The History Component of the DSpace™ Institutional Repository

Jason Kinner
Hewlett-Packard Company, Princeton, New Jersey, USA
Mick Bass, Hewlett-Packard Company, Loveland, Colorado, USA

Abstract

Systems which provide a basis for digital preservation within a trusted digital repository must be able to systemically generate and record information about significant activities undertaken upon materials under stewardship. This paper describes the history component of the DSpace™ institutional repository system, which offers an information model suitable for recording such information, and generates metadata which describes how the content and metadata associated with works within the system change over time.

DSpace[1,2,3] is an open-source software platform originally developed by HP Labs and MIT Libraries that provides the basic functionality required to operate an institutional digital repository. The system is intended to serve as a base for future development to address long term preservation and access issues. An instance of the system has been deployed in production at MIT Libraries since November 2002, where it serves as the basis of MIT Libraries' digital preservation service offering. DSpace has included a history component since its deployment at MIT and release as open-source software. The DSpace history component has since been extended as part of the SIMILE Project[3,4] to strengthen the underlying information model, and to include networked access and query capability.

This paper describes the history component of the DSpace institutional repository system: the motivation for the history component, the functionality of the history component; the information model which underpins the metadata which it generates; architecture and design tradeoffs encountered during its development; lessons learned having undertaken the work to date; and areas for future work.

Motivators for History Metadata, Audit, and Data Provenance Capabilities

A trusted digital repository is one whose mission is to provide reliable, long-term access to managed digital resources to its designated community, now and in the future.[5] One definition of digital preservation is "the managed activities necessary for ensuring both the long-term maintenance of a bytestream and continued accessibility of its contents".[5] Reliable digital repositories manage preservation activities by following documented policies and procedures, and enable materials to be disseminated, as authentic copies of the original or as traceable to the original.[5]

An important characteristic of systems which underpin a trusted digital repository is the capability to generate and record information about significant activities performed upon materials within the system. Active curation may entail transforming works in ways consistent with the preservation mission of the digital repository, and in support of the needs of its designated community. Digital preservation activity often entails creation of a range of artifacts derived in whole or in part from the original, thus creating networks of related artifacts. Artifacts in this network may be created directly by those administering the archive, or by a range of external services. When information is held digitally and is easily modified, it is valuable for systems to create sufficient information to establish this chain of events, and to serve as a basis for answering questions of data provenance.

Functionality

The history component of DSpace generates metadata which describes how the content and metadata associated with a work within the system changes over time.

This metadata enables a range of possible additional capabilities, from audit of the administration of the archive, to supporting root-cause analysis of preservation issues, to supporting human-moderated rollback of materials under stewardship.

The history component is invoked whenever events of archival interest occur within the system (for example, when a community is created, an item's instance metadata is edited, or the members of a collection are modified). The history component produces metadata which models "snapshots" of the primary information objects within DSpace (e.g. communities, collections, items, etc.) at various points in time, as well as the situations and temporal events that relate these snapshots. Defining which events within the system are of archival interest is an important curatorial decision undertaken by institutions offering trusted digital repository services. The history component is then invoked whenever these events occur, at which point history metadata is systematically generated.

The information in DSpace history snapshots is recorded using open standards. A key feature of the History Component is that these changes are noted using the Resource Description Framework.[6] The generated history snapshots are graph-oriented, and are usable outside the DSpace system by semi-structured data manipulation toolkits such as the open-source Jena toolkit[7] created by HP Labs' semantic web research team.[8]

Information Model

The history component information model comprises schemas, each specified using the RDF-Schema[9] schema definition language, from three sources: First, the ABC Ontology from the Harmony International Digital Library Project[10] – intended as a base ontology incorporating a number of basic entities and relationships common across other metadata ontologies including time, object modification, agency, places, concepts, and tangible objects.[11] Second, the Dublin Core Metadata Element Set[12] – which models a useful common set of descriptive and bibliographic metadata elements. Third, a DSpace Object Model – which models the content, structure and related metadata elements of materials within a DSpace system. Taken together, the history snapshots provide a time-based record of significant changes to the DSpace corpus.

