HistorySystem

New History System - old page preserved below

- THIS PAGE IS OBSELETE ***

This page has been superceded by HistorySystemPrototype

Plan for New History System for DSpace

After examining the state of the current

```
HistoryManager
```

and the data it produces, I'm inclined to discard
it and write a complete replacement.
Since the PLEDGE project needs
a functional history system, we are motivated to rebuild it
in time for the next major release (1.4).

The next few sections describe my strawman proposal for a new DSpace
History System. The original contents of this page are preserved below,
and contain some different perspectives on the design of a history system.

NOTE: This is an initial sketch of a proposal, it is not yet even
complete, but I want to put it in the wiki as early as possible to
gather responses – so please add your comments in the "Comments" section
below.

Goals

- Preserve a fixed, unchangeable record of all significant changes to those data objects in a DSpace repository that are to be preserved.
  - Only consider changes to

    ```
    Item
    ```

    objects.
  - "Changes" include creation, add/remove bitstreams, changes to Item-level metadata, changes to bitstream-level metadata, withdrawl /reinstatement, deletion.
  - Associate the person responsible with each change event.
  - Record only such details of each event as are really necessary for provenance, e.g. new metadata values, but not bitstream contents.
- Give access to the history data, by Item and through free-form queries.
- Carry the relevant history data with an Item when it is moved from one DSpace repository to another.
- Migrate whatever data can be reclaimed from the old DSpace history system.

Non-Goals

- This is not a versioning system, so it does not usually attempt to record the substance of a change (e.g. contents of bitstreams).
- Do not record any events that do not result in changes to the archive, so the history will not have records of disseminations.
- Do not record changes that are only relevant within the archive, such as authorization policies, EPersons, Community and Collection hierarchies, etc.
- Only record history data for objects that are in the archive, which means history is not recorded for Workspace and Workflow objects.

Justification

If the purpose of saving history data is to establish the provenance of
objects archived in DSpace as an aid to future preservationists, it makes
sense to only save data about the objects which are to be preserved.
This excludes transient object (e.g.

```
WorkspaceItem
```

,

```
WorkflowItem
```

,
and objects that are only meaningful in the context of their home
DSpace archive:

```
Collection
```

,

```
Community
```

, and especially

```
EPerson
```

.
Note that the grouping of Items represented by a Collection can also
be expressed in the Items' metadata, a much more "preservable" manner.

Since preservation probably means transferring (or copying) Items to another
repository or archive at some point in the future, there should be a way
to include the relevant history data with each Item.

History Data Model

After examining the goals, the existing system, and similar designs such
as the PREMIS Event model, I've outlined this data model for
representing DSpace object history.

It is based on the current version (3) of
ABC Harmony as an RDF ontology.
The ABC Harmony model is reasonably close to what
I had in mind, and using an existing
ontology lets us leverage its documentation and examples.

The first-class objects are:

*__Item__, corresponds to the DSpace Item identified by a certain persistent identifier (i.e. Handle).
*__Event__, describes a change to another object.
*__Archive__, identifies a single instance of a DSpace installation.
*__EPerson__, names an individual agent responsible for an event.

There are also two "second-class" objects: Since these describe aspects
of Items that can change over time, they only appear in the context of
an Event that creates or modifies an Item.

*__Bitstream__, describes the aspects of a DSpace Bitstream object relevant to provenance.
*__Metadata__, a metadata item attached to an Item in the DSpace object model.

Item

The Item is a holder for the DSpace Item's only immutable property,
its persistent identifier (i.e. Handle). An Item is identified by
its Handle so it has the same identifier on every DSpace archive
where it is present.
It has the RDF type value

```
dspace:Item
```

,
which is a subclass of

```
abc:Actuality
```

.

An Item's URI is its persistent identifier in URN syntax, e.g.

```
hdl:123456789/123
```

.

Archive

An Archive identifies the DSpace instance in which an Event occurs.
It is identified by the World Wide Web URL of its top-level page, e.g.

```
http://dspace.mit.edu/
```

.
It has the RDF type value

```
dspace:Archive
```

, which is a subclass of

```
abc:Actuality
```

.

It also may contain the following Dublin Core properties:

*__dc:title__ - Descriptive name of the archive, e.g. "Miskatonic University Digital Archive"

NOTE: This is just my naive interpretation of DC.
Metadata mavens and Dublin Core critics are invited to correct my
usage of these DC elements.

EPerson

An EPerson identifies, as precisely as possible, an agent authenticated
to the DSpace archive who was responsible for initiating an Event.
Its information may be of limited value for provenance,
since, in the DSpace architecture, an EPerson is defined only within
the context of an archive and is not given a persistent identifer.

