

Hybrid Archives

Preserving Resources that are Frequently Updated

Document Details

Document type: Project Report
Author: Derek Sargent
Draft / Version: Version 1.0
Date of edition completion: 12/11/2004

Contents

Contents	2
Introduction.....	3
Traditional Methods.....	3
User Data Backup	3
SQL server database	4
Newer Technologies	4
Planning for Preservation.....	5
Key Factors	5
Metadata	7
Implementing Preservation of Frequently Updated Resources	9
Determining updates.....	9
Transferring Updated Sections	10
Resource Preservation Management Metadata	10
Managing Versions of Collection Level Metadata.....	11
Conclusion.....	11

Introduction

Much of the information that can be obtained today will not be available tomorrow. This is not the result of negligence or system failure; it is simply due to change. Updates to digital resources come from a variety of places: the owner of the resource ensuring that it is current; a third party changes information that is referenced by the resource; or a user is allowed to add information. (For example, bus timetables, teletext pages, e-auction websites.)

The types of dynamic resources which particularly interest the AHDS are image collections, learning resources, and statistical datasets. Traditional full digital deposit is poorly suited to dynamic web-based resources, particularly where the data and functionality are interwoven. The following section discusses some of the IT methods that could be applied to preserving frequently updated resources. Key factors involved in planning and implementing a successful strategy are highlighted.

Three resources will be used to focus the discussion:

The Corpus Vitrearum Medii Aevi (CVMA)¹: creating a digital archive of approximately 30,000 images of medieval stained glass.

PeoplePlayUK²: an Internet resource based on 1,500 images from the Theatre Museum, with activities, structured learning packs, and links to performance venues in Britain.

netflag³: a modern piece of web art that is created through interaction with the users. This is hosted as part of the Guggenheim variable media initiative.

Traditional Methods

Managing files and computer systems that are frequently updated is a task that has faced IT departments for decades. Whilst this has not been with intention to preserve the resources, many of the techniques employed can inform a preservation implementation. The older, more established, methods are introduced first as these contain the basic concepts for the foundation of the more modern techniques.

User Data Backup

Where an institution has its own intranet, offering filespace to several users it is critical to have a backup and recovery strategy. Within the institution each user will have different usage patterns and requirements for types of data. The scope of the varying user differences will depend on the nature of institution and its divisional structure, for example a financial company may have one primary user class which deals with the bulk of data and data usage is consistent across this class, and also have a smaller group of sales user with more varying usage but within less data. A university is likely to have a much wider variety of user and data usage, typically with fewer users from computing intensive research departments generating and modifying large volumes of data.

These institutions identify their key file servers and other critical data storage hardware, and implement a backup and recovery strategy. At regular intervals (weekly or fortnightly) a full dump of all user files is made to magnetic tape. To allow for re-creation of files with better granularity than the file as it was the weekend before (or the weekend after) the date required, incremental backup to tape is performed each evening. Incremental backup checks the modified timestamp on all of the files and only writes those files which have changed. Scheduled full and incremental backup cycles are co-ordinated using a vendor supplied database system, so that a magnetic tape containing the latest copy of the file for a particular evening is quickly locatable. The incremental dump is quicker than a full dump as fewer files are written. Normally, because of the number of magnetic tapes used for a full cycle (the

whole week) of incremental backups, these tapes are re-used after a few cycles (maybe every two months). This does mean that the extra granularity for recovery is lost after this period. Modern products also offer a differential dump instead of an incremental dump.

SQL server database

Relational databases can monitor the changes made to the data in a transaction log (TLog), which records enough metadata to redo the transaction from an earlier full dump of the database. The diagram indicates how a database is backed up. A differential backup allows the sections of the database that have changed since the last full or differential backup to be dumped.

