A Generic Alerting Service for Digital Libraries

George Buchanan
UCL Interaction Centre
31/32 Alfred Place
London, United Kingdom
g.buchanan@cs.ucl.ac.uk

Annika Hinze
Department for Computer Science
University of Waikato
Hamilton, New Zealand
a.hinze@cs.waikato.ac.nz

ABSTRACT

Users of modern digital libraries (DLs) can keep themselves up-to-date by searching and browsing their favorite collections, or more conveniently by resorting to an alerting service. The alerting service notifies its clients about new or changed documents. Proprietary and mediating alerting services fail to fluidly integrate information from differing collections. So far, no sophisticated service has been proposed that is integrated with the digital library software and covers heterogeneous and distributed collections. This paper analyses the conceptual requirements of this much-sought after service for digital libraries. We demonstrate that the differing concepts of digital libraries and its underlying technical design has extensive influence (a) the expectations, needs and interests of users regarding an alerting service, and (b) on the technical possibilities of the implementation of the service. Our findings will show that the range of issues surrounding alerting services for digital libraries, their design and use is greater than one may anticipate. We also show that, conversely, the requirements for an alerting service have considerable impact on the concepts of DL design. Our findings should be of interest for librarians as well as system designers. We highlight and discuss the far-reaching implications for the design of, and interaction with, libraries. This paper discusses the lessons learned from building such a distributed alerting service. We present our prototype implementation as a proof-of-concept for an alerting service for open DL software.

Categories and Subject Descriptors

H.3.4 [**Information storage and retrieval**]: Systems and Software—*User profiles and alert services*; H.3.3 [**Information storage and retrieval**]: Information Search and Retrieval—*Information filtering*

Keywords

alerting, publish/subscribe, digital libraries

1. INTRODUCTION

Active information seeking is often followed by the task of monitoring useful sources for new, or changed, information on the same topic. Users of modern digital libraries can keep themselves up-to-date by repeatedly searching and browsing their favorite collections (see dashed lines in Fig. 1, left). The Hermes [10] and Dias [15] project have shown that an alerting service which integrates information from a wide variety of information providers can be indispensable to users: It relieves them from the tedious and cumbersome tasks of searching and browsing, or even from subscribing to individual alert services such as Springer Link Alert[1], Elsevier Contents Direct[2], or ACM Table-of-Contents Alerts[3] (see solid lines to independent DLs in Fig. 1, left). Both Hermes and Dias are independent mediating alerting services for digital libraries which support the filtering and notification of event messages regarding documents stored in digital libraries. These alerting services act as brokers between the digital libraries and their users (see Fig. 1, right).

This mediating approach has shortcomings on both the providers' sides and on the users' sides: (1) On the provider side, this approach requires the providers of digital libraries and publishing houses to both provide the data and also to agree on a common protocol for alerting. Hermes attempted to overcome these problem by employing observers and wrappers for providers. Their experience has shown that it is a tedious and ongoing task to adapt the service for the changing interfaces of the publishers. (2) On the user side, the interaction with the alerting service is separated from the interaction with the digital library. That is, the alerting service is a complementary service to the digital library, which is inhibiting a fluid user interaction with all the library services: Consider a user who retrieves a document from their favorite collection; the user cannot readily turn their regular search into an alerting query because the services are not integrated. The flow of interaction is interrupted and the mediating alerting service might not even support the same retrieval methods as the DL search engine.

We propose to integrate the alerting service into the digital library software as a complementary service to information seeking in the digital library (see Fig. 2). We see alerting as the natural extension of the information seeking paradigm into temporally-continuous information access. Consequently, we propose to tightly integrate a distributed alerting service within the software that manages

[1]available via http://springerlink.metapress.com/
[2]http://www.contentsdirect.elsevier.com/
[3]available via http://portal.acm.org

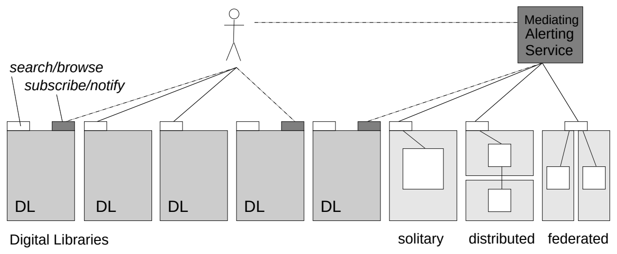

Figure 1: Information access in DL

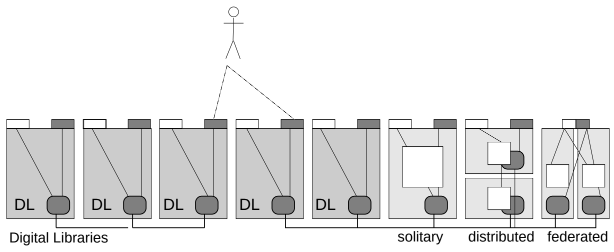

Figure 2: Proposed integrated alerting in DLs

a digital library. Our approach leads to a more homogenous (and less interrupted) user experience while accessing the documents using the two different retrieval paradigms of information seeking and alerting. The following technical advantages can also be achieved: immediate internal access to the document-related data without problems in access rights and observation of document changes, and reuse of the information seeking functionality offered by the library.

This design approach creates new challenges and opportunities that are neither addressed by existing digital libraries (and their alerting services), mediating alerting services such as Hermes and Dias, nor independent event-notification systems. In particular, the challenges addressed are to (1) build a distributed system for alerting in digital libraries integrated with the DL software and collections; (2) support distributed as well as local collections that are federated under homogenous access points; and (3) create fluent user access to documents independent of access paradigm (seeking or alerting), i.e., alerting equivalent of searching and browsing.

