Alerting in a Digital Library Environment
Do Channels meet the requirements?

Daniel Faensen, Annika Hinze and Heinz Schweppe*

Institute of Computer Science, Freie Universität Berlin, Germany

{faensen,hinze,schweppe}@inf.fu-berlin.de

Abstract

The classical paradigm of finding information in the WWW by initiating retrieval and browsing becomes more and more ineffective. Other techniques in the context of digital libraries have to be considered. Automatic delivery of contents to the user according to her needs and filtered by her profile of interests is required. Current implementations of such *alerting services* at content providers side have several drawbacks, e.g. users interest cannot be defined appropriately. In this paper we present an architecture for an alerting service within a general framework of events. Two competing technologies, Netscape's Netcaster and Active Channels by Microsoft, have been proposed for implementing alerting services. Both technologies will be evaluated according to the requirements of a digital library alerting service.

Keywords: digital libraries, push technology, alerting, channel, CDF, Netcaster

1 Introduction

Imagine one morning you just arrive at your office and switch on your computer to have a look at the recent news in your special field of research. Little pictures for each topic tell you that some interesting documents arrived. Behind one icon you find for instance the new announcements for congresses, behind another one some interesting papers, tables of contents of your favorite scientific journals, announcement of new books in your field. In a particular manner it reminds you of the old times, when you just went to the library and the librarian came up with some news: 'Oh, this might be interesting for you'. She knew which field you are working in, your specific topics of interest, the papers that may concern you. These times will never come back, simply because the quantity of scientific publications doubles every 10 - 15 years [8]. This means that the number of publications written from the very beginning until 1985 is roughly the same as the number of publications since 1985 until now.

What is happening? In the times of the World Wide Web, one had to navigate through different places on their own. When searching for information one is bound to unsatisfying tools. One poses a request to a bibliographic or

*Part of this work has been done while staying at the Intern. Computer Science Institute, Berkeley, CA 94704.

even full text database or browse the Internet with the help of a search engine. But what one really wants is an information offer. *Being informed* instead of actively chasing after hyper-links is the paradigm of the future: *What You Get Is What You Want.*

Two presumptions must be met in order to offer such a service: a technical infrastructure for alerting and a service that knows the users topics of interest.

In this paper we investigate a model of information dissemination in a digital library. The model is particularly oriented towards science libraries – libraries which are among the most important tools for scientific research. Even in a conventional library information overload is an issue nowadays. A digital library has access to all digital sources in the world – at least in principle. Ignoring costs, still existing technical problems like interoperability [9] and the lack of computing infrastructure in many institutions outside computer science, the urgent problem is how to get access to relevant information. This problem has been traditionally studied in information retrieval [10]. The request-reply paradigm underlying most retrieval systems has been partially substituted by the browsing paradigm of the World Wide Web. But with the advent of hundreds and thousands of web servers offering high quality information – publishers, preprint servers, archives of peers in a field – browsing is no longer a cost effective technique.

Different kinds of web-related tools are offered today which inform clients when new information is available. Table of Contents (TOC) services are the most popular among these. This kind of alerting seems to be also appropriate for a digital library scenario.
Even perfect alerting services offered by different providers would be of no great help for a user, if she has to deal with many providers on an individual basis. We will therefore also suggest an alerting architecture which will notify her about all relevant 'information events' independent of the source. This is what we would expect from a *perfect bibliographer.*

The emphasis of this paper is on alerting, not on techniques for filtering according to relevance criteria of individual users. We will therefore only shortly discuss issues of defining individual profiles of interest.

Despite its obvious usefulness, nothing can be found on alerting services in the literature on digital libraries. Important standards like Z39.50 [6] do not even mention this type of service. On the other hand, some publishers are offering alerting services since a few months and others are planning to do so. Good examples are Springer Link Alert [14] and Elsevier Contents Direct [15]. Tables of contents are delivered by e-mail. Elsevier additionally offers a CDF-based service called Science Channel [16]. The main limitation is the restriction to individual services. Suppose your library offers a table of contents service – you may order paper copies of the TOC of individual journals. Would you like to register for each individual publisher who offers one of your favorite journals? This is exactly the situation we are facing today. Only some publishers offer more than a TOC service, e.g. the definition of a user relevant keyword list. Some non profit services like Ariadne [17] or SEL-HP [13] offer keyword-based profiling: clients are notified as soon as documents matching the profile arrived. Profiling services have also been offered by bibliographic databases, some with sophisticated profile definition languages, like CompuScience [22]. This heap of different offers has some unpleasant side effects:

- Profile semantics. The Boolean retrieval model has proved ineffective [11], but keyword lists are treated as degenerated Boolean queries.

- Privacy. The user gives personal data (profile) to many of different persons or companies.

- Completeness. The user probably misses some important information providers. In the networking environment, one has everything available at their fingertips, but one usually does not know exactly where.

- Poor support. You are on your own in looking for information. The situation is unsatisfactory compared to traditional use of libraries, where you find nearly everything at its place, or have a friendly bibliographer helping you.