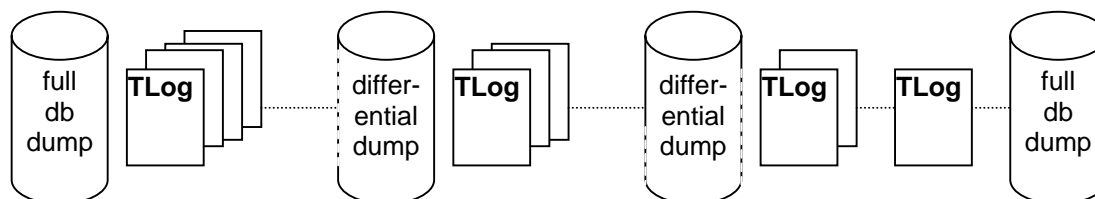


Figure 1: Keeping a relational database backed up

To restore the database to its state at any point in time, the database is restored to the latest full backup earlier than that point then the database is updated according to any subsequent differential backups and final each TLog entry is applied in turn. There are two very good reasons not to do a full dump of a database too frequently: this requires a large amount of data storage; and the performance of the database degrades while the dump is running, which can provide a headache for choosing which time period should suffer (especially for international 24 hours a day services).

Newer Technologies

Often there is a need to make sure a computer system is current by transferring the state from a remote system. One example of this is remote synchronisation, say between a laptop and office workstation particularly within a corporate business setting. Another example is uploading a website to the webserver host. There are many vendor solutions for synchronising remote systems.

Peer-to-Peer (P2P)⁴ technology is used for the collaborative applications market, and is particularly famous for its file sharing capabilities. The P2P model allows various third parties to plug their files and applications into a network for access by others. Aside from the potential for lack of information integrity within P2P, it has clearly been successful at maintaining catalogue metadata for the files shared.

Grid technologies⁵ are used to bring together distributed geographical resources. Of particular relevance to preserving frequently updated resources is its use of distributed digital data sources. Data management⁶ technologies are used to maintain dataset integrity, although most of the current thrust is focussed on metadata rather than the dataset itself (Edinburgh's use of XML for grid data management⁷ of extremely large datasets may be an exception). Many confuse P2P and Grid technologies⁸ but both have ideas to offer for our problem.

Lots of Copies Keeps Stuff Safe (LOCKSS)⁹ is a co-ordinated P2P system that deliberately duplicates files across the internet participants in order to preserve the files.

Planning for Preservation

Assuming that decision has already been reached about the significant properties of a resource are, it is possible to anticipate how the resource will change and the impact of these upon the significant properties.

Using the examples: the **CVMA** collection will change whenever a new image is digitised, and this is likely to be the only cause for update; **PeoplePlayUK** will replace or cycle through its activities, provide new structured learning packs, and review and update links to performance venues; and **netflag** will change whenever a user makes or alters one of the flags.

All of the changes anticipated here might also update the metadata, and the collection level metadata. While the anticipated updates will be the focus for preservation activity there should be some planning for unanticipated updates. For example, collection level descriptive metadata may need to be updated due to a change in the collection's provenance independently of any changes to the components. Another area for caution is the documentation associated with (but not part of) a collection. Within the hybrid model it is important for both parties to discover where such documentation exists.

Key Factors

The focus of the planning for preservation and successful implementation of a preservation system comes down to three key factors:

- the frequency of updates
- the regularity of updates
- consistency of size and scope of updates

The **CVMA** collection is the easiest to assess for these factors. In the five years (1999 - 2004) approximately 12,000 images have been created by digitising the photographs. This is an average frequency of 200 images a month. The regularity over this period has changed, but it is possible to establish what the pattern of updates will be. In terms of consistency of size and scope, the collection always grows by a consistent size for each image added.

PeoplePlayUK is harder to assess: its updates are likely to be frequent and irregular, and probably with little consistency of size. The scope of changes can be predicted though, with only the activities, learning resources, or venue links being altered at any one time.

Napier's **netflag** incorporates statistics that could be used to establish patterns for the key factors. In the three or so years of being online over 65,000 changes have occurred. These are very frequent, but irregular, and highly consistent.