In the next section, we introduce some basic terminology and key concepts regarding alerting services. In Section 3 we explore user scenarios for alerting in digital libraries so that we may discover opportunities that an alerting service may hold for designers and users of a digital library. Based on the scenarios, we present in Section 5 the results of our requirement analysis of different types of libraries, their users, and the offered services. In the following section (Section 6), we propose the architecture and design for an open alerting service for DLs. As a proof of concept, we describe our prototypical implementation of a distributed alerting service that is integrated into the Greenstone Digital Library software in Section 7. A comparison of our design proposal and the prototype with existing work is presented in Section 8. In Section 9, we discuss our findings and lessons learned regarding the design of alerting services for digital libraries , which also impact the design of the digital library software. We summarize this paper in the last section.

2. ALERTING TERMINOLOGY

In this section, we briefly introduce some terminology to assist the reader in understanding the rest of the paper. The concept of *alerting* has a number of names, such as selective dissemination of information (SDI) [17], information filtering [2], or event notification system (for details see [12]).

An *event* occurs when a change occurs in a digital library. This results in an *event message* being sent that carries the information about the event. Often, alerts are generated during the library's building cycle (when a number of new documents are added and indexed in the library). User's interests are represented by *profiles*. Profiles may have different structures, such as queries, or keywords. The alerting service stores and indexes the profiles in a profile repository. Changes in the library ar either automatically reported to the alerting service, or they have to be observed (i.e., passive and active *observer*).

Incoming event messages are filtered against the stored profiles. We distinguish subject-based and content-based filtering. In subject-based filters, the user profiles contain topics or subjects (often selected from a list); event messages are grouped into these subjects. Content-based filtering analyses the content of the event messages; the filtering is more sophisticated and may be better tailored to each users needs.

A *filtering* engine executes profiles on a stream of incoming event messages. When a match is found by the filtering engine, it passes the matched event message to a *notifier* which, obtaining the user's preferred method of receiving the message from the matching profile, sends a *notification* to the user.

3. USER SCENARIOS

We present a representative selection of use cases (UCs) to highlight the variety of purposes that notifications could serve. These scenarios were developed by eliciting user requirements through literature analysis (focussing on DL user requirements), claims analysis [4] (addressing DL users and librarians), and a questionnaire [18] send to a DL mailing list (reaching mainly developers, also users and administrators). Our user study for alerting is the first one that addresses a wide range of groups that interact with, and are affected by, digital libraries. This provided a previously untapped scope for the use of alerting technologies in DLs.

UC1: Recommending Texts A professor at a university keeps a list of recommended texts for students on her ICT courses. When the university library catalog is updated, she wants to be notified to ensure that her recommendations are up-to-date. It is critical to her that when a new edition of an existing text becomes available, or when a reference document is no longer available through the library that she learns of this at once. She also wants to ensure that when a new ICT book arrives in the library, that she hears of this within a week so that she is able to advise students of its value.

UC2: Changing documents A computer science student at a university keeps track of the arrival of Java programming books in the university library. However, he also uses a number of open source programs whose documentation is kept on a free-access independent digital library. He wants to ensure that when changes are made to key parts of the documentation of his database management software of choice, he finds out as soon as possible.

UC3: Music A music lover is interested in music of a certain period. In her personal DL, she collects different versions or recordings of the same pieces. She is interested in being notified once the library collections add some new music pieces, or documents about the music. She has access to a Low-Fi recording of a life broadcast and wants to be notified when a high quality version becomes available. For some older music pieces, she has bitmap scans of the sheet music; she is interested in acquiring the same piece in a computer-readable format. Once a new DL is founded that also offers music from the period she is interested in, she would like to be notified about its collections.

UC4: Direct ingest E-Garden, a publisher of electronic gardening books supplies some of its works direct into the catalog of a public library using an awareness service [8]. The librarian who is responsible for the cataloging of horticultural items, wishes to be notified when new books arrive so that he can fully categorize the new document into the library's own scheme. A patron [11] of the library also likes to track new publications from E-Garden, to keep the recommendations list she publishes in the library up-to-date. If a publisher withdraws a book, the librarian will have to introduce a placeholder for the deleted work. The patron also wants to be notified when there are any changes in the topic classifications for *horticulture* of the DL.

UC5: System administration Both the administrator of the DL and a local software developer for the library are interested in new version of the DL software being released by the DL-software publisher. The software is open source and therefore the software developer would like to know about new bugs encountered in a new version she is working on with other programmers. She also likes to know about new libraries installing her software to see how much it is used.

As can already be seen from these simple use cases, alerts can be caused not only by changes to documents (UC1), but also other parts of the digital library such as classifications (UC4) and the underlying software (UC5). In order to address all facets of the use cases, we have to extend our model for digital libraries in the light of alerting services.

4. MODEL OF DIGITAL LIBRARIES

In this section, we introduce our model of digital libraries and its services in the light of alerting services.

4.1 Types of Libraries

As library users are not uniform, neither are libraries themselves. Goals and objectives vary due to management, history, funding, role and many other factors. We will show that this has influence on the DL's alerting service.

One distinctive form of library is the *archival library*, which seeks to preserve documents for posterity. Documents may be chosen not for their current interest, but for their potential interest in the future. In such institutions, destroying or removing documents from the library is a rare and undesirable event. In addition to new documents, superseding versions of existing documents, such as a new edition of a book, will be added to the corpus of the library. Often, existing stocks are large and funding is substantial. However, such libraries are, if famous, relatively rare. Use cases that may apply here are UC1 and UC5.

In contrast, *reference libraries* seek to provide information of immediate relevance, and documents of declining importance are soon discarded. Preservation is an additional cost only met in cases of high demand. The library will often keep only the latest edition of documents for immediate reference. Reference libraries are small or mid sized with often highly specialized collections. [UC1, UC4, and UC5]

A sub-class of the reference library is the *small specialized library*. These libraries are often provided by enthusiastic experts of a certain field, who also act as patrons of the collections. Work practices that are commonplace in traditional libraries may not be done at all. An example collection could be the documentation for an open software project. Here, new versions of documents, such as the latest version of the documentation, replace older ones; documents may change while published. Documents may undergo structural changes, caused, e.g., by group authoring. Specialized libraries are numerous, may be short-lived and idiosyncratic when compared to standard library practice. [UC2, UC3, and UC5]

Clearly, many libraries are situated somewhere between these extremes. The varied forms of library will give rise to differing user interests and profiles.

