Constructing Digital Library Interfaces

David M. Nichols and David Bainbridge
Department of Computer Science
University of Waikato
Hamilton, New Zealand
+64 7 8585130
{dmn, davidb}@cs.waikato.ac.nz

Michael B. Twidale
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
IL 61820, USA.
+1 217 333-3280
twidale@uiuc.edu

ABSTRACT
The software technologies used to create web interfaces for digital libraries are discussed using examples from Greenstone 3.

Categories and Subject Descriptors
H.3.7 [**Information storage and retrieval**]: Digital Libraries – *user issues.* K.3.2 [**Computers and Education**]: Computer and Information Science Education – *information systems education.*

General Terms
Human Factors.

Keywords
XSLT, interfaces, computational sense, digital library education

1. INTRODUCTION
The use of software to allow library science students to practically create online repositories is an important element of a digital library (DL) curriculum. When current DL software is used in an educational context it can highlight both good (e.g. workflow support, platform independence) and bad (e.g. module complexity, lack of 'rollback') features [3]. In this paper we highlight issues associated with interface creation for DLs and present the new approaches adopted in Greenstone 3.

2. BACKGROUND
The limited evidence available in the literature suggests that many collection creators encounter problems in learning to effectively use digital library software [3,6,7]. An aspect of Greenstone use that is known to be problematic is the customisation of the DL interface [3,8].

In an analysis of activity on a sample of the Greenstone-users mailing list McCurdy [2] found that topics related to interface design issues were common. Messages about 'customisation' (23%) and 'configuration' (17%) were two of the top three primary subject areas (the other was 'functionality' at 23%).

The two main methods used for interface construction in Greenstone 2 are 'format statements' (for document surrogates) and 'macros' (for page structuring). To understand more about Greenstone users' experiences with these methods a simple online survey was publicised via mailing lists and contacts in institutions known to use Greenstone in DL education.

26 responses have been received so far and two main themes emerged. The Greenstone Librarian Interface (GLI) [8] is reported to be a useful tool for managing workflow and a generally supportive environment … apart from the features for interface customisation (e.g. 'the librarian interface is easy to use in every respect but formatting features'). Frustration is expressed with both 'format statements' and 'macros': 'I spent far more time trying to customize our interface than I did adding content to our library', 'it's far too difficult for the average user to create a custom interface, an important and desirable feature for any digital library', 'interface design is currently geared very much toward programmers' and 'format statements are overly complex for most librarians'. Several responses favorably mentioned the 'drag and drop' interface construction used in Microsoft's Visual Studio.

Interestingly a smaller group of responses were largely content with current functionality ('macrofiles and format statements are powerful and easy to use once you get to know them') and in fact some wanted 'enhanced conditional statements'. Interviews of project members at Waikato who had conducted Greenstone workshops reinforced these two main themes. Simply, users wish to easily produce DL interfaces made of clean accessible HTML.

Simple heuristic analysis of the formatting elements of the GLI highlights the: unique syntax of conditional statements, small text area for format statements, lack of editing support for error prevention (e.g. HTML syntax tag highlighting), limited power (no looping constructs) and that some HTML output is derived from a C++ program that can't be edited without re-compilation. Macros are separate from the GLI, expressed in a further unique syntax and are found in various places in the server's file system.

The combination of evidence from the literature [6], student feedback [3], interviews of those who have run workshops, heuristic interface evaluation, responses to the online survey, the mailing list analysis and anecdotal feedback to project members, all suggest that interface customisation is an important topic that is not currently well-supported in Greenstone 2. Although these issues affect all users we believe they have a disproportionate effect on those learning about DL software.

3. GREENSTONE 3 IMPLEMENTATION
The Manakin/DSpace project [4] and Greenstone 3 have both chosen XSLT as a key internal technology. In Greenstone this decision was based on internal software engineering considerations and a desire for standardization, based on experiences with the ad-hoc development of 'macros'. However,

http://doi.acm.org/10.1145/1255175.1255240

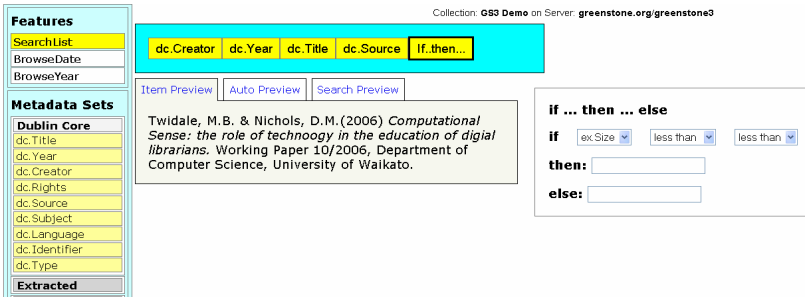

Figure 1. Prototype direct manipulation interface for interface customisation in Greenstone 3

the actual user interactions needed to create and customise end-user interfaces have, so far, been left under-specified.