But what are the ramifications of these key factors to the preservation of a digital resource? Clearly the most similar scenario to preserving a static digital collection is when the updates are infrequent, regular, and of a small size. The other extreme is where changes are frequent but irregular and widely vary in size and scope (so many different parts of the resource are affected).

Frequent very small updates that only affect small sections of the resource may be a plus where we need to reconstitute various subsets of the resource. However such updates can make it infeasible to reconstitute the whole resource on the fly. An aspect of frequent updates that has the potential to be overlooked is when several updates apply to the same component of the resource, and the component is updated into its initial condition. Suppose that an automated email notifies AHDS of each update (or the component's 'last modified' timestamp is used), then the archive may perform unnecessary work when reconstituting that component. A seemingly zero change can be produced by simply opening and closing a document (where "would you like to save the changes?" is encountered). Unless the majority

of updates are zero change it is safer to treat these in exactly the same way as other updates, since the behaviour of the resource may be affected by the timestamps of its components.

Regular updates make it much easier to plan and schedule ingest activities and provide sufficient disk space in the staging area. However this becomes untrue when there is a lack of consistency of size or scope. Irregular updates can also be a positive thing as the preservation system receives periods of respite. Where updates are being sampled at regular intervals there may be no apparent difference between regular and irregular. For example, if the **CVMA** images are being transferred to the ingest staging area every month it probably does not matter that the live collection makes the new images available every week or adds them ad-hoc throughout the month (maybe some catalogue metadata is entered manually and requires checking against other information sources). The point here is whether the preserved collection needs to be able to reconstitute the live collection as it appeared at a moment in time, or whether it is sufficient to provide the most up-to-date information (or somewhere between these two, such as reconstitute it to within a fortnight of the date). Ability to reconstitute a collection is not constrained by the sample rate of updates, as there may be metadata associated with the components that indicates when it was added to the live system. This metadata may be implicit (for example the 'date created' timestamp of the catalogue entry, or the image file timestamp) or possible to calculate (perhaps from a metadata entry 'date digitised').

Updates of a consistent size would appear to be the most favourable as this should also mean that the transfer time of the update is consistent, as is the time taken during automated ingest processing. This may not always be the case, depending on the size of the update compared to the overall size of the components that the update modifies. Where a single component is modified, where the update affects 75% of the component (60MB of an 80MB component), it is simpler transfer the whole component than it is to extract 75%, transfer it, create technical history metadata to allow reconstitution of that component, and apply the update. However where the update affects 1% (20MB of a 2GB component) there is a pragmatic gain to only transfer the update and do as much work as possible on the small fragment rather than on the whole component. **netflag** is a good example of where an update only affects a tiny proportion of a component. Updates of varying size will only be problematic when they place too much burden on either the transfer or ingest process, particularly where the proportion of a component modified varies wildly. For many collections, including **PeoplePlayUK**, the majority of updates will be of inconsistent size. The size of learning resources depends highly on the topic and the media used, hi quality streaming video can be very large yet Shockwave animations may be much smaller. This links to the other aspect of consistency, scope. Whether or not an update modifies one or several parts of the collection, and whether updates consistently modify the same parts of the collection and hence leave the rest of the collection unchanged. Again, handling updates that consistently modify distinct parts of the collection will be a pro when the preservation system needs to reconstitute subsets of the resource on the fly.

Updates that were not anticipated will always occur as irregular updates. For example, **netflag** could add new flags to its set of base drawing objects and could also update the information about the meaning of items within the flags. **PeoplePlayUK** could run some special summer activities or allow theatre companies to include special promotional material.

Part of planning for the preservation of frequently updated resources is estimating, across the whole resource, just how large the resource will become and how the change comes about. The key factors give an indication of how dynamic the resource is. If the resource has been in dynamic operation before AHDS starts to preserve it, then useful indicators are:

- How large is the resource now?
- What patterns of update has it already undergone?
- How large did the resource use to be?
- Are any major changes planned?

Whilst gauging the resource in relation to the key factors and looking at the indicators is helpful and provides the basis for agreement and schedules to be made with the third party,

these should be flexible to allow for review based on the update pattern observed in practice while preserving the resource. Provision should also be made for unanticipated updates.