4.2 Document Identity

To provide an alerting service, each document in a library must have a consistent identifier over time. Some web-based DL systems do not provide consistent identifiers for content that are honored across different user sessions. Such systems cannot, therefore, be used as a basis for providing an effective alerting service.

The issue of identity also emerges in a second manner when a document is changed [8]. A new version of a document could be assigned a new identifier while the old one retains its identifier. Alternatively, the new document may take the identify of the previous version, while the old document is archived (and obtains a new identifier) or deleted. In addition, both the new and old version could be assigned new identifiers. These problem are even more acute if the digital library and its collections are distributed. To the library system, it may not be clear that two versions of the same document are handled.

Conceptually, librarians abstract beyond a particular edition or format by identifying the *work* of the author. For example, a new edition or an audio recording of a book are only further variations on the one work. So far, none of the widely accepted DL systems has a strong model of the work.

4.3 Distributed and federated Libraries

Digital libraries can be solitary (stand-alone), distributed, federated, or even a hybrid of these (see Figures 1 and 2). In a stand-alone DL, providing an alerting service is relatively straightforward. When a library is distributed across a number of separate DL servers, or is federated alerting becomes more complex. The library needs to be aware of changes in its constituent servers, and to inform its users of those changes. A single server's content may be used by more than one distributed or federated service.

Another difficulty that arises for distributed library systems is the inconsistent availability of parts of the library. However reliable the network and the servers are, it is inevitable that at some point part of the library is unavailable whilst the rest can still be accessed. This also affects performance and design of an alerting service in such contexts.

5. CONCEPTUAL DESIGN DESIDERATA

In this section, we discuss considerations for the design of an alerting system as a component in a digital library. These considerations emerge from analysis of our scenarios (see Sect. 3) as well as organizational and technical considerations (see Sect. 4). For an overview see Table 1.

5.1 Library Notification Services

As interactive information seeking may be achieved by more than one method, e.g. searching and browsing, alerting users of changes in content can be achieved in many ways.

Patrons could support alerting in a library by providing default profiles for particular interests in the library (*UC4*). They could also provide commentaries on new books (not so dissimilar to our Professor in *UC1* - such recommendations could themselves be passed as events through the alerting service. Thus, an alerting service may be used by librarians and patrons to notify readers of other changes in the library not directly connected to index and content changes. Examples here could include advance notification before changes actually occur (e.g. that a book will be deleted in a month) or service notices (e.g. temporary closures).

Bringing new content to the attention of readers is a common task in a physical library. To match this need, the *accession shelf* or *acquisition section* is used to highlight new material that the library has obtained. An alerting service could provide an analog of this - a part of the classification hierarchy that lists recently added documents. A well personalized notification provides a similar service to the personal recommendations given by a librarian.

5.2 User Model

We identified five different user groups and more than thirty types of events. Table 1 gives an overview, numbers in parenthesis in indicate richer substructures about which we do not go into detail here. Interest in particular event types are strongly related to the work of particular user groups. There are also technical consequences from the interaction between the event types listed above and the types of libraries introduced in Sect. 4.1.

For instance, a new document version in an archival library will take on a separate identity whereas in a specialized library, it might overwrite the old version. Consequently, the user profiles will have to be defined following different patterns. Classifications will typically change progressively and slowly over time, but in small, specialised libraries run by administrators with little technical librarianship skill they might change more dramatically.

A rich conceptual design for an alerting service will permit these different characteristics, but where an implementation is created, emphasis may be given to particular forms of event over others. Finally, for the users, clarity and support is needed bout the supported patterns of alerting.

Concept	Details
Providers	librarians and patrons, local DL, distributed and federated DL, external sources
Notifications	Personal notifications, announcements within DL (accession shelf)
User groups	reader, librarian, patron, DL administrator, DL software developer
Event types	documents: new, changed, deleted, unchanged document for given time, new metadata value, changed metadata, deleted metadata value, new metadata field, deleted metadata field, new collection, deleted collection, collection rebuilt, new classification, deleted classification, category events (4), index events (2), new library, closed library, library installation events (6), library system (software) change (4),
Support	typical predefined profiles, patron-supported profiles, integration with DL retrieval services

Table 1: Conceptual Design Desiderata

5.3 Provider Network

Providers of library-related event notification can be the local DL where the alerting service resides, distributed DLs, external publishers, and other services (such as Hermes). Supporting different types of providers requires active and passive observation of events, i.e., direct observation of the DL content versus filtering of external event messages.

The distributed alerting network consists of a number of *nodes*. Each node can be any of the following: libraries originating event messages, hubs that pass communication between nodes, services that digest multiple event streams into one, and notification services that finally transmit information to users. Any single node in the distributed network may perform one, many or all of these functions. Originators of event messages, such as digital libraries and other alerting services are referred to a providers.

The issue of mediating alert systems such as Hermes have already been addressed in other papers. In our design, we focus on the origination of messages, and their distribution within one conceptual library (distributed or federated).

One complication that arises in the context of distribution is that the messages that thus pass from one library to another should avoid duplication. In the context of event research, the accepted approach is to build a acyclic overlay network of alerting nodes, so that no messaging loops occur, and that duplicates are avoided.

If all messages are held until a server becomes available again, then substantial storage may have to be retained (depending on the number of alerts passing across the network).