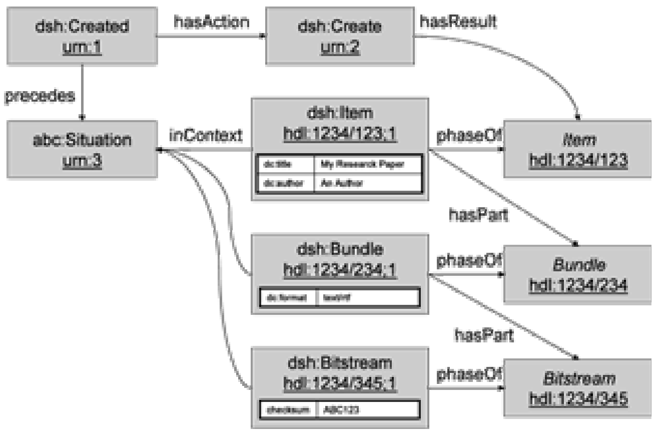

Figure 1. An example history model

As an example of the operation of the information model, consider a fictional research paper that is submitted to DSpace. When the paper is submitted to DSpace, a Created event occurs with the associated Create action that indicates what has been created. In particular, the result of the Create action is a complete Item with associated Bundle and Bitstream (see Figure 1).

All these created items have associated properties that are represented in the History System model of the event. The resting state of the model is that there is an Item, Bundle, and Bitstream (the result of the Create action) as well as associated Item, Bundle, and Bitstream revision instances that represent the current state of the properties of these created objects. Each of these revision instances are in the context of the initial check-in Situation. Note that the revision instances are not related via the hasPart relationship, as the resources themselves are.

The example so far reveals only a static model. Consider what would happen to this model if a property of the Item were to change. The History System represents this change as additional objects and property values in this model (see Figure 2). The change requires the Item to have a new revision instance created (indicated in the figure by the URI, hdl:1234/123;2) that contains the new value for the Dublin Core title property. An administrative property, phaseOf, links this new revision instance back to the root Item resource.

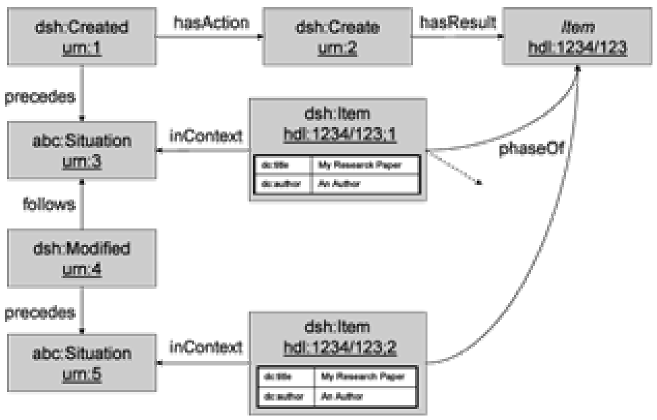

Figure 2. An additional event in a model

An important advantage of modeling history metadata using RDF-Schema and RDF is the ability to easily incorporate into the information model to metadata generated by downstream services. That is, the events and situations modeled by DSpace history can easily include information generated by external services, as well as information generated by the DSpace system itself. The basic concepts of events and situations defined by the ABC Ontology provide the conceptual basis for integration of these separate information sources, while RDF facilitates merging metadata from multiple disparate schemas. The distinct modeling of Dublin Core from the DSpace Object model is an example of this capability.

Architecture and Design Choices

The DSpace history component comprises:

- the write system, part of DSpace;
- the read system, a stand-alone Joseki instance;
- the Harmony ABC RDF Model; and
- the DSpace Object Model.