The contribution of this paper is a framework of *alerting services* (Section 2) and an investigation of *channels*, which are a popular technique for information dissemination in the business-oriented web. In particular we will investigate the competing *Netcaster* and *Channel Definition Format (CDF)* based technologies. CDF, a document type definition specified with the eXtended Markup Language (XML), has been created in order to make it easy for information providers to notify their clients about any changes in their offering. It is used today primarily by news agencies and product announcement. We will analyze this technology with respect to digital science library needs (Section 3).

2 Alerting services for Digital Libraries

In this section we first introduce a general model of event handling that can be used by an alerting service. Such an alerting service in the context of a digital library has several specific requirements: filtering information according to users interest and making the heterogeneity of providers transparent to the users. User interest is defined in so-called *profiles*. We describe several profile types. At the sections end we propose an architecture for a digital library alerting service.

2.1 A general model of event handling

In this section we discuss an implementation independent model of alerting. An *alert relation* A is a time-dependent relation between *suppliers* S, *consumers* C and *objects* O.[1] Suppliers offer certain objects. Customers may *register* or *unregister* for certain objects belonging to certain suppliers. The corresponding tuples are inserted into or deleted from the relation A. Objects have a state. A state transition of an object is called an *event*. The projection[2] $E = \Pi_{S,C} A$ is called the set of *event channels*. Each pair (s, c) together with the set of objects $O_{c,s} = \{o | (s, c, o) \in A\}$, constitutes an event channel.[3] Let $O_{t',t}$ be the set of objects that changed their states within some time interval (t', t). Then $E_{t',t} = \Pi_{S,C}(A \bowtie O_{t',t})$ is the set of channels affected by the events that happened in (t', t).

[1] We follow the terminology of the Corba Event Service Specification (COSS) [19].

[2] We employ the well known notation of the relational algebra.

[3] COSS calls the relation itself an event channel.

Only suppliers are aware of events. More precise they are aware of all events on objects they offer. An *alerting service* informs customers about events of objects they have registered for. In the *push model* of alerting the supplier s is the active part. It has to inform all customers in $\sigma_{S=s}E_{t',t}$. In the *pull model* a customer c regularly checks the relation $\sigma_{C=c}E_{t',t}$ for which she registered. In case it is empty, no event occurred.

One of the main features of alerting – i.e. pushing or pulling events – is asynchronity. Events can happen at any time t_e, they are pulled or pushed "through the event channel" at an arbitrary time $t_p \geq t_e$, however. An alert relation may therefore be enhanced by an alerting *time policy.* Such a policy specifies when events are to be submitted through channels. Time policies are defined by suppliers, customers or both. Examples of such a policy are:

- *immediate*: an event is transmitted to a customer when it occurs
- *periodical*: events are transmitted every n units of time, if there are any,
- *quantitative*: when m events have occurred, they are transmitted.

Up to now, the alerting relation was given explicitly. A customer specifies the set of objects, the events of which she wants to be informed about. We assumed this set to be given extensionally, e.g. as a set of object identifiers. However it is useful to allow also for an intensional definition by means of a *filter predicate.* We will call such a filter predicate, defined on the set of all objects offered by suppliers, a *profile.*

The relationship between customers and clients is an $n:m$-relation. We will later discuss why a decomposition into a $1:n$-relation and a $1:m$-relation is useful.

The abstract view models quite different alerting – sometimes called event notification – applications:

- an online stock exchange broker offering quotes.
- the distribution of the change list of parts in an engineering project.
- keeping replicated data consistent.
- a news agency feeding its clients with the latest news by means of web pages.

This last type of applications have promoted implementations by means of CDF or Netscape channels. Before we discuss whether these techniques are suited for digital library applications, we relate the abstract model to alerting services of digital libraries.

2.2 Alerting in Digital Libraries

In the context of digital libraries, suppliers are the *providers* of *documents.* Hence the different kinds of electronic documents correspond to the more general term *object* in the model. Providers are typically scientific publishers. In this paper we assume, that the providers are known to the clients.

There are common patterns of how scientists use scientific literature. The most important one – looking up a reference while working on some problem

– is outside the scope of this paper. This is due to the inevitability searching the article in a database, the library or on the shelf. But this has nothing to do with alerting.

A scientist usually follows the publications in a few journals and conference proceeding, those that fit into her research area. She scans the TOC of the new issue, reads a few abstracts and carefully studies one or two of them. This is exactly were an alerting service becomes useful. More and more publishers are offering such a service.

Yet another pattern of accessing scientific literature is browsing in the local library. One picks up some journal one does not read regularly, quickly scan some papers and perhaps read one carefully, which happens to fall into your scope of interest. Browsing in the vast amount of electronic documents spread over thousands of servers is completely unreasonable. It is therefore valuable to have access to a *profile service*. As introduced in the model, a *profile* is a filter predicate. In the context of digital libraries this is nothing but a retrieval query executed periodically without explicit intervention by the user. Only the results are presented to her. Of course a single user may have defined more than one profile.

2.3 Profiles

Profiles differ from ordinary queries only in one respect: they are evaluated periodically without direct user intervention. The issues of querying in information retrieval are well know – precision, recall, subjectiveness of relevance judgment. Since more than one provider is usually involved in an alerting service, we may encounter the problems of distributed query processing. This is particularly demanding in an heterogeneous environment. However the situation is not quite as hopeless with alerting services, at least in simple but important situations. We define three types of profiles:

- a set of identifiers for documents, journals in particular,
- a list of keywords which can be either selected arbitrarily or from a thesaurus given by the alerting service, and
- a query in a full-fledged retrieval language.