Since alert messages can be received by the system from more than one library server, the question as to the location and distribution of profiles and event messages arrives. This interacts with the question of network reliability. In

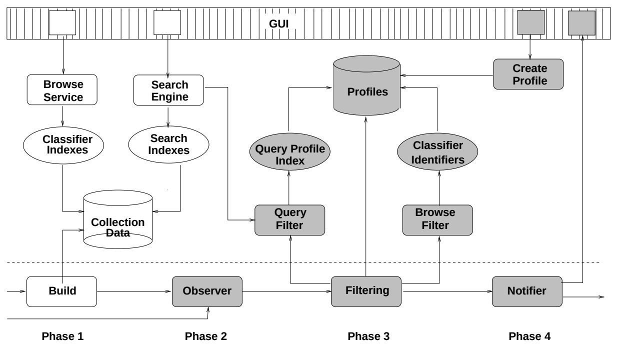

Figure 3: Hybrid Alerting Architecture. DL architecture (white) with hybrid alerting architecture (gray)

the context of digital libraries, networks are often segmented and dynamic. This raises the problem of dangling profiles and false positives. Dangling profiles are profiles that were distributed to other servers and that are now left behind in a network sector that is disconnected. When the profile is cancelled, the user would still receive notifications for events they are no longer interested in (false positives). Users should also not be required to subscribe identical filter profiles individually to a number of separate servers: errors may arise and the user faced with an undue degree of repetitive labor.

6. DESIGN AND ARCHITECTURE

We present our final design and architecture in two parts. First, we focus upon the alerting service structure on a local digital library. Second, we address the design of the distributed and federated part of the system. Finally, a summary of the design and its features will be given.

6.1 Local Alerting

We propose a general alerting architecture at the local library level, presented in Fig. 3. The existing components of a typical digital library are with a white background: components for *building* a library, the library repository for the *collection data*, indexes for classification and search, and the search and browse serves that can be accessed via a GUI.

As explained earlier, alerts regarding events in a local library, e.g., changes to documents, start with the (re-)build of the library collection (Phase 1). The following three phases (under the dashed line) are added components of the alerting service. This four-phase process matches the accepted template for event notification work-flow presented in [13]: Generation (in our case, rebuilding the library index), Observation, Filtering and Notification. All components of the alerting service are shown in gray. Above the dashed line, we show the auxiliary components and tasks that support the alerting process. Users can create profiles, which are indexed and stored in the profile repository. The filtering uses the profile indexes and the search functions provided by the digital library. In our design, we use components of the digital library's existing services to complement and support specialised alerting components. The filters permit features such as search of the full text of a document or search in a piece of music to be used without reimprisoning complex code. Given the relative similarity of DL protocols for services such as searching [1] and their small number, this approach can readily be applied to a number of popular DL systems.

We now study the key components of the local alerting process in detail: Profile creation, Observation, Filtering and Notification.

Profile Creation. The architecture which we present in this paper does not determine how accurate profiles are created that well match the interests and preferences of the user. Profiles for a user may be created directly by the user; through automatic generation based on user interaction histories; adoption of pre-defined profiles created by librarians or patrons, etc. One approach to ease the creation of user profiles by users themselves is to minimize the number of new concepts which a user has to learn: the transformation of successful search queries into profiles would support this goal. By maintaining the underlying principles of searching and browsing, as in the case of routing [2] profile creation could be simplified.

The structure of the auxiliary alerting components is essentially the same as that for the elements of a digital library. Browsing and searching both find analogs in the profiling subsystem, though storage and indexing are done in different ways. Just as a user can browse through a digital library, eventually arriving at a particular document or classification, a profile represents the same task continuing over a temporal stream of information. Thus, for example, browsing a single document becomes, as a profile, a monitor for any changes in that document. Similarly, a query becomes a monitor for any new matching documents.

However, a query profile needs to be handled differently to a monitor on a document or classification. As classifications and documents are explicit objects in a library, monitors on them are relatively easy to implement. However, as queries do not exist as separate objects, changes are more difficult to capture, and commonly matching document identifiers for the search against known changed or new documents is the easiest way of implementing the search. A profile provided by a user could be divided into several separate, simple, profiles in the alerting service, only being recombined through the central filtering task.

The profiles stored in the alerting service are inserted into the profiles database through a profile creation process. In our example, new profiles are supplied through the DL interface.

Observation of Events. The observer process is the gateway to receiving event messages for filtering and, eventually, delivery through the alerting service. Conceptually, event messages can be received from the DL's Build process or from other (external) sources. Where event messages arrive in different formats, the Observer phase will forward them to the Filtering process in an understood format for matching, translating if required. The observer could also monitor the content of the DL and create the event message.

An effective observer can receive event messages in different formats from different sources. For example, a remote DL host with Z39.50 could send messages to the observer from a saved search. However, the Observer must be programmed to receive each event message format.

The problem of versioning has been described before: Such problems vary between libraries, but they are endemic where there is weak support for higher level concepts than the document itself.

Filtering of Events. The operation of the filtering process is the keystone of the alerting service. It consults three sources in the course of evaluating a change event. First, it determines whether the event matches a specific object in the library browsing hierarchy - i.e. if the changed item matches is a document or classification. If so, then the Browse Filter will be used in handling the event. Secondly, it consults the Query Filter to see if the change is relevant to any query-style profiles. A single change may be relevant to one or both of these filters. These two filters then respond to the Filtering process with the identifiers of any profiles that matched the change event.

The filtering process then obtains further details on the matching profiles direct from the profiles database, and performs further analysis of the event. This may result in some individual profile matches being discarded. This apparently complex process is necessary due to the potential richness of an individual profile. For instance, a profile may ask for a notification if any new documents are added by a certain author in a set of subject classifications. The most efficient way of finding this profile will probably vary from library to library, depending on the distribution of metadata and which search indices are available.