There will be situations where it is not feasible to collect very frequent updates for the preserved collection. Some of the eScience datasets on the UK Grid⁷ are truly large and can be frequently updated by several institutions simultaneously. When it is not feasible to collect all the updates a value judgement must be made, based on the relative significant properties, how critical are the changes that occur? As with any decision made about the preservation strategy for a resource, this should be documented and referenced from the preservation history metadata entry.

Perhaps a weekly incremental harvest of the updates with a quarterly full harvest would allow the resource to be reconstituted to any week in time. If the full harvest is too large, but the frequent updates are small enough, then the programme could be changed to take a daily incremental harvest with only a full harvest each year. This would cause problems to reconstituting the dataset and impede access via the AHDS preservation system, but would also preserve the dataset with excellent temporal granularity.

Metadata

Essentially there are two types of metadata to consider, descriptive and technical metadata. However it is also useful to distinguish the metadata for components of a resource from the collection level metadata. The emphasis of this section is not creating and populating metadata at either the component level or the collection level. It is solely on the interaction between frequently updated components, frequently updated component metadata, and the meaningful preservation of the collection.

- Do updates to components of a collection cause any part of the collection level metadata to need modification?
- Do changes to component metadata require modification of collection level metadata?

Updates to component level metadata can be dealt with in exactly the same way as updates to the components themselves. This will depend on how the key factors indicate the component level metadata will change. This is also likely to be related to the updates to the associated component. Every new image digitised for the **CVMA** not only means an update of an additional component but also means an update of a component level descriptive metadata record. (It is unlikely that there will be addition of any technical metadata for this component.) The preservation system will treat any metadata at the component level in the same way as data components. So depending on the key factors the component metadata will be sampled, transferred, and processed through ingest with the same schedule as the data. Following the AHDS policy of migrating data into standard preservation formats, the component level metadata objects may also be migrated into a suitable format for preservation.

Since updates to component level metadata are treated in the same manner as updates to the data, this does not add any complexity to the model, however complications may occur where any update needs to be reflected through a change in the collection level description. Occasionally there may also be a change to the collection level technical metadata. Here are some strategies for updating collection level metadata:

- harvesting from the depositor
- identifying which scope of component updates necessitate collection level metadata update
- creating the collection level metadata on-the-fly

Harvesting a collection level description extracts a snapshot metadata record of the current state of the live resource. If the snapshot is taken too frequently this metadata record may be no different from the previous snapshot, and if there is a large interval between snapshots

some temporal granularity could be lost (and the metadata loses sync with the collection). Creating collection level metadata on-the-fly is the same as harvesting, except it takes place on the archived resource rather than the live and does not depend on the choice of snapshot interval. A potential disadvantage of this on-the-fly approach is that the resource needs to be reconstituted and the collection level metadata extracted before the metadata becomes available. A solution to this is to pre-generate the metadata record during the ingest process. The most efficient approach is to identify which parts of the resource change the collection level metadata when they are updated. Whenever an update to such a component (which could be either a data or metadata component) occurs the relevant entries in the collection level metadata are updated accordingly. The main flaw with this approach is that the scope of component updates that need reflecting in the collection level metadata must be identified correctly, or the collection level metadata will no longer be accurate. These approaches can be improved by combining them, for example component updates that are known to affect the collection level metadata could send a signal to the archive, at which point the archive can harvest a complete collection level description record, and at regular intervals (say monthly) the complete record harvested (even where no updates have been signalled).

Supposing that the collection level metadata is being frequently updated, it will be necessary to preserve the record before and after the change. If the record is small, or most of it has changed between versions, it is simplest to preserve both the old and the new record. If the record is being generated on-the-fly then this question does not even apply. However if a small section of the record is changed it is worth using either the 'incremental backup' or 'differential backup' technique.