The EPerson is identified by a combination of its home archive and
a cryptographic message digest of its attributes that are unique within
that archive (i.e. the email address). Its RDF type is

```
dspace:EPerson
```

,
which is a subclass of

```
abc:Actuality
```

.

It also may contain the following Dublin Core properties:

*__dc:title__ - Personal name in canonical format, e.g. "Jack Florey".
*__dc:identifier.uri__ - "mailto" URI containing email address, e.g. "florey@dspace.org"

NOTE: This is just my naive interpretation of DC.
Metadata mavens and Dublin Core critics are invited to correct my
usage of these DC elements.

Bitstream

Represents a bitstream added to an Item. It includes information that
may be relevant to the item's provenance (to cross-check the contents
of the archive against history later, for example).

A Bitstream only appears as the subject of statements belonging to a
"Situation" that is the result of an Event which created or modified an Item.

*__rdf:type__ is

```
dspace:Bitstream
```

(a subclass of

```
abc:Actuality
```

).
*__dc:identifier.uri__ - The SequenceID of the bitstream, e.g. "#1"
*__dc:title__ - the name attribute of the bitstream, e.g. "thesis.pdf"
*__dc:format__ - short name of BitstreamFormat, e.g. "Adobe PDF"
*__dc:format.extent__ - size in bytes of the contents, e.g. "314592"
*__dc:type__ - type or purpose; i.e. name of the bundle containing bitstream, such as "ORIGINAL".
*__dspace:checksum-algorithm__ - name of checksum algorithm, e.g. "MD5"
*__dspace:checksum__ - value of checksum of bitstream contents, e.g. "6df9d97f2e8f9.."

Metadata

This represents one metadata value belonging to an Item.
When an item has multiple values for the same metadata element/qualifier,
they appear as separate nodes in the RDF model, not as multiple values within one node.

Metadata only appears as the subject of statements belonging to a
"Situation" that is the result of an Event which created or modified an Item.

It may have the following properties:

*__rdf:type__ is

```
dspace:Metadata
```

(a subclass of

```
abc:Actuality
```

).
*__dspace:mdSchema__ - the metadata schema (default is "dc").
*__dspace:element__ - name of the Dubln Core-styled field identifier.
*__dspace:qualifier__ - qualifier of the Dubln Core-styled field identifier.
*__xml:lang__ - language code, e.g. "en".
*__dspace:value__ - value of the metadata field.

Since DSpace metadata is traditionally in Qualified Dublin Core fields,
there is a shorthand for listing these. The Metadata value only needs
the appropriate DC or QDC property, whose value is the metadata value,
and the optional

```
xml:lang
```

property, e.g. (in N3 format):

```
...
abc:contains [ rdf:type dspace:Metadata ;
dc:title "The Little Prince" ] ;
abc:contains [ rdf:type dspace:Metadata ;
xml:lang "fr" ;
dc:title "Le Petit Prince" ] ;
```

Event

The purpose of the history system is to record "events", so the Event is its central data structure. The history record is simply a collection of Events. Each Event represents a change to an Item – although each transaction on the DSpace server may result in more than one Event being recorded.

Each event has the following properties:

- The subject URI consists of the archive's URI followed by a locally-unique identifier for the Event.
  *__rdf:type__ is

  ```
  abc:Event
  ```

  *__abc:hasParticipant__ - value is the EPerson object who was the authenticated user responsible for this change.
  *__abc:creates__ | __abc:hasResult__ | __abc:destroys__ - identifies the type of action, the value is the URI of the affected Item.
  *__abc:atTime__ - Timestamp at which the event was logged, e.g.

  ```
  "Tue Jan 24 17:46:49 EST 2006"
  ```

  *__abc:precedes__ - Optional, this refers to the

  ```
  abc:Situation
  ```

  that results from this action. It is a blank node that describes the details of what was changed.
  *__abc:involves__ - Value is the URI of the DSpace archive in which this event occurred.

Situation

In the
ABC Harmony
model, a Situation descibes the "existential" (i.e. time-varying) aspects of an Actuality at a certain point in time. The alterations to the state of an Item after an Event make up a Situation that the Event *precedes*. (They are connected by the

```
abc:precedes
```

property, which says the Event precedes the Situation.)

*__rdf:type__ is

```
abc:Situation
```

*__abc:contains__ - value is

```
Bitstream
```

or

```
Metadata
```

added to the Item at this time.
*__abc:removes__ - value is

```
Bitstream
```

or

```
Metadata
```

deleted from the Item at this time.