Reporting Notifications. The notifier accepts event messages that were matched during filtering. Given the user preferences recorded with the matched profile, it selects the appropriate medium for delivery of the notification, and compiles together different notifications to be delivered to the same user if desired. Notifications can be delivered or presented to users in a number of ways. The stereotype of notification delivery may be unicast (e.g., email delivery) or broadcast (e.g., RSS feeds). However, many other forms of delivery can be equally or more effective and the system may follow different strategies for different users.

Alternative forms of reporting include: an "accessions" display within the digital library interface; an "accessions" or "new" classification(s) in the topic hierarchy; or presentation of special objects in a user's DL workspace (e.g. Garnet [5]). In this case, the DL software will have to support the delivery of the notification in this manner, i.e., have a specialised notification component to its interface.

6.2 Distributed Alerting

For distributed alerting, three aspects have to be considered: (1) the distribution of the alerting components shown in Fig. 3; (2) the distribution of local alerting servers as brokers within an alerting network; and (3) the distribution of library content.

Distributed Components. In Fig. 3, we do not present the issue of how the alerting implementation is distributed. One natural division would be to run the alerting processes and storage (in gray) on a different machine to the main digital library system (in white). Other divisions are also possible, e.g., the notifier process could be running separately from the Filtering process. Ideally, query profiles in the alerting services can be matched with the assistance of the digital library server's search engine. If this is not possible, then the query can be processed entirely on the alerting service machine. Though performance could be lower in this case, any efficiently implementing filtering engine will achieve a more than adequate response time.

Distributed Brokers. For discussion the distributed architecture, it is important to introduce a key distinction in alerting service architecture: Two approaches can be used to bring event messages to the profiles which they need to be matched against. First, one can keep profiles on their originating machines, and forward events across the network (event forwarding). Conversely, profiles can instead be forwarded across the network and event messages remain at their point of origin (profile forwarding). Each approach has different merits, depending on the relative rate of change in profiles vs event frequency, network topography and other factors (for details see [3]).

The choice between these two approaches is influenced by the key problem of storing profiles in the context of distributed alerting in Digital Libraries. Unlike event-based systems on the middleware layer, users of digital libraries want to able to access their profiles across a number of different access points (separate DLs). As observed in Section 4.3, it is preferable to allow users to edit their profiles as if all profiles were held in one place. The easiest way to do this is to maintain all the profiles in one actual location. This suggests the approach of event forwarding described above. Event forwarding is also preferable where network connections are not guaranteed, and for other technical considerations. We therefore determined to base the distributed architecture on an event-forwarding principle.

The consequence of this choice is that given a network of servers, a communication network is now required to distribute the events from one server to another. Links to the network are shown in Fig. 3 as incoming event messages to the observer from external sources, and as outgoing notifications from the notifier to external sources. Our design proposed here is agnostic to the chosen implementation of the network. However, it is important to note that research in alerting systems has converged on a tree structure of communication nodes to avoid the traps of loops and duplication referred to earlier. We believe that, unfortunately, an inter-connected network of DL servers cannot always be assumed. Our implementation of a solution to this problem will be discussed in detail later.

Distributed DL Content. Alerting strategies that deal with the distribution of DL content, i.e, the content of the collection data repository (see Fig. 3), depend heavily in the implementation of the data repository. It can therefore not be addressed in detail here. Our implementation describes one possible solution; this can be transferred to other DL implementations following the same internal structures.

6.3 Technical considerations

We earlier identified in Section 4.2 the difficulties caused by the deletion or replacement of documents in the library. If content that is being replaced or removed is immediately erased from the DL indexes, then no information on it can be retrieved. Where the information required for a particular filter to identify a document is required is erased before the filter is applied, then clearly the change cannot be successfully identified. Consequently, deletions must be identified by processing before the actual deletion is executed. This task is further complicated by batch build processes used in D-Space and Greenstone for full-text searching. In such build processes, it is common to provide either a roll-back, if the build is cancelled before completing, or a two-phase build in which the new indexes are completed, and then the old indexes explicitly replaced when the build's success is verified by the librarian. In either case, the notification should not be sent before the new index is completed and made 'live'. The consequence of these different constraints is that the material and indexation of deleted documents must be retained until the event filtering is done, and whilst the new indexes are also available.

7. PROOF OF CONCEPT

In this section, we briefly describe the implemented proof-of-concept system we have completed. Our reference implementation has been created on top of the latest generation of the Greenstone open-source digital library system [19]. Greenstone is particularly interesting as an implementation base, as collections on different hosts can be combined to create distributed and/or federated collections.

7.1 Greenstone 3

The Greenstone digital library software is a popular open source DL system that empowers users to build their own digital libraries. We chose Greenstone (GS) as a testbed system because its high configurability and broad range of features support a large number of potential digital library architectures. A single GS server can host a number of separate collections. Collections can be federated, i.e., they reside on different hosts but have a single, uniform access point. Collections can also be distributed, i.e., a single collection can consist of data sets (sub-collections) on different hosts. Users are not aware of the internal (networked) structure of the collections. They perceive the collections as homogenous structures with a single entry point to all data sets identified by the main collection name. The distribution of sub-collections is also transparent to the users: a sub-collection is presented as being part of a collection regardless of the actual location of the data.

7.2 Alerting Components

The local alerting components were implemented in Java. Users can define profiles by defining search queries as profile queries; document browsing in GS is extended by a "watch this"-button which triggers an identity-centered observation. Each profile is a Boolean combination of a number of attribute-value pairs (on macro level). The term 'values' is used here in a broader sense: Values might be sub-queries (micro-level) such as: (1) a list of IDs, e.g., for hosts and documents; (2) wildcards; or (3) filter queries. The macro profile is evaluated using a variant of the equality-preferred algorithm [9]. On the micro level, we exploit Greenstone's openness by combining index-based filter strategies with the system's own retrieval functionalities. More details about the profile language and the filter implementation can be found in [18].