TOC services today use identifiable objects like journals, content-oriented services use keyword lists. Both are valuable, but it is obvious that queries in a state-of-the-art query language would be the best choice.[4] It might even support relevance feedback or documents as queries (*find all documents like this one*), but we ignore these particularities this paper, since we do not discuss pros and cons of specific retrieval operators. However, we will discuss how these sophisticated ways of profile definition could be included in an alerting service.

A language suitable for our needs is *STARTS* [4]. It allows for formal attribute queries (called filters) and retrieval queries with different kinds of operators (e.g. proximity, weight assignment), called ranking expressions. A simple but useful profile might ask for all new publications of some research group. This query can be formulated by means of formal attributes. The same holds for TOC profiles, which could also be easily expressed in this way.

[4] It could of course also be profiles of the first two types.

In the following, we assume a full-fledged query language like *STARTS* [4] for formulating profiles. We assume appropriate wrappers to map query exressions to the interface of a provider.

2.4 Requirements

The most important requirements from a user's point of view are:

- an appropriate language to express indivual profiles and
- a unified view of the service.

The first requirement depends on the sophistication of the query language. The second one is as important as the first one: imagine one has to register for five or even more alerting services offered by different providers. This might be as confusing as dealing with five different libraries at a time. It is therefore inevitable, to split the $n : m$-relationship between providers and clients by means of an an alerting service: Clients register with this service, which in turn is connected to several providers. To be more specific: the local library could be the right institution to host this service. This would even avoid the privacy problem mentioned above, i.e. to give profiles of interest to different institutions or companies.

Alerting works asynchronously, however the client should be able to define when she wants to be informed. Even e-mail delivery of alerts has disadvantages, despite the fact that an e-mail can be processed at any time by the user. As soon as she gets lots of e-mails each day, however, – a situation many scientists are facing already today – it it very tempting to delete the less urgent ones. Therefore it should be possible to separate the alerting service from other tools on the user's desktop.

The user interface should be customizable: depending on the user's taste she could be alerted by a colored button in her browser or by an e-mail flag of a mailbox dedicated to the alerting service. There is an important technical requirement for alerting services: scalability. The university library could host the service. It is the natural partner of publishers – not only because it makes the contracts and pays the fees for electronic journals. If we consider thousands of students and scientists register, the service will slow down, be vulnerable and inflexible – as most centralized systems. For this reason a distributed technical solution is necessary. It is however beyond this paper to discuss the various technical and organizational issues of such a solution.

2.5 Architecture

Let us first view the Alerting Service *AS* as a black box. On the client side, the interface is simple: a client registers to the service by specifying her profile and other parameters – *when? how?* – the service alerts and supplies her with documents according to the needs specified.

The provider interface is more sophisticated. Providers may be *cooperating* or *non-cooperating*. Cooperating providers offer alerting services themselves, e.g. a TOC service or more. Non-cooperating providers allow to access their material – e.g. a database of journal articles, but they are not able or willing to announce new material by means of a proprietary alerting system.

Let us first discuss a non-cooperating provider P. The alerting service must periodically evaluate the client profiles against the data offered by P. This could be a nearly impossible task, if the provider does not even supply a search interface. A situation like that could happen e.g. when a research group offers an ftp archive to the public and nothing more. In this case, all files created or changed after the last inspection of AS have to be transferred and checked against the profiles. The situation is even worse, if date or time attributes of documents do not exist or are inaccessible for AS. It is, however, unreasonable to down-load all documents of a provider. An agent interface of a provider P would help in this case: the agent code, sent by AS, could filter the providers documents according to user profiles.

A problem with both cooperating and non-cooperating providers is to keep track of the data a particular client has already received. If $O_{P,C}$ is the set of objects supplied to C at time t, they should never again be presented, except when they have changed. This gives raise to a buffer managed by the alerting service for each individual client, which keeps track of what has already been delivered. In the most simple case, the buffer contains nothing but a time for each client C and each provider t_P: objects of P must only be shipped to C if they have a newer state than at time t_P.

We briefly discuss how the alerting service AS executes profiles in case of *cooperating providers*. Let us assume that the provider understands the language of profiles, otherwise the profile has to be approximated by some wrapper. A naive way of utilizing the alerting facilities of the provider would be to send one profile – the alerting service profile – to the provider, which comprehends the profiles of all clients. As a consequence, AS itself has to reevaluate all client profiles as soon as the providers alerting service delivers documents which fit to the "unified" profile of AS. It is therefore more appropriate to forward all client profiles to the cooperating provider. They may be anonymized in order to respect privacy, but the burden is up to the provider: She has to keep track of multiple client profiles, just as in the case of no intermediate alerting service. But since all of this happens behind the curtains, there is still the big advantage for the clients: they only register at one alerting service.

In Figure 1 we roughly sketch the overall architecture of the service.

3 Alerting Technique

In this section we explain different ways to implement alerting techniques with special respect to do personalization of the object selection.

3.1 Push vs. Pull

The two major techniques for notification services are *push* and *pull*. Hybrids are also possible, we will explain one exemplary.

push In client/server applications push-technology means sending data to a client without the client requesting it.

pull The client requests data from another program or computer (server).