It is possible to use XSLT as the sole customizing technology though the only evidence of its learnability suggests that it may be a high barrier to entry: "one archivist noted that “learning XSLT … has been a major block to implementing EAD at our institution"" [5] and "XSLT is an exciting and powerful language, but not an easy one to get to grips with" [1 (pg 2)].

To address the complexity of raw XLST, Greenstone 3 contains a second layer of abstraction (a 'gsf' namespace) that more closely maps onto library science concepts and the methods used in Greenstone 2. For example, a `<gsf:metadata name='dc.Title'/>` element is used to represent a more complex XLST `value-of` element for metadata retrieval and a `<gsf:link>` element is used to hide the mechanics of URL path construction. Greenstone 3 maps `gsf` elements onto more complex XSLT expressions which generate the final HTML.

This approach lowers the barrier to entry for XSLT authoring in Greenstone 3 whilst still leaving the power of full XSLT available to those with greater technical skills. Interestingly, although developed independently this tiered approach mirrors that of Manakin [4]. Furthermore, for applications where learnability is crucial, such as in DL education, we are developing a *further* abstraction layer to aid in interface customisation.

3.1 Tools for Constructing DL Interfaces

The technological options presented to users (to customise their DLs) should allow for different levels of technical expertise. For programmers an API is often sufficient, whereas for experienced digital librarians an XSLT approach might be appropriate. However, for many users, especially those taking a first class in DLs, a more supportive environment, that better aligns with their skill-set, will be more effective [6].

The approach described here trades power for (initial) simplicity, reducing the amount of technical information that has to be learned before anything productive can be achieved. This approach parallels earlier work on the development of specific applications that aim to make certain functionalities available to less technically-skilled users. An example would be how spreadsheets were developed to allow end-users to utilise computers to make complex arithmetic computations *without* needing to learn a programming language [7].

Our assessment of the complexity of XSLT authoring has led us to develop a prototype Greenstone 3 AJAX environment for the formatting of document surrogates (Figure 1). The system provides explicit listings of available metadata elements (left hand side), a drag and drop interface for surrogate formatting (top), instant previewing of editing changes (bottom) and supportive templates for programming language constructs (right hand side, showing a conditional statement). We have adopted the instant preview feature of spreadsheets to encourage learning through experimentation. The web application outputs a Greenstone 3 surrogate format at the `gsf` namespace level before further server-side XLST processing into HTML.

4. CONCLUSION

Although still in prototype stage Figure 1 illustrates the style of interface interaction we believe is necessary to align with the level of 'computational sense' [7] of many of those learning about DLs. This topic is currently under-represented in the DL literature [6]; in particular, although usability studies have been performed on *searching* in DLs, they need to be undertaken for DL *creation*.

5. REFERENCES

[1] Kay, M. *XSLT Programmer's Reference*, 2nd Edition. Wrox Press Ltd., Birmingham, UK. 2001.

[2] McCurdy, R.M. *Technical Support for Open Source Library Systems: a content analysis of Koha and Greenstone email discussion lists*, MLIS Thesis, School of Information Management, Victoria University of Wellington. June 2006.

[3] Nichols, D. M., Bainbridge, D., Downie, J. S., & Twidale, M. B. Learning by building digital libraries. *Proceedings of JCDL '06*. 185-186. ACM Press. 2006.

[4] Phillips, S., C. Green, J. Leggett, A. Maslov, A. Mikeal, & Surratt, B. *Manakin Developer's Guide: The Second Version of the DSpace XML UI Project*. Texas A&M University Library. 2005. Retrieved February 2, 2007 from http://di.tamu.edu/projects/xmlui/manakin/resources/DevelopersGuide.pdf.

[5] Prom, C.J. The EAD Cookbook: A survey and usability study. *American Archivis*t 65, 2 (2002) 257-275.

[6] Suleman, H., Marsden, G. & Feng, F. Customising Interfaces to Service-Oriented Digital Library Systems, *Proceedings of ICADL 2006.* 503-506. Springer. 2006.

[7] Twidale, M.B. & Nichols, D.M. *Computational Sense: the role of technology in the education of digital librarians*. Working Paper 10/2006, Department of Computer Science, University of Waikato. 2006.

[8] Witten, I.H. & Bainbridge, D. *How to build a digital library*. Morgan Kaufmann, San Francisco, CA, 2003.