The interface of Greenstone 3 was adjusted to support separate profile definition and the buttons to document and classification viewing pages were added. The search result and browsing displays have "new" markers placed beside each document or classification that was added since the last rebuilt. Similar visual cues were added for some other changes (e.g. where a document's content had been changed). An additional accession shelf is offered in two versions, integrated as GS page and as RSS feed. The detailed implementation of the local library alerting process is described in [18].

The deletion problem identified in Sections 4.2 and 6.3 is resolved in our Greenstone 3 implementation by an adjustment to the build process. In the adjusted build process, document content that is being removed is kept available to the alerting process before final deletion occurs - i.e. it is tagged for deletion, before finally being removed. During the alerting process, both old and new information can be accessed separately to provide comprehensive input information for the filtering process. In our implementation, the observer component is overlapping with the built component: the last phase of each collection building incorporates the sending of event messages to announce the documents in the collections.

7.3 Distributed Alerting

In our reference implementation, the filtering is distributed across a network of hosts, with individual event messages being forwarded through the network before being processed on connected machines that host Filtering processes. This network structure also allows the efficient handling of change events when individual collections or libraries are themselves distributed or federated.

Networks for Alerting. As described in Sect. 6.2, each implementation requires a network of alerting brokers. The network service used for this forwarding is called the Greenstone Directory Service (GDS); it is an independent network of Directory nodes (see Fig. 4). In the figure. circles represent DL servers, boxes refer to nodes in the GDS. Here, we will only briefly present the GDS architecture; further information on it can be read in [6]. In the Greenstone Directory Service, Digital Library servers register with one GDS node. A GDS node receives messages from the server when its collections are rebuilt, and forward messages to it when collections on other machines change. A tree of GDS nodes forms a GDS community, which can send messages to and from any Greenstone server registered with a GDS node in the tree. GDS messages are sent and received in an XML format using SOAP.

Greenstone servers that host cub-collections are connected to the hosts providing the super-collections via the GS network (see connection in Fig. 4). Both networks are used for the distributed alerting. In addition, information from external services can be incorporated into our system either via the GS network, the GDS network or via Z39.50.

Federated Collections. For federated collections, we chose to follow the concept of event flooding as proposed in our design. Users submit their profiles to a certain Greenstone server. The profiles reside at the server to which the user submitted the profile. After a collection is rebuilt, event messages are emitted by its Greenstone server, which have to be filtered according to the user profiles. After filtering the local profiles, the events are flooded across the GDS network to all other Greenstone servers. This technique is inspired by the event flooding as proposed by Carzaniaga [7]. Our implementation significantly differs from their design as here

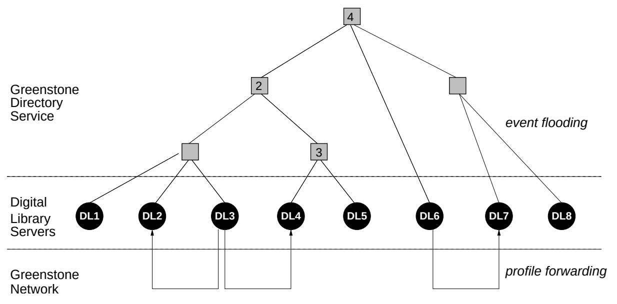

Figure 4: Distributed alerting implementation

we use the Greenstone Directory Service as a communication network and not a network formed by Greenstone servers.

Let's assume a new collection is formed at the Library server $DL1$ in Fig. 4. Subsequently, an event message is created by $DL1$ server announcing the documents in the new collection. The message is forwarded upwards in the GDS tree (via nodes 1, 2, and 4) and downwards towards all the leaves using the GDS protocol. We consider a client connected to the server DL7: Their profile is stored at $DL7$. As soon as the event message arrives at server $DL7$, it is filtered and a notification is send to the client.

Distributed Collections. If a sub-collection that resides on a different server than the collection is rebuilt, the server where the main collection resides will not be aware of the rebuilt sub-collection. The main collection must therefore be a subscriber to the events that originate on the sub-collection server. In order to optimize the performance of the alerting network, we use the concept of *Profile forwarding* via the Greenstone network. Here, the main collection's profile is stored on the sub-collections server, and in consequence messages are forwarded to the main collection(s) directly. Consider the example in Fig. 4: $DL3$ has sub-collections on $DL2$ and $DL4$. Auxiliary profiles for the distributed collection are forwarded to its sub-collections (on $DL2$ and $DL4$). If a sub-collection changes, a message will be send only to the super-collection. The super-collection then announces the event to the rest of the network via the GDS¿

8. RELATED WORK

This section presents the results of our analysis of related approaches pertinent in the context of the proposed application for digital libraries.

8.1 Alerting in Proprietary DL

Individual alerting services are offered by publishing houses (such as Springer Link Alert, Elsevier Contents Direct, or ACM Table-of-Contents Alerts and secondary publishers (e.g., SwetsScan[4] and citation services (e.g, ISI services[5]). These are solitary (centralised) services that neither cooperate with other services nor do they openly support independent digital libraries. Thus, in terms of providing a generic support for alerting that can also be integrated into the digital library - our objective - these primary systems do not provide the answer to our technical goals. Furthermore, when comparing the services mentioned here with the use-case scenarios outlined in 3, a significant gap is identified between the richness of the potential messages of interest, and the provided functionality. These services primarily notify subscribers about new content (i.e. documents) that is available - other change notifications seem to be absent. The services mostly support subject-based subscriptions table-of-contents services but only rarely content-based filtering.

8.2 Protocols usable for Alerting

Only a subset of digital library protocols support features that could be used as the foundation for alerting. Two common protocols stand out: the comprehensive and extensive Z39.50, and the simple and commonplace OAI-PMH. We will briefly introduce both, then discuss them together.