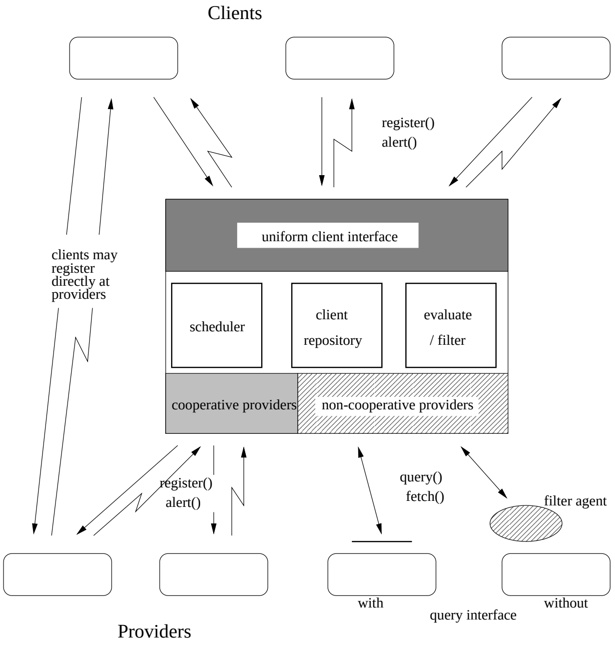

Figure 1: Architecture of the alerting service

smart pull Most implementations of push are technologically not true push but rather a scheduled pull. This is called 'smart push' or 'smart pull'. Partly this is due to the fact that users are not always online. It is possible to configure the schedule interval on the client side. The major advantage on the server side is clearly the avoidance of bookkeeping for each user. However bandwidth is wasted since clients will normaly pull more often than necessary.

3.2 Implementations

Alerting services within the context of digital libraries can be implemented in several ways. Here we focus on the channel technology by Microsoft and Netscape. Two other possible approaches are summarized shortly.

The *Corba Event Service* [19] allows applications to communicate with one another no matter where they are located or who has designed them. The object request broker (ORB) is the middleware that establishes the client-server relationships between objects. Using an ORB, a client can transparently invoke on a method of a server object, which can be on the same machine or across a network. The event service decouples this process and allows several servers to communicate asynchronously with the clients. The suppliers may push or pull their objects.

Active databases could also be used as a platform to implement an alerting service. The main drawback of current active database technology is that event-triggered actions are either performed inside the database management system itself or as an invocation on a registered client software. In the digital library the information provider holds all information concerning the documents while the user profiles are kept elsewhere. Matching newly entered documents with user profiles will be hard or impossible.

The channel technology by both Microsoft and Netcaster are very simple to implement and are system independent. Currently (April 1998) both technologies still require Microsoft Internet Explorer and Netscape Netcaster, respectively. However they are both submitted to the W3 Consortium to become a standard for channel technology and it is expected that other browser vendors will support at least one of this solutions. It is also possible to develop your own alerting applications using CDF, a format upon which the Microsoft channels are based.

3.2.1 CDF-channels

CDF (Channel Definition Format) is the proposed open industry standard for data definition of content to be pushed across the Internet. CDF is an application of XML, which is very similar to SGML. Therefore CDF defines a document format. CDF documents specify at what time which data are to be multicasted from a web server to clients, that have submitted a permanent request for the data – and which have a CDF compatible receiver program.[5] "Broadcasts" are implemented by smart pull for the reasons given above.

CDF documents are independent of any transport protocol. The existing implementation for Internet Explorer 4 (IE4) by Microsoft uses HTTP. CDF is also being embraced by most of todays push software vendors, including e.g. PointCast [26] and BackWeb [27].[6]

We first want to introduce to a 'simple' channel and later explain the point of personalization which is especially interesting for profile services.

Simple Channel

Provider Side. To build a channel one needs a CDF file that references the content of your channel. A channel can contain documents and/or programs in different formats like HTML, ActiveX or Java. For simplicity we call these different contents *channel item*. The syntax of a CDF file is similar to HTML. There is a set of tags which are used to link the items and give some conditions like how to notify the subscriber and how often. The channel content can have a hierarchical order. A short overview of the structure of a simple channel is given in the following code fragment:

```
<?XML Version="1.0">

<CHANNEL HREF=http://...>
<LOGO HREF=...>
<SCHEDULE STARTDATE=...>
  <INTERVALLTIME DAY=../>
</SCHEDULE>

<ABSTRACT> text </ABSTRACT>
<TITLE> channel titel </TITLE>
```

[5] Microsoft calls this Webcasting [24].

[6] For a complete list see [23].

```
<ITEM HREF=...>
  <ABSTRACT> text </ABSTRACT>
  <TITLE> item titel </TITLE>
  <USAGE VALUE='DesktopComponent'>
    ...
  </USAGE>
</ITEM>

</CHANNEL>
```

The file starts with a line that identifies the XML version. The main page of the channel is declared by the CHANNEL tag and all information about the contents is stored within the `<CHANNEL></CHANNEL>` block. In the SCHEDULE block one can specify how often a channel needs to be updated. The intervall the channel applies can also be included. As well as for each channel item one could include title and abstract of the channel. ITEM defines a subpage and delimits its necessary information: the URL with the location, the title and the abstract with a longer description. With the USAGE tag one can define the form the item should be presented to the user, here as a particular desktop icon.