NOTE: There is actually no

```
removes
```

property in the ABC Harmony ontology,
but there is nothing else equivalent so we are taking the liberty of
adding it.

Examples

Here are some examples of event data, as RDF in N3 format:

```
@prefix rdf:     &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    &lt;http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc:      &lt;http://purl.org/dc/elements/1.1/> .
@prefix abc:     &lt;http://metadata.net/harmony#> .
@prefix dspace:  &lt;http://dspace.org/rdf/1.4/history/> .

dspace:EPerson a rdfs:Class .
dspace:Item a rdfs:Class .
dspace:Bitstream a rdfs:Class .
dspace:Metadata a rdfs:Class .
dspace:Archive a rdfs:Class .

# 1. Event describing the creation of a new Item.

# the archive:
<http://dspace.miskatonic.edu/dspace> rdf:type dspace:Archive ;
        dc:title "Miskatonic University Digital Archive" ;
        dc:identifier.uri "http://dspace.miskatonic.edu:8080/dspace" .

# the EPerson

<http://dspace.miskatonic.edu/dspace/eperson/3900328> rdf:type dspace:EPerson ;
        dc:title "Jack Florey" ;
        dc:identifier.other "florey@miskatonic.edu"  ;
        dc:identifier.uri "mailto:florey@miskatonic.edu"  ;
        dspace:Archive <http://dspace.miskatonic.edu/dspace> .

# the Item
<hdl:1721.99/123> rdf:type dspace:Item .

# the Event.  Note that the bitstreams are described as they exist at the
# time of the event, since they might be modified later.

  <http://dspace.miskatonic.edu/dspace/event/20060215233447422> rdf:type abc:Event ;
    abc:hasParticipant <http://dspace.miskatonic.edu/dspace/eperson/3900328> ;
    abc:creates <hdl:1721.99/123>  ;
    abc:atTime "Tue Jan 24 17:46:49 EST 2006" ;
    abc:precedes [ rdf:type abc:Situation ;
                   abc:contains [ rdf:type dspace:Bitstream ;
                                  dc:identifier.uri "#1" ;
                                  dc:title "thesis.pdf" ;
                                  dc:format "Adobe PDF" ;
                                  dc:format.extent "314592" ;
                                  dc:type "ORIGINAL" ;
                                  dspace:checksum-algorithm "MD5"
                                  dspace:checksum "6df9d97f2e8f9.."
                                ] ;
                   abc:contains [ rdf:type dspace:Bitstream ;
                                  dc:identifier.uri "#2" ;
                                  dc:title "thesis.ps" ;
                                  dc:format "PostScript" ;
                                  dc:format.extent "124592" ;
                                  dc:type "ORIGINAL" ;
                                  dspace:checksum-algorithm "MD5"
                                  dspace:checksum "51282c86df9d.."
                                ] ;
                   abc:contains [ rdf:type dspace:Metadata ;
                                  dc:title "Endochronic Properties of Resublimated Thiotimoline"
                                ];
                   abc:contains [ rdf:type dspace:Metadata ;
                                  dc:contributor.author "Issac Asimov" ;
                                ];
                   abc:contains [ rdf:type dspace:Metadata ;
                                  dc:contributor.advisor "Vannevar Bush"
                                ];
                 ] ;
    abc:involves <http://dspace.miskatonic.edu/dspace> .
```

Example: Modifying an existing item by changing a metadata entry,
and adding another:

```
<http://dspace.miskatonic.edu/dspace/event/200602152XY447422> rdf:type abc:Event ;
    abc:hasParticipant <http://dspace.miskatonic.edu/dspace/eperson/3900328> ;
    abc:hasResult <hdl:1721.99/123>  ;
    abc:atTime "Tue Jan 24 18:24:49 EST 2006" ;
    abc:precedes [ rdf:type abc:Situation ;
                   abc:removes  [ rdf:type dspace:Metadata ;
                                  dc:contributor.advisor "Vannevar Bush"
                                ];
                   abc:contains [ rdf:type dspace:Metadata ;
                                  dc:contributor.advisor "Gregor Mendel"
                                ];
                   abc:contains [ rdf:type dspace:Metadata ;
                                  dc:type "Thesis"
                                ];
                   abc:contains [ rdf:type dspace:Metadata ;
                                  dc:date.submitted "1954"
                                ];
    abc:involves <http://dspace.miskatonic.edu/dspace> .
```

Your Comments?