The whole issue of technical metadata is simplified by the use of standard archival file formats, although as the resource changes it is necessary to detect the introduction of new formats and select an appropriate format for their preservation. At this point the technical metadata at the collection level may also need updating. If the collection level technical metadata is a list of all the different file formats (including vendor versions) involved in the collection together with information of how to determine which format applies to a component, then updates just require addition of new entries.

The archive also needs to maintain some metadata for the collection that distinguishes between the deposited and migrated versions of a component from earlier and more recent updates to the content of that component. This also applies to component level metadata and collection level metadata as well as to the components themselves. This extra metadata fits into the OAIS reference model as part of the PDI:provenance, although in the working preservation system it would be handled by a database within either data management or ingest management. Such metadata captures the preservation history (or even the history of the ingest process), recording which components were transferred and migrated on which dates, and for what purpose. Some of this metadata may be encapsulated inside the components themselves (for example in the header of a TIFF file for the **CVMA** it is likely that there will be a DateDigitised property). The purpose of the preservation history metadata is to link the component versions to their temporal relevance although it may fulfil a dual function for auditing of the preservation process (within OAIS Preservation Planning).

Where the scope of a collection has changed over time and the archive holds rich metadata describing the change over the time period, it may be beneficial to present this information to the user. For some resources this will not be necessary as the most recent version contains all that an earlier version had plus some more. For these resources a single catalogue entry which reflects its entire state is ample. Other resources will need a mechanism to create editions based on the amount of change that the updates have caused. A special catalogue interface is then needed to allow the users to look at the state of the resource at a particular point in time. The Internet Archive: Wayback Machine¹⁰ is an example of one such interface that allows webpages to be viewed in a historical state¹¹. It should be noted that it is not always necessary or desirable to present an interface to the fine granularity that is preserved in the archive. The interface may only want to present access to the full snapshots, or regular intervals rather than to each ad-hoc update.

Implementing Preservation of Frequently Updated Resources

There are a variety of approaches to implementing the data transfer and collection management for frequently updated resources. Since a fully automated implementation is needed, all of the approaches discussed assume network connectivity (although in practice physical media are still widely used to transfer data¹²). At CNES Huc and Boucon have done some leading edge research into standardised ingest¹³, and cover some issues that are not addressed here. Sub-dividing the workflow for preservation of frequently updated resources indicates four crucial tasks:

- determining when updates have occurred, and where
- transferring updated parts to the ingest staging area
- generating collection management metadata to facilitate resource reconstitution
- managing versions of collection level metadata

Determining updates

One method to determine when updates have occurred on the depositor's system is to have a notification system. If practical, this could be hooked into the package that the depositor uses to make the update or triggered by a modification of the catalogue database. An alternative is to have a background task which periodically recursively lists the file details within the resource, and notifies the archive of any changes. The archive can deal with each notification separately, although if this response is immediate it could place an unwanted load on the depositor's system and network; or the archive can build a profile of notified updates and act on this according to a schedule.

If an agreed schedule of updates exists then the Ingest system can use this to determine which updates to expect. Whilst this should not need any notification system, it is necessary to have something in place in cases where the schedule cannot be met.

Rather than relying on the depositor to provide notification of updates, the archive system can perform regular remote probing of the depositor system. Probing can vary from being able to obtain file details within the live resource to using ODBC to query a metadata database, or even using a web-spider to crawl through the resource (the University of Michigan used HotBot to harvest metadata¹⁴). The web server on the depositor system can easily be set up to only allow the archive to use a web-spider, and the web-spider can be set up to harvest whatever information is desirable to determine updates.

It must be possible to determine when updates have occurred in the component level metadata as well as to the components. There may also be some non-HTML components which have changed, or some components where the change is not evident: for example some data sources have been updated, but not the scripts that collate them for presentation.

There are several methods available for notification, with the primary being either a message to a server listening on a socket (using a protocol like HTTP) or an email (in a similar manner to using email to configure LISTSERV). Using email does not require any installation of special software, but it relies on open relays and is not direct or instant. While a listener on a socket is instant, there is a growing risk that any open socket is targeted for denial of service or even system hijack attacks.

