server) and to store arbitrary documents (one of the basic features of Hyperwave). For data analysis statistical components of Microsoft Excel (tables, bar graphs, pie charts...) are used. Data has only to be grabbed from the attributes of the database hierarchies (which, of course, is done with the help of another Python COM server).

As a concurrent result this improvement process made one of DFG's currently used software packages superfluous. Its whole task was to create mail merges. With the help of our approach several thousand lines of code were replaced by something that was already there and just had to be used.

The current system has proven itself in several demonstrations and evaluations to work efficiently and reliably. For example, it was used successfully to handle the submissions for Phase II of the strategic DFG research initiative on Digital Libraries (V^3D^2)[1]. In addition to the one division which has already made a decision in favor of our system, several other departments at the German Research Foundation are planning to introduce it, as they are convinced that the amount of routine work can be significantly reduced with the help of GoldenGate.

At the moment we are still learning and analyzing what all of the conventional processes and workflows at DFG really look like. Since these competencies were never written down formally and nobody really knows all of them, we had, and still have, to use an iterative approach. Whenever we implement something new, people at the DFG have new ideas and discover new possibilities how GoldenGate could help them. Up to now, we have been able to realize all these with a minimum amount of implementation work.

The most important lesson learned during this ongoing project is that because of our approach to combine standard (office) tools with lightweight interfaces, the gain of flexibility goes hand-in-hand with a reduction of effort (time and money) for maintenance and adaption to new environments or needs, as compared to proprietary or existing commercial solutions. While the complexity of the implementation phase is similar to other systems, we have the advantage that the actual amount of 'code' (including templates and interfaces) we have to maintain or to update is only a small portion of the whole product. The by far larger part (Hyperwave Information Server, MS Office'97) is maintained and improved by other providers.

Our future tasks are the extension of the system to cover more steps in the fields of reviewing and approval of grant proposals. So far, we see no problems to integrate them using the techniques and ideas that have proven to work (see above). Another very important point is additional support for the applicants, e.g. templates for other common word-processing systems like LaTeX or Adobe FrameMaker, which we already have and which will be integrated into the system as a next step.

Once the described system is fully operational, an obvious extension is the reuse of the available information in the context of data mining as well as cross-media publishing. As all grant proposals (and additional material documenting the applicant's research competence) are electronically available in well-structured and well-defined

[1] `http://www.cg.cs.tu-bs.de/V3D2`

standard formats (e.g. XML, PDF), relevant information can be extracted automatically and new ‘documents’ such as annual reports or overview articles for selected research areas can be created with a minimum amount of additional work. Futhermore, these documents can easily be produced for potentially different and varying media such as CD-ROMs or WWW contents.

Acknowledgements

We would like to thank our partners at the German Research Foundation and all members of the Digital Library Group at the Braunschweig University of Technology, especially R. Berndt and C. Oberscheid, for their support. Special thanks go to A. Engelke, DFG II D 6, for initiating the project and playing a key role in the design and to U. Drewen, DFG III N for providing a real-life evaluation framework for GoldenGate.

References

[Adobe Systems Inc., 1990] Adobe Systems Inc. (1990). *PostScript Language Reference Manual*. Addison-Wesley, 2 edition.

[Bienz and Cohn, 1993] Bienz, T. and Cohn, R. (1993). *Portable Document Format Reference Manual, Adobe Systems Incorporated*. Addison-Wesley.

[Fellner and Zens, 1999] Fellner, D. and Zens, M. (1999). Research Project GoldenGate. http://www.cg.cs.tu-bs.de/research/projects/gg.

[Hammond, 1999] Hammond, M. (1999). Win32 Extensions to Python. http://www.python.org/windows/win32all.

[Hyperwave GmbH, 1997] Hyperwave GmbH (1997). Hyper-G Client-Server Protocol 7.17. http://www.hyperwave.de/7.17-hg-prot.

[Hyperwave GmbH, 1999] Hyperwave GmbH (1999). Hyperwave Programmer's Guide. http://www.hyperwave.de/program.

[Kappe et al., 1993] Kappe, F., Maurer, H., and Sherbakov, N. (1993). Hyper-G: A universal hypermedia system. *Journal of Educational Multimedia and Hypermedia*, 2(1):39–66.

[Maurer, 1996] Maurer, H., editor (1996). *Hyper-G/now Hyperwave – The Next Generation Web Solution*. Addison-Wesley, Harlow, England.

[Microsoft Corp., 1999] Microsoft Corp. (1999). The Component Object Model: Technical Overview. http://www.microsoft.com/com.

[Netscape Inc., 1999] Netscape Inc. (1999). Client-side javascript reference. http://developer.netscape.com/docs/manuals/js/client/jsref/index.htm.

[Rossum, 1995] Rossum, G. v. (1995). Python reference manual. Technical Report 25, CWI, National Research Institute for Mathematics and Computer Science in the Netherlands.

[W3C XML WG, 1997] W3C XML WG (1997). Extensible Markup Language (XML) 1.0. http://www.w3.org/XML.