- Cart before horse: The history system work desperately needs use cases to inform it. Suggest should be immediate focus, worry about tech details later.
  *In the absence of anyone with time or motivation to develop use cases, there is still a desperate need for **some** provenance metadata, development is proceeding.*

- Many comments in previous version of page below (which was very recent, December 2005) not addressed.
  - Why not use PREMIS instead of Harmony/ABC?
    Harmony is more complete, extensible, and completely machine-parsable. RDF is easier to repurpose.
  - How to represent situations?
  - Is history data part of AIP?
    Probably yes, see AipPrototype

- Suggest using METS manifest to represent a situation/state of an object
  - Holds entire state of item + bitstreams (including checksums of bitstreams, which can be used to check if bitstreams have changed)
    Might be nice, but too bulky and compute-intensive to discren differences for now.

- "Objects that are only meaningful in the context of their home DSpace archive: Collection, Community" – on what basis are collections determined to be "only meaningful in the context of their home DSpace archive"? Depends on use case?
  True; that is why the archive is now part of the schema of an event, and collections and communities are considered archival.

Old History System page

Initial text by TimDonohue - Please add your thoughts comments, etc.

[I've taken the liberty of copying over comments from JimDowning on the DevelopmentAreas page - TimDonohue]

=== Thoughts regarding a new "history" (aka "audit trail" aka "provenance") system for DSpace: ===

 * Improved history system. Namespace handling fixed, able to persist to database. Benefits include making history system more useful for analysis and less onerous on filesystem (easier backups). I have a out-of-date patch from Jason Kinner at HP, shouldn't be too difficult to merge. Could either require a schema addition, or a standoff database. JimDowning

=== Some "Real-Life" Use Cases ===
 * Scenario 1 [This scenario was mentioned by UIUC legal counsel - TimDonohue]
  * A faculty member has loaded a technical report into DSpace but has chosen (or is required) to restrict access to the local campus for six months. After the six months is up, the technical report is made available worldwide. A patent attorney is interested in knowing exactly when that technical report was made available worldwide because it shows evidence of prior work that would effect his client's (not the faculty member) ability to get a patent.
  * Necessary history information to capture:
   * Date/Time the tech report became "live" in DSpace (i.e. when it successfully completed the Submission Workflow) and what access restrictions were placed on it at that time.
   * Date/Time the access restrictions were lifted and the tech report was made available worldwide.

=== Issues with History System ===

 * How to represent each state of an object in the history system. Currently, the history system stores all DC metadata every time an object is changed. This could result in a large amount of data being stored. It may be better to represent each state of an object as a checksum/digital signature/secure, signed timestamp of the AIP (either just the METS manifest or some combination of all the constituent parts); however, for this to be meaningful, one also would need to store each revision of the AIP (or manifest) so that the checksum/signature could be verified. Not sure if the History system itself is the place to store these revisions, though maybe it is.

 * Getting the right information:

   * Identifying significant events. Currently every HTTP transaction that changes an object is considered an event which is represented as an event in the History system. This may result in data being excessively loaded with events.   Ideally, one event may consist of e.g. the actions of a curator in one session. A format migration is ideally one event, not a series of events (new format file added, changed metadata X, changed metadata Y...)

   * Getting information about a user's intent. When a user performs an action, they are probably doing it with a particular purpose (e.g. 'format migration'). However, all the system gets is a set of HTTP transactions, and all the DSpace content management component gets is a group of method calls. Right now it's the content mgmt component that invokes the History System but this has no way of knowing the user's intent. A very tricky one, this.

 Underlying the above two points is the fact that <code>item.update()</code> invokes the History System to update its records. <code>item.update</code> is essentially a database-level operation, rather than a semantic event. Concrete manifestations of this problem are:

   * Every change made to in-progress submissions (prior to licence granting and submission workflow) is recorded. This is superfluous information -- the item essentially comes into existence as far as the History System is concerned when the deposit licence is granted by the submitter (or some other ingest process completes).

   * Likewise, every click of a workflow review (which causes <code>item.update()</code> to be invoked) also prompts a History System event, which results in lots of useless data.

 * Serialise history system data in the METS-based AIP (or as an mdRef if its non-validatable RDF/XML), so that it is also part of the digitally signed object.

 * The [http://www.oclc.org/research/projects/pmwg/ PREMIS Data Dictionary] offers a simpler and more focussed data model than the [http://metadata.net/harmony/ABCV2.htm Harmony/ABC model] currently used by the History System, and since we're using PREMIS in the METS profile it reduces the number of data models/schemas we need to use.

 * Migrating existing History System data to an updated system. May be tricky.