A final method of notification is where the depositor alerts the archive of an update by actually sending the file that contains the update. When the depositor makes an update, that system pushes the file straight across the network into the ingest staging area. Suppose that a resource has a small enough user base, it might be possible to take snapshots based on when a user accesses the resource. This would be especially useful when a user analyses a dataset, and needs to be able to replicate the analysis at a later date. Each time users access the live resource, the archive can be notified and trigger the harvesting of updates. If the

preservation is reconstituting the resource on-the-fly some usage history metadata (from both the live and the preserved resource) can be used to facilitate intelligent caching.

Transferring Updated Sections

Most FTP clients contain built in facilities to allow downloads across a number of sessions. This facility can be adapted so that any fragment of a file can be downloaded, although this is only practical where the method for determining the occurrence of updates is intelligent enough to provide detailed information about updates of fragments. If the whole file is downloaded, then this can be compared to the previous version and any unchanged sections can be discarded (this is discussed below). Databases and datasets may suit ODBC for transfer in place of FTP. There are also secure tunnelling data transfer tools (such as SFTP¹⁵) where standard FTP might not be viable through certain firewall policies.

Whether download of fragments is necessary or not depends on the type of approach being used to preserve the resource at different points in time. Both the incremental and differential approach (for component level updates) only need the fragments, whereas a full snapshot requires the whole file. It is still possible to apply the incremental and differential when only whole files are being preserved, although this will mean that more data is being stored than is necessary for the approach being used.

Resource Preservation Management Metadata

In order to facilitate the reconstitution of a resource for a point in time or a particular version, special metadata must be generated and preserved. Several of the products available to handle versioning of files and projects contain internal metadata to do just this, although it is usually held in a bespoke and deeply enconced format. An example of such a product is CVS (Concurrent Versioning System) which is normally applied for controlling major source code development, where several versions of the software may be distributed at the same time. Indeed OAICAT¹⁶ is a development that utilises CVS in this way. For the kinds of resource dealt with in this report, stubs referencing preserved component versions could be placed into CVS and then the CVS used to determine which versions of preserved component are needed for any reconstitution of the resource.

Rsync¹⁷ is a tool which provides a fast method to bring remote files into synchronisation. Although this is more designed to allow central desktop replication to laptops and remote workstations, Rsync could be modified to control and determine the transfer of updated components. Rsync is capable of synchronising a whole directory tree, and works by transferring any differences in the files. This tool has some very strong points to offer a hybrid archives model: it is open source, and can communicate using a secure remote connection (ssh). There are other utilities designed to replicate a remote system, such as Mirror¹⁸, although it is unclear if this can be adapted for preservation purposes. Some resources may not fit into the model of a hierarchical set of files, say a massively distributed eScience dataset, and co-ordinating an Rsync approach may become too complex. The logs of differences transferred are stored as preservation management metadata.

A special system is needed for the storage, manipulation, and retrieval of this extra preservation management metadata (recording the history of updates and versions). This system may be simply a single table within a SQL server database. In this case the database can use TLogs, which can be applied to calculate the temporal state of the resource at any point in time. Such a database would be easy to maintain and preserve. Alternatively, CVS could be employed as the underlying system for managing any preservation management metadata – this would require the crafting of an interface to map between CVS and a representation convenient to the preservation system.

Although it may be possible to generate very detailed metadata about the occurrence of updates and differing versions of components, it may not be technically feasible to reconstitute the resource with the same fine granularity. Limits on the strategy used for reconstituting the resource are: data storage costs, as for extremely large and dynamic

collections it may be too expensive to even store the differences for every update; network bandwidth, as more updates may occur before a large data transfer is completed; and the computational power of the system used to reconstitute the resource, either in the ingest staging area or on-the-fly from the preserved resource. The length of time needed for harvesting and ingest processes may also have an impact on the preservation approach.