For a 'simple' channel (which means just for broadcasting) the CDF-file only needs to be placed on an HTTP server. Now the users can subscribe to the channel.

User Side Upon visiting a CDF enabled Web site by IE4 a user can subscribe to the site's channel by clicking a hyperlink to the CDF file. Within the following process of registering the user is allowed to customize channel behavior. One can choose how to receive the notification (by e-mail, by a changing icon on the desktop or by delivery of the new content to your screensaver[7]) and how often the channel should be tested for changes. There is a choice between a download of the new material or only a notification. If it is a hierarchical built up HTML-based channel the user can decide how deep she wants to search for news.

Internal View If a user customizes a subscription to only check for updated content IE4 periodically visits the site, downloads only the CDF file and updates the channel hierarchies (Figure 2). The CDF file provides information about new content and categorized links to channel topics.

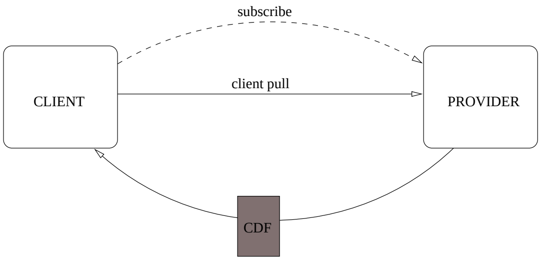

Figure 2: Schema for channel update for *online* use

[7] This last choice is particularly suited for online delivery of stock values

By subscribing to a channel for offline use (Figure 3) the CDF file and all new or updated channel content referenced in the CDF file will be downloaded.

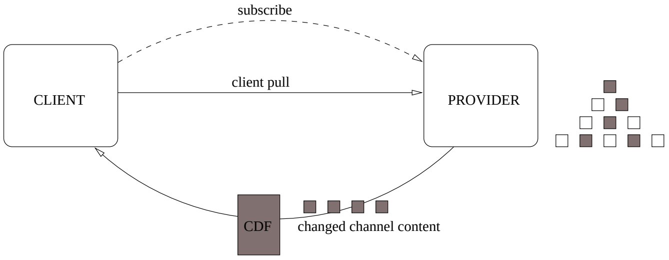

Figure 3: Schema for channel update for *offline* use

The technique used by IE4 is called smart-pull by Microsoft. This way one can watch a channel as a simple alerting service.

Alerting and personalization
Based on the alerting model introduced in Section 2.5 we have several providers, one mediative alerting service and several clients. As mentioned before we have two types of providers: A cooperating type like Internet publishers and a non-cooperating one like a database of scientific documents commonly used by a geographically distributed research community.

A cooperating provider creates CDF channels with references to the published objects. The alerting service subscribes to this channel. Because the CDF file documents the time objects are updated or new the service is always aware of the changes.

The service also knows of the user profiles. Now there are various possibilities to implement the personalization.

1. The alerting service offers a hierarchy of subjects the user can subscribe to. Each document is assigned to at least one of the subjects. The service provides an HTML form where the user can select the subjects she wants to get informed about and gets notified when

   (a) objects under this topic are changing,

   (b) new objects with this topic occur and

   (c) objects with completely new topics enter the channel.

   Each time the browser performs a scheduled pull the service dynamically creates a CDF file for each user according to the users profile. The user is identified by the use of cookies. The profile can be stored either on the service side or within cookies on the client side. The Science Channel [16] offered by Elsevier is implementing this technique in a similar way. The already mentioned drawback is that the user has to deal with individual providers.

   The notification of the alerting service by cooperative providers can be done using simple CDF channels. This method has the disadvantage that the user cannot enter her own keywords and is always bound to the hierarchy implemented by the service. Classification of the objects has to be done by the provider. Non-cooperating providers are hard to include.

2. The user subscribes to the alerting service and defines a profile. This will be stored on the service side. The service defines a CDF file for each profile or generates it automatically on demand. The service checks all incoming objects and decides to include them into the channels according to the profiles. We attend the point of *how* to check the objects later. To involve a non-cooperating provider the service initiates a request to the database via http. The resulting HTML page is saved and added to the channel.

   The advantage of this implementation is that it takes different types of users into account. The user has more freedom in choosing interesting topics. There is a real personalization.

Object selection
The filtering of the objects regarding the profiles can be done in different depths.

1. For each item in the CDF file the provider can add an abstract and a keyword list. Either the keyword list can be compared with the profile, or the abstract is searched for the topic. If the keyword list is detailed this could be an easy and useful method to classify objects. In this approach one is always bound to the will and the competence of the providers in constructing the keyword list.

2. Because the service holds the links to new objects and the defined profiles it can do the comparison. One problem can be that the service will probably not be allowed to have access to the objects.

3. The service initiates an inquiry to the search engines of the providers according to the profiles. Objects of providers who do not maintain a search engine can not be graded.