The DL protocol Z39.50[6] possesses an extension for *saved searches*. Such searches are executed periodically, and the results are then posted through Z39.50 to the reader. The options provided by the saved search feature are dependent on the implementation (i.e. the DL server software). However, a periodic saved search suffers the immediate problem of all sampling approaches to event detection - over-sampling to prevent missed events, or under-sampling leading to false positives and negatives (for more detail see [14]). Similarly, searches can only be defined for metadata fields defined at the time that the search was created. Deletions seem to be undetectable. Finally, the degree of adoption of this advanced feature seems to be low.

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)[7] is a popular and established pull protocol that enables clients to harvest the metadata of the content of an online digital archive. OAI is simple to implement, and has been adopted by a large number of institutions. OAI-PMH does not define a notification service for new documents, but there is optional support in the protocol for searches to return only the metadata of items that changed after a given date. Unfortunately, this feature has been developed only conceptually - the actual meaning of the date is not defined. The recently introduced caching option for this information could serve as an equivalent of the accession shelf of a physical library. However, OAI's focus on metadata is inconsistent with the desired content-based filtering. Furthermore, the harvesting protocol does not supports search, which limits the alerting functionality.

Both systems readily identify new documents, and some other changes could be recognized by a sophisticated Z39-50 implementation, but both would require regular sampling, at the expense of either efficiency or timeliness. Neither protocol is complete or sufficient in itself for alerting.

8.3 DL Systems and Alerting

A number of research-level DL systems support some form of alerting services. Two popular and modern systems that support these features are D-Space, and EPrints.

D-Space is a reference model for document management systems, and supports the storage and retrieval of electronic documents. D-Space supports full-text search. A reader using a D-Space server can place a *subscription* to a given collection that notifies then about all new documents. No additional constraints can be added to a subscription - the

[4]available via http://www.swetsblackwell.com/
[5]available via http://isiknowledge.com/wos

[6]http://www.niso.org/z39.50/z3950.html
[7]http://www.openarchives.org

subscriber is simply emailed a notification each day listing any new documents added to the collection.

EPrints is a simple open source system for providing an internet accessible document repository. EPrint servers are stand-alone computers that provide document retrieval and storage. EPrints supports simple *subscriptions* that alert a reader when a new document is inserted into the repository.

The focus for alerting in EPrints and D-Space is on new documents only. Compared to the proprietary systems, D-Space offers a particularly basic subscription service, reporting all new documents, without any selection. This simple approach is inadequate for the envisioned application (cf. UC1). Even EPrints more complex options fail to monitor issues such as changes to classifications (UC4). Therefore, these DL systems cannot serve as an exemplar for a solution to the requirements we identified earlier. Many other well-accepted DL systems, such as Fedora and Greenstone 2, do not at present support any sort of alerting service.

8.4 Mediating alerting services for DL

A few document-centered systems have been proposed which could be used in this context: One of the earliest systems developed was SIFT [20], a centralised tool for wide-area information dissemination, that is now commercially operated as InReference. Support for open heterogeneous document collections is found in Hermes [10], CQ [16] and Dias [15]. Hermes [10] is a mediating alerting system that covers heterogeneous services and provides a single point of access for the users. The options for profile definition focus on typical queries regarding scientific publications, such as authors, title, or keywords. New sources can be integrated; the service operates independent of the library implementation using (active) email or (passive) web pages for information access. Typically, Hermes would be operated by a scientific library (secondary provider) as a service for its users, notifying about documents provided by primary providers. Unlike our alerting system, Hermes only aggregates and integrates alerting from different sources, and is limited by the underlying types of alerts that it receives. Hermes is a centralised system that cannot support distributed user access or distributed collections. It was this restriction that motivated the further work presented in this paper.

CQ supports keyword based queries and focusses on query routing an acyclic distributed environment. The keyword approach is classic but too limiting in our context.

DIAS [15] adopts the basic ideas of Hermes. In contrast to Hermes, it doesn't rely on existing middleware but builds a distributed service based on the design of Siena [7]. DIAS extends Siena's architecture into a peer-to-peer system. The data model of Dias is based on free text and the profile definition language supports a boolean model with proximity operators. Thus, Dias supports a predefined document structure focussing of textual documents. We see this approach as too limiting for a open digital library supporting collections of arbitrary document types (e.g., music, pictures, text documents). Given the presence of non-text media in our use cases (e.g. UC3), such a restriction would not answer the problem we sought to address.

Tools such as Hermes and Dias suffer significantly from not being integrated with the digital library. For example, the problems identified in Section 4.2 are particularly acute when one has no explicit knowledge of the underlying structure of the collections to which notifications, including the set of valid document identifiers. Similarly, change detection that relies on underlying impoverish alerting systems that support limited event types (i.e. "new documents" only) cannot hope to provide notification of changed documents or adjustments to the classification hierarchy.

9. LESSONS LEARNED

In this section, we discuss the lessons learned for the design of a generic alerting service for open DL software. In particular, we present five key lessons learned that have considerable impact on the concepts of DL design:

Rich Event Types A few basic underlying concepts have been shared by previous alerting services - e.g. the idea of a 'new document' notification. However, our user scenarios demonstrate that the limited concept of 'new document' alerts fails to capture the full range of possible user demands. We have shown in our design and the prototype implementation that a richer event space can be supported.

Deletion and Replacement Alerts about deleted and replaced items in the library require particular support from the digital library itself. For example, if a deleted document is immediately removed, we cannot process its content or metadata to identify profile matches. This means that the handling of deletions and replacements in a digital library has to be changed: more information is to be preserved or the whole process has to be changed. Note that the implications are different for items kept within the library (e.g., indexes and classifications) and items where part or all of their content resides outside the DL (DL collection data and multimedia files).

Work Concept Concepts such as a *Work* should be used within digital libraries to ease the identification of succession (i.e., a new edition of a work) or the release of the same work in a different (file or media) format or in another language. Here, lessons from library science can be drawn on to great effect in the context of alerts. For example, a particular work (e.g. Hitchcock's "North by North West") can be known by an entirely different name in another language (e.g., "Der Dritte Mann" in German). Thus, storing a multi-language collection becomes a potential source for multiple titles and a confusion over the actual language of a document. To uphold the concept of a particular work becomes an even more demanding task when using distributed and federated DLs.)