Managing Versions of Collection Level Metadata

When the collection level metadata requires updating there must be an automatic way of generating the changes. This record can be harvested from the live (depositor's) collection or from the collection on the archive machine (either in the ingest staging area or fully preserved) or changed according to applying a set of rules. Once the new collection level metadata record is obtained, this can be compared to the previous record (using Rsync, winDiff, etc). Depending on the size of the record and the proportion of difference, a full snapshot can be made of the record or an incremental style backup made.

Again, a single database table together with appropriate backup can be used to preserve the collection level metadata throughout its temporal versions. If the records are small it may be pragmatic to maintain a snapshot of the records that line up with any special catalogue interface for users accessing the resource in its different versions.

Where the relationship between component level changes and the collection level description is utilised, there may be no need to record versions of the collection description as well as the component level changes. The collection level description could be constructed from a basic metadata template, containing some static elements such as a title and description, and drawing the rest of the description from the components which exhibit relevant change. This could be particularly suited to areas like spatial and temporal coverage, subjects, and keywords. The main caveat is that the relationship between the component level metadata and the collection level metadata must be well understood and accurately reflected in the template.

Conclusion

For large digital resources that undergo frequent changes, it is not feasible to capture a full snapshot for each change. Such an approach would not only need prohibitively large storage space, but also high network bandwidth. Looking at some traditional information technology methods shows that there are several products and solutions that can be adapted to aid the successful preservation of these resources.

Examination of the key factors: frequency, regularity, and consistency of size and scope of the updates allows the best strategy of capturing the resource to be applied. A high proportion of the successful preservation of these resources is down to the extra metadata generated and the preservation of the collection level metadata. Web-crawlers can be programmed to automate harvesting (of data and metadata), but they may not be adaptable to resources that are not based on web technology. However, it is ideal to deploy a consistent approach using client-server technology over the network whichever resource is being preserved.

When the collection ceases to be updated one of two things happen, the collection (including all the earlier versions) becomes static, or through advocacy the collection is given new lease of life and updates.

Issues will always exist about the unpredictable nature of the future, as uptake of new technology and file formats may simplify the extraction of metadata and the overall preservation. Data storage costs are still falling, although the growth rate of resources is still increasing.

The closer digital preservation moves to the creation of the resource, the more chance it has of being successful. For resources that undergo frequent change this means bringing the



preservation closer to each change. While it is still not possible to preserve every change to every resource, it is possible to identify and preserve the critical changes in a timely manner. This means that digital preservation is no longer limited to static, no longer current, resources. Dynamic and adaptive sections of our digital heritage can be given true longevity.

- [1] The Corpus Vitrearum Medii Aevi <http://www.cvma.ac.uk/>
- [2] PeoplePlayUK <http://www.peopleplayuk.org.uk/default.php>
- [3] Guggenheim Internet Art http://www.guggenheim.org/internetart/internetart_index.html
- [4] O'Reilly Open P2P <http://openp2p.com/>
- [5] Grid technologies <http://www.qub.ac.uk/escience/gridtech.php>
- [6] Grid data management <http://www.gridpp.ac.uk/datamanagement/>
- [7] Edinburgh grid data <http://www.epcc.ed.ac.uk/DIRECT/grid/node43.html>
- [8] Grid computing Info center <http://www.gridcomputing.com/>
- [9] LOCKSS <http://lockss.stanford.edu/>
- [10] Internet Archive: the Wayback Machine <http://web.archive.org/>
- [11] Cedars, wayback http://web.archive.org/web/*/http://www.leeds.ac.uk/cedars
- [12] *Survey of existing data harvesting/transfer techniques and tools*, Tony Austin
- [13] *Producer-Archive Interface Methodology Abstract Standard White Book*, 2002, Danielle Boucon & Claude Huc http://us-vo.org/pubs/files/CCSDS_651_W2.pdf
- [14] OAister <http://oaister.umdl.umich.edu/oaister>
- [15] SFTP <http://www.openssh.com/>
- [16] OAICAT <http://www.oclc.org/research/software/oai/cat.html>
- [17] Rsync <http://samba.anu.edu.au/rsync>
- [18] Oblivion Mirror <http://home.in.tum.de/~jain/index.html>