3.2.2 Netcaster

Netcaster channels collections of files that use Netcaster to deliver dynamic information to the clients. Netcaster is a component of Netscape Communicator. Netcaster channels can be delivered in two ways: as standard-based web-server channels or Castanet Transmitter channels [21]. We will focus on the second ones because they give a chance to involve personalization. Castanet channel technology has been developed by Marimba [25]. The channels are bidirectional connections between *Transmitter* and *Tuner*. The channels are hosted on a *Castanet Transmitter*. It can consist of HTML pages or an entire web site including images and sound, the pages must be subordinate to a particular directory. It can also be a Java application or applet.

Provider Side The channel consists of the files of a special sub-directory and additionally a properties file, which specifies the name of the channel, its type, how often it should be updated and so on. Then one has to publish the channel with the Castanet publisher tool. This copies the files to a Transmitters channel directory.

User Side User may decide to click on a link of a Castanet channel Transmitter to receive a list of channels offered by that Transmitter. Selecting a channel name in the list subscribes to the channel. The channel behavior is defined by the provider and the user has no influence on it.

Internal view Castanet is a client pull system. The updates are done dependently from channel and tuner configuration. They are incremental that means that there will be transmitted only the changes of the files.

Alerting and Personalization
Both Java- and HTML-based channels can send user feedback to the transmitter. Java channels return any sort of data; HTML channels return records of the user behavior. These data are stored in a log file to be analyzed offline. Alternatively one can write a Transmitter plug-in that analyzes the data in real time.

The feedback supports the concept of personalization. The same mechanisms as used for CDF channels are applicable. The alerting service will connect the users to more than one provider.

1. The alerting service maintains one channel for all users with a hierarchy of subjects. As mentioned the user's behavior is stored in a log file. Hence it is easy to implement a Java applet to guide the user in a special profile definition mode. With clicking on the topics the user can construct the profile.

2. Similar to the procedure with CDF the user subscribes to the alerting service and defines a profile. The service defines a properties file according to each profile.

For the shown similarities the consequences mentioned in the CDF case are also applicable for Netcaster.

4 Evaluation of Channel Technologies

In this section we will evaluate how the different requirements to an alerting service defined in Section 1 will be met by the two competing channel technologies Netscape Netcaster [20] and Microsoft Active Channel [23]. We first concentrate on CDF technology to compare later the additional features of Netcaster.

The use of both technologies strongly depends on how the contents is to be filtered, i.e. how the user profile is to be defined. To support comprehension the evaluation is done regarding an example where the documents of interest are articles from electronically published journals. A summary of the evaluation is given in Table 1.

1. **Profile is defined by links to document collections.**

   An example for this is a table-of-contents (TOC) service. The user defines her profile through a collection of journals. Each time a new issue is delivered the user will be informed. The role of the alerting service is to hide and integrate different publishers.

   Implementation is easy. Each provider offers a channel containing all its objects. The alerting service subscribes to these channels and updates the

		TOC	Profile definition by				
			Subject	arbitrary keywords		sophisticated	
				metadata	FT	related doc.	weight
alert between	AS - U	stat./dyn.	dyn.	dyn.	dyn.	see discussion	
	coopP - AS	easy	provider supplied[a]	provider supplied	external		
	ncoopP - AS	special[b]	external[c]		external		
	profile stored at	U or AS	U or AS	U or AS	AS[d]	U or AS	AS

[a] provider has to assign subjects/keywords to metatags

[b] analyzing page structure with external programs, e.g. HyperNavigator [2]

[c] via search interface to the providers database

[d] because of asynchronity

Table 1: Evaluation of the use of CDF as technology to implement a digital library alerting service with respect to the way content (journals and journal articles) should be filtered. For each way of profile definition we show where the user profiles can be stored, how notification can be sent from alerting service to user (AS - U), how the communication between information providers and alerting service is supported (coopP - AS) and how non-cooperative providers can be included. For details see text.
Legend:
TOC – The user will be informed on the Table Of Contents of newly published issues of journals she is interested in
FT – keywords are matched against FullText
U – user / user side
AS – alerting service
coopP – cooparative provider
ncoopP – non-cooparative provider
stat./dyn. – CDF file is hold statically respectively generated dynamically

user's channels according to their profile when notified by the providers. Update can be done at notification time or dynamically when the user client pulls.

In the first case the profiles have to be kept in the services repository, in the second case they can be stored in cookies at the user's side.

Both CDF and Netcaster are appropriate for delivering content from both alerting service to user and from (cooperative) provider to alerting service. Dealing with non-cooperative providers is more complicated. Here the information has to be extracted from the providers pages in a scheduled pull process. This can be done e.g. with the HyperNavigator [2] if it is offerd as web pages. This is not trivial since for each provider rules describing the page structure have to be defined.

2. **Profile is defined by subjects.**

   The user composes her profile by a subset of subjects offered by the alerting service. She will be notified if documents are published which are subordinate to one of the subjects. The alerting is at a finer granularity than just TOC.

   The implementation at the provider's side is almost the same as in 1. Additionally it becomes necessary that the provider classifies the documents according to the set of subjects hold at the alerting service. In the CDF file the `<ABSTRACT>`-tag could be abused. The alerting service dynamically generates CDF files for each user by matching provider assigned subjects against her profile. This can be done on-the-fly when the client pulls if it doesn't lead to prohibitive response times. If CDF files are generated offline the profiles have to be kept in the services repository. In Netcaster Channels this is even easier. The providers can store additional information in the properties file. The alerting service gets information about the user decisions by checking the log file. Non-cooperative providers have to be queried as described in the following paragraphs.