History System Data Integrity

One important design decision is that the History System need not be up-to-date with the DSpace archive. In other words, if the History System were to be queried for the latest information about an object in the most recent situation, the information returned is allowed to conflict with the DSpace storage mechanisms that store the up-to-date state of an object. This design decouples the History System from the on-line storage supplied by DSpace. Decoupling the two systems allows the Content Management layer of DSpace to determine what events are worthwhile to store in the History System.

Asymmetric Read/Write Interface

The History System provides its services to clients not through the typical DSpace user interface, but through a service-oriented API known as Joseki.[7]

This API allows systems to use RDQL[5] to pose queries against the History System RDF model. This mechanism is the only way to read the model produced by the History System and does not provide a way to update the model.

Similarly, the History System API used within DSpace is the only way to write information to the RDF model. Using asymmetric interfaces provides a level of protection of the DSpace data, making it difficult for unauthorized persons to alter historical data while still opening access to a broad community who may be interested in accessing this data.

Unified RDF Data Store

Besides simply storing a record of events that involve DSpace-managed resources, the History System must also be accessible to systems that want to ask questions about those events. Before the current work, the RDF models that represented these changes were scattered across a file system, making querying a monumental task. By consolidating these models into a single data store, the complexity of searching is greatly reduced. A single RDF persistent model also makes it possible to perform analysis across different events and objects.

Use of Identification Schemes

The following rules govern the use of various identification schemes as they are used within the History System:

URIs

URIs are used throughout the History System and represent the primary identification mechanism. All other identification schemes used within the History System will represent a subset of this scheme. Using URIs as the primary identification scheme makes it possible to annotate resources using RDF.

URLs

URLs may be used to refer to digital manifestations of a resource that are, in fact, accessible via the URL. An example of this may be a bitstream that represents a document, image, or other tangible digital resource.

URNs

URNs may be used to refer to a resource that does not have a digital manifestation or to a resource whose manifestation is not usable outside of DSpace or the History System. Examples include using an MD5 or SHA hash algorithm to generate an identifier for a bitstream that may not be disseminated by DSpace or an internal identifier that cannot be interpreted outside of DSpace.

Handles

CNRI Handles[8] (Handles) will be used to identify all resources maintained within DSpace. Using Handles will allow resources to be uniquely identified locally, within a DSpace installation, and globally, between DSpace installations.

Handles have the property that they may be resolved using a Handle Resolution Service. Although this feature may be useful in the future, it is not a requirement for the usage of Handles that the Handle be resolvable. In fact, Handles may not generally be resolvable due to issues such as publishing scope and security constraints.

Inferencing Support

Inferencing refers to the resolution of a statement within the context of a set of constraints on a given model. Within the History System, these constraints are stated using RDF-Schema.[3] Unlike other schema languages, RDF-Schema supports only basic statements of inheritance via the concepts of sub-classes and sub-properties.

The DSpace object classes derive meaning from the Harmonay ABC base classes. By using this technique and providing a query system that supports inferencing (Joseki), DSpace enable clients to form queries using only Harmony ABC and/or Dublin Core Metadata Element Set syntax.

Use of RDF Types

The History System makes heavy use of the schemas described above, as well as the schema that describes the classes and properties required to maintain information related to DSpace and to the History System (administrative metadata). The use of RDF types improves the capability of the system to validate and to infer relationships between instance statements during queries.

Lessons Learned

The lessons learned during the production of the updated history component are organized here by functional area.

DSpace History Manager

Extensibility

The present implementation of the History Manager requires recompilation to introduce new DSpace object types. Although the DSpace objects used today are sufficiently general that this should not present a major problem, the use of custom or fine-grained types may make it more difficult to produce valid RDF models of these new objects. Objects that are not subclasses or refinements of existing objects are the most difficult case of extensibility to support. Derivations of existing object types would at least use the existing serialization mechanisms, which would omit any extended metadata but would include metadata associated with the base class type.

Extensibility of a system like the DSpace History System must be approached from several perspectives, including the RDF model used to represent the extension, the RDF schema that defines the type information of the extension, and the serialization model that maps between native types and the RDF representation.