Distribution and Federation Distribution and federation represent a significant challenge. Where a collection is spread over different servers, or different collections are brought together as a virtual entity, we not only have to inform users when the content on one machine changes, but also the other machines as well. This is made particularly difficult as often DL do not cooperate and the connecting networks are therefore unstable and fragmented.

User Issues In a DL environment it is critical for the users to have a single homogenous access point to all their profiles and alert data. Mediating services answer this challenge by providing separate access points that homogenize incoming event information. Integrated services matching the design of a common service proposed here, have to inherently arrive use a different solution: Our design incorporates the functionalities of mediating services by accepting them as providers. Our design has the additional advantage of being closer to the data, avoiding problems in access rights and observation of document changes. Another important design feature is the reuse of the information seeking functionality offered by the library in order to provide fluid user interaction with familiar library services.

10. CONCLUSIONS

Alerting services have become an increasingly common feature of digital libraries. In this paper, we have built upon our experiences with previous systems such as Hermes [10]. Running a basic alerting service on a proprietary stand-alone digital library such as the ACM DL is relatively straightforward, and can draw with confidence on the existing work.

We commenced our research by identifying realistic use cases for a DL alerting service, considering readers, librarians, and technical staff. From this analysis, we extracted a range of design desiderata. When current alerting services for DLs were compared against these use cases and requirements, they prove to support only a limited range of the desired features. Consequently, we presented a general design and architecture for alerting at the local and distributed level of digital libraries.

Through a reference implementation of this design, we overcame problems in the observation of certain events, and prove a hybrid architecture that is open and adaptable to different environments of use. Our distributed system can also be used to ingest alerts from other services - i.e. to provide mediating services as well. The system is also open to support currently unknown document types in the DL by reusing the core-DL retrieval functions that are provided by the DL designer.

We arrived at five key lessons learned that have considerable impact on the concepts of DL design: (1) simple events of 'new' documents are not sufficient in a rich DL environment; (2) current deletion and replacement methods in DL create problems for alerting and require fundamental rethinking; (3) traditional concepts from library science should be adopted to better support quality services; (4) distribution and federation of library software and data require particular attention; and (5) for effective user interaction, generic alerting services cannot be use in a DL without careful tailoring to the DL context.

In future, we intend to expand our current implementation to support further event types, such as carrying notifications regarding reviews of documents. The integration performed so far yields acceptable performance, but with a proof of concept established, more detailed performance tuning will now be undertaken.

11. REFERENCES

[1] D. Bainbridge, G. Buchanan, J.McPherson, S. Jones, A. Mahoui, and I. Witten. Greenstone: A platform for distributed digital library applications. In *Proceedings of ECDL*, 2001.

[2] N. Belkin and W. Croft. Information retrieval and filtering: Two sides of the same coin. *Communications of the ACM*, 35:29–38, 1992.

[3] S. Bittner and A. Hinze. Classification and analysis of distributed event filtering algorithms. In *Proceedings of the OTM: CoopIS, DOA, and ODBASE*, 2004.

[4] A. Blandford, S. Keith, I. Connell, and H. Edwards. Analytical usability evaluation for digital libraries: a case study. In *Proceedings of the JCDL*, 2004.

[5] G. Buchanan, A. Blandford, H. Thimbleby, and M. Jones. Integrating information seeking and structuring: exploring the role of spatial hypertext in a digital library. In *Proceedings of HYPERTEXT'04*, 2004.

[6] G. Buchanan and A. Hinze. A Distributed Directory Service for Greenstone. Technical Report 01/2005, CS Department, University of Waikato, New Zealand, January 2005.

[7] A. Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks.* PhD thesis, Politecnico di Milano, Milano, Italy, December 1998.

[8] A. Crespo and H. García-Molina. Awareness services for digital libraries. In *Proceedings of the ECDL*, 1997.

[9] F. Fabret, F. Llirbat, J. Pereira, and D. Shasha. Efficient matching for content-based publish/subscribe systems. Technical report, INRIA, 2000. http://wwwcaravel.inria.fr/pereira/matching.ps.

[10] D. Faensen, L. Faulstich, H. Schweppe, A. Hinze, and A. Steidinger. Hermes – a notification service for digital libraries. In *Proceedings of the JCDL*, 2001.

[11] D. Goh and J. Leggett. Patron-augmented digital libraries. In *Proceedings of the ACM DL conference*, 2000.

[12] A. Hinze. *A-MEDIAS: Concept and Design of an Adaptive Integrating Event Notification Service.* PhD thesis, Freie Universität Berlin, July 2003.

[13] A. Hinze and D. Faensen. A Unified Model of Internet Scale Alerting Services. In *Proceedings of the ICSC (Internet Applications.)*, 1999.

[14] Annika Hinze. How does the observation strategy influence the correctness of alerting services? In *Proceedings of the BTW (German national DB conference)*, 2001.

[15] M. Koubarakis, T. Koutris, C. Tryfonopoulos, and P. Raftopoulou. Information alert in distributed digital libraries: The models, languages, and architecture of dias. In *Proceedings of the ECDL*, 2002.

[16] L. Liu. Query routing in large-scale digital library systems. In *Proceedings of the ICDE*, March 1999.

[17] G. Salton. *Automatic Information Organization and Retrieval.* McGraw-Hill, New York, 1968.

[18] A. Schweer. Alerting in Grenstone 3. Master's thesis, University of Dortmund, Germany, 2005.

[19] Ian H. Witten and David Bainbridge. *How to Build a Digital Library.* Elsevier Science Inc., 2002.

[20] T. W. Yan and H. García-Molina. SIFT - a tool for wide-area information dissemination. In *Proceedings of the Usenix*, 1995.