3. **Profile is defined by arbitrary keywords.**

   The most common way scientists define their interests is by using a list of Boolean linked keywords (or other metadata like author). We distinguish two different ways documents are rated relevant according to a keyword defined profile: Keywords can be matched against the document metadata or (in the case of scientific papers) against the fulltext.

   In the first case the provider delivers CDF files with all relevant metadata. Filtering can be done the same way as described above. Profiles are kept at user's or services side. The second case is more complicate since the alerting service has no access to the documents. Being notified that new documents are available the alerting service induces one query for each profile on the fulltexts using the providers search interface. From the results CDF files are generated. This implementation has several drawbacks. The search interface (if available) should be standardized but normally it is not. It is not clear how to handle ranked retrieval results. Querying several thousand profiles against the whole database of each supplier is very ineffective. Few providers offer restriction of queries to new objects.

With non-cooperating providers the situation is the same. At least a search interface has to be provided. Querying is done by the alerting service against metadata and / or fulltexts.

4. **Profile is defined sophistically.**

   One way of defining more sophisticated profiles is by using STARTS [4] where one has features like weighting of keywords, proximity, or by describing users interest by related documents.

   We focus on the implementation of the latter which is the most complicated. To rate matching similarity objects with sample documents the service somehow needs access to the objects. Since direct access is impossible – an alerting service does not keep documents locally – comparison has to be done at the provider's side. Via the query interface the reference document has to be transmitted to the provider, either the full document or a pointer to a source from where the provider can get it. The query result would be a relevance ranked list – with all the problems handling relevance ranking.

   Alternatively to keeping profiles at the alerting service they can be made known to the providers. In this case the role of the alerting service is to perform a distribution of the user profiles transparent for the users. Since sample documents are usually memory-consuming, pointers to sources of these documents can be stored in the profiles instead. Keeping the profiles at the user side is not convenient since querying is too time consuming to be done at the time of the 'smart pull'.

   In a Castanet Channel for each item additional information or dedicated Java applets can be added. With the applets access to search interfaces can be implemented. This gives the possibility to offer the user an easy way to define complex profiles. With an additional Transmitter plug-in the log file can be analyzed online so that the channel can automatically adapted to the user needs. In this area lies the real advantage of Netcaster Cannels – it gives more possibilities to adjust the channel technology to the application needs.

   User channels can be generated according to profiles in the same way as described above. Channels offered by content providers are only useful for notifying the alerting service about new content which could induce a query. Non-cooperative providers have to be queried in a scheduled manner.

   Profiles with weighted keywords can be implemented as described in 3. Since matching reveals some kind of similarity measure a threshold has to be defined to rate objects as relevant or non-relevant.

5 Conclusion and Future Works

The traditional paradigms of working with electronically published information – request-reply and browsing – become more and more ineffective. The increasing amount of available information requires new technologies for information access. Information should be *pushed* to the user filtered according to user's interest defined by a *profile*. Behind a scenario of a digital library providing access to a heterogeneous set of information providers, we proposed in this paper an architecture for an alerting service notifying users for newly published objects. We evaluated both Microsofts Active Channels and the Netscape Netcaster, if they meet the requirements that arise in this context.

The most remarkable difference between Michrosoft Active Channels and Netscape Netcaster is (i) the necessary efford to implement a channel and (ii) the flexibility the technology offers. While a CDF file is quickly written only few program logic can be added. Netcaster Channels give the programmer a much more flexible tool. However, the implementation effort required by Netcaster is much higher then by Active Channel.

In our architecture channels can be used at two communication pathways: (i) as a tool for information providers to notify the alerting service about the occurance of new objects and (ii) to push profile filtered collections of new objects from alerting service to users. For the latter case both technologies revealed to be adequate. For the first case – as discussed in Section 4 – the use of channel technology strongly depends on the complexity of the profiles. In a simple TOC service where the alerting service only acts as a mediator between user and provider, channels can easily be implemented on the provider side. If profiles become more complex, responsibility for the provider increases. She has to add appropriate metadata to the channel items since querying is done against them. If queries have to be posed against the whole objects (e.g. full texts) the provider channel can still be used to notify the alerting service which in turn initiates a query against the provider's database using its search interface. Providers without such an interface cannot be supported. Non-cooperating providers, i.e. those which don't want or are not able to implement channels, in each case have to be queried in a scheduled manner.

The combination of retrieval and alerting leads to some problems. Most search interfaces give a *relevance ranked* query result. It is not easy to deduce from the rank value if an object is relevant or not. This binary classification is necessary as none of the evaluated technologies supports priorization of channel items. Most provider's search interfaces do not offer a restriction to *new* (may be since a specified date) objects. The query is done against the whole database. The alerting service has to keep track on which objects the user was already notified.