Multiple Objects

It may be convenient to include multiple objects in a single DSpace event, depending on how events are most appropriately scoped. Consider an event such as "Creation of a Collection." This event may comprise more than the simple creation of a Collection object, and may span such activities as creating the first item in the collection, assigning metadata properties to the collection, etc. The current History Manager API supports the participation of multiple objects in an event through an extended API that is currently the underlying implementation of the prior History Manager API. The extended API allows multiple objects to be enlisted into a History Context that represents the present Harmony Situation being modeled.

Event Granularity

The granularity of an event being recorded is extremely fine in the present DSpace system. Although fine-grained history is positive from the perspective of capturing a lot of detail, it is negative because of the event correlation issues in the Harmony ABC model. From a system design perspective, it is difficult to predict the possible queries that a user may make against the History System, but common curatorial events should be modeled and represented as a single event (possibly with sub-events) to make these events more visible.

Query

Existential Assertions

Existential assertions, including negative existential assertions, on an RDF model can be of great practical use. Consider examining an RDF model for the latest situation for a given object. Without negative existential assertions, a query expression cannot be formed to ask, "For a given object, what is the Existential Actuality in a Situation that has a preceding Event but has *no Event following it*?" This inability to determine the beginnings and ends of chains of statements makes it difficult to ask other valuable questions, including schema inheritence questions, which might be valuable to a requesting client.

Inferencing

Inferencing is an essential tool to providing the interoperability of semantic data promised by RDF and the Semantic Web. Dynamic inferencing, in which inferred statements are considered during query, is far more effective than storing inferred statements to the model to be queried. This technique prevents the need to update the statements in storage for every change in the schema defining the inferred statements, and it reduces the size of the stored model significantly. The value to the client is that queries need not understand the full complexity of inheritance in the RDF-Schema model in order to extract useful information. By formulating a query consisting of base classes, relationships that the client was not initial aware of may be used by the query engine to produce results that could not have been anticipated based on the state of the system when the client is created. This insulation from future change is another benefit provided by basing the system on RDF and RDF-Schema.

Harmony ABC

Flexibility

Harmony ABC provides a good base model for representing changes over time. Through the appropriate use of RDF-Schema, the model can be extending in either dimension — time or actuality. The flexibility provided by the model does make it difficult to choose a strategy to extend the model for a specific application. The History System did not require any extensions to the base Harmony ABC model in order to support the basic requirements of tracking the creation, modification, and deletion of DSpace items. Harmony ABC chooses to model these event types as properties of the Event class. Although this mechanism is consistent, it can cause complexity in the schema by defining a large number of specialized properties for a given class.

Query Complexity

Because of the separation of events, which model points in time, and situations, which model the state of one or more actualities between points in time, the complexity of queries is higher than in a model that simply represents the state at a point in time. One benefit of this added complexity is the opportunity to provide an event correlation model that does not rely on grouping all state at one point in time or under a single situation. An event may be introduced into in the model representing a known event that correlates other sub-events that occurred during the correlation event.

DSpace Object Model

Extensibility

Extending the DSpace Object Model can be done through subclassing or through the use of subproperties. This distinction is vague in RDF due to the rich semantics that can be applied in either case. Properties should be general enough that they can apply to multiple classes. It is perfectly reasonable in an inferencing environment to define a subclass that also refines the usage of a property defined in the base class. In this case, the inferred productions from the schema for the more refined class will include the more generalized productions that would exist in the base class. For this reason, care must be taken to match the level of abstraction between the properties and the classes in which they are defined. Using a more refined property in a more abstract class will result in semantic limitations which impede extensibility.