This leads to a principle problem with channel technology in general: How does the supplier know if the pushed event is noticed or handled by the client? Information on new objects has to be kept for at least a specified time. But nobody wants to get notified multiple times on the same object. On the other hand information must not get lost by overwriting it. Whereas this problem was avoided with e-mail submission, a solution is still missing with channel technology

One drawback of the Microsoft and Netscape approaches is that they are still bound to certain client software, but since both are open standards we expect that either one or both will prevail and consider this as a minor problem.

The proposed alerting service architecture requires that cooperative as well as non-cooperative providers are known to the service. This includes a knowledge of the interfaces. In the WWW each day new providers emerge which can be a valuable source for the users. The alerting service should automatically detect such providers, evaluate their offers according to the profiles and adopt itself to the provider's interfaces. Intelligent Agents is a promising technology to extend the features of the alerting service.

This study resulted from the first phase of a project that implements an alerting service for the digital library of the Freie Universität Berlin. We are currently in the stage of designing the software system. Our next step is the implementation of the service. Cooperation with a publishing house is planned. To deal with non-cooperative providers, the HyperNavigator [2] will be used.

References

[1] Crespo, A.; Garcia-Molina, H.: *Awareness services for Digital Libraries.*, In: Carol Peters, Costantino Thanos (Eds.): Research and Advanced Technology for Digital Libraries. First European Conference, ECDL '97, Pisa, Italy, 1-3 September, Proceedings. Lecture Notes in Computer Science, Vol. 1324, Springer, (1997), 147-171.

[2] Faulstich, L. C.; Spiliopoulou, M.: *Building HyperNavigation wrappers for querying publisher sites.* Submitted.

[3] Gebhard, F.: *Berührungspunkt zwischen Information Retrieval- und Experten-Systemen*, Working-papers of the GMD 167, Gesellschaft für Mathematik und Datenverarbeitung (1985)

[4] Gravano, L., Chang, K., García-Molina, H., and Paepcke, A.: *STARTS: Stanford protocol proposal for Internet retrieval and search.* Number SIDL-WP-1996-0043. Stanford University, August, (1996) at `http://www-diglib.stanford.edu/cgi-bin/WP/get/SIDL-WP-1996-0043`

[5] Hanson, E. N.; I-Cheng, C.; Dastur, R.; Engel, K.; Ramaswamy, V.; Tan, W.; Xu, C.: *A flexible and recoverable client/server database event notification system.* The VLDB Journal 7 (1998), 12-24

[6] Moen, W. E. (1995). *The ANSI/NISO Z39.50 Protocol: Information Retrieval in the Information Infrastructure.* Bethesda, MD: NISO Press. at `http://www.cni.org/pub/NISO/docs/Z39.50-brochure/50.brochure.toc.html`

[7] Möller, G.: *Diplomarbeit Benutzerprofile für das Information Retrieval: interne Repräsentation, Erstellung und Visualisierung*, Carl von Ossietzky University of Oldenburg, Department of Computer Science (1996)

[8] Odlyzko, A. M.: *Tragic loss or good riddance? The impending demise of traditional scholary journals*, International Journal of Human-Computer Studies 42 (1995), 71-122

[9] Paepcke, A., Chang C. C., Garcia-Molina H., Winograd, T: *Interoperability for Digital Libraries Worldwide.*, Communications of the ACM, 41(4), (1998)

[10] van Rijsbergen, C. J.: *Information Retrieval* Butterworths, London, 2nd ed., (1979)

[11] Salton, G.; Fox, E; Wu, H.: *Extended Boolean Information Retrieval.*, Communications of the ACM 26 (1983), 1022-1036.

[12] Verhoeff, J., Goffman, W. and Belzer, J.: *Inefficiency of the use of boolean functions for information retrieval systems*, Communications of the ACM, 4, (1961), 557-558, 594

[13] The London Parallel Applications Centre: *The London & South-East Centre for High Performance Computing* at `http://www.lpac.ac.uk/SEL-HPC/`

[14] Springer: *Link – The visionary information service.* at `http://link.springer.de`

[15] *Elsevier Science* at `http://www.elsevier.com`

[16] Elsevier: *Science Channel* at `http://channels.reed-elsevier.com/ScienceRTW/ElsevierScience/docs/toolbar.asp?\_FileName=welcome.htm`

[17] Medoc: *Ariadne – the red thread trough the web* at `http://ariadne.inf.fu-berlin.de:8000/index.html`

[18] *Object Management Group* at `http://www.omg.org`

[19] Object management Group: *Corba Event Specifications* `http://www.omg.org/corba/csindx.htm`

[20] *Netscape* at `http://netscape.com`

[21] Netscape Communications Corporation: *Netcaster Developers Guide* at `http://developer1.netscape.com/docs/manuals/netcast/devguide`

[22] FIZ Karlsruhe: *CompuScience* at `http://www.fiz-karlsruhe.de/stn/Databases/compusci.html`

[23] Microsoft: *Press Release* March 1997, at `http://www.microsoft.com/corpinfo/press/1997/Mar97/Cdfrpr.htm`

[24] Microsoft: *Webcasting in Microsoft Internet Explorer 4.0 White Paper* (1997) at `http://www.eu.microsoft.com/ie/press/techinfo-f.htm?/ie/press/whitepaper/pushwp.htm`

[25] Marimba at `http://www.marimba.com`

[26] PointCast at `http://www.pointcast.com`

[27] BackWeb at `http://www.backweb.com`