Usage of Containers

Models in RDF subscribe generally to one of two models, known as the hedgehog model and the container model. For repeating properties (those with more than one value or with alternative values), the hedgehog model repeats the property arc for each value or alternative. The container model encourages the use of a single property arc that refers to an RDF container construct that contains the values or alternatives. The advantage to the hedgehog model is that separate models can be easily combined or compared by combining or comparing simple statement constructs. Using the container model requires the set to be navigated, which is a more complex operation. For the DSpace Object Model, the hedgehog model is used both for the additional simplicity in creating the model as well as the simplicity of merging operations. Whether multiple instances of a property represent alternatives or mutliple values can be annotated via the RDF-Schema that defined the DSpace Object Model.

Suitability of RDF

Although RDF was the only model explored during this project, it proved to be a useful way to produce models of the changes in the history system. One advantage it has over other schemes, such as XML, is the ease with which different subsystems can operate on the same model. A single model can be generated from the output of several subsystems, or these subsystems can all add statements to a common model. In either case, the resulting architecture is very loosely coupled, requiring only the cooperation of the subsystem, who writes the statements, and the end client, who interprets the statements.

These advantages are negatively offset by the lack of type information in RDF. However, appropriate type information can be added via the RDF-Schema mechanism, informing the client of type relationships that may be required during inferencing operations. This mechanism is bound to improve as the XML community and the RDF community continue to work together in order to incorporate type-correctness into the Semantic Web.

Query Complexity

The intention of the DSpace Object Model is to limit query complexity to areas such as Harmony in which this complexity provides distinct advantages. The complexity of the model can grow, however, depending on how the model is extended for specific purposes. For example, extending a class while also refining a property will cause a large number of inferred statements during queries. Navigating these inferred queries to find the item that matches the level of abstraction that the application requires adds complexity to the query facility.

Usage of Handles

During the design phase of the project, much of the debate centered on the use of CNRI Handles for identifying and locating DSpace objects. On one hand, Handles provide a URI-based abstraction from the physical location of an object and provide infrastructure for publishing an objects metadata. On the other hand, use of Handles increases the overall complexity of the system by adding an additional resolution protocol and database for dereferencing Handles. Concerns over the service level implied by handles also dominated the discussion at times.

Handles are used in the History System in order to allow the possibility of resolution while allowing a persistent identifier to be used for historical data. In practice, multiple identification schemes will most likely apply to an item over time, and it may be appropriate to separately model these transitions, perhaps using a model like Harmony ABC. This technique would allow these identifiers to be correlated, but some identification scheme that applies to all objects of a certain type is certainly useful to clients of the History System data. It remains to be seen if the Handle System adds value to DSpace or to the History System.

Areas for Future Work

RDF Model Extensions

The RDF model and schema must be implemented in such a way that extension schemata do not change the scope or level of abstraction of existing properties. If such a change in scope is required, a new property should be defined. If this property is semantically related to the original property, then the subPropertyOf relationship should be used to indicate this and to expose the property through the base class definition via inferencing. There will likely be cases in which complex data must be stored as property values during serialization.

The three options presently available include producing a child RDF instance with a corresponding schema, producing a pure XML representation that will be parsed by the RDF engine, or producing an escaped representation (including XML) that must be parsed by the client. If pure XML is possible, the second option is recommended. Future versions of RDF may incorporate the XML-Schema type facility, making generation and consumption of embedded XML documents much more straightforward.

Harmony ABC

Event Correlation

The current Harmony ABC event representation reduces the ability to correlate events that happen at or near the same time that may be related. It requires explicit event correlation to occur using the subEventOf relationship and an aggregate Event. Even this technique does not represent any relationship between the sub-Events and indicates only a containment relationship. If ordered or cause-and-effect correlations are required, the model would need to be extended to support these.

In the future, it would be appropriate to analyze what events may need to be correlated in the content management layer. The Harmony ABC model provides for long-duration events through the inclusion of a sub-event type. Sub-events occur in the context of a larger event. All events in Harmony ABC can span periods of time. This mechanism could be used to track changes of objects in a larger context, such as a data migration project.

Query

Existential Assertions

Although mathematically speaking,[2] it is legitimate to argue that a query mechanism for RDF need not handle the case of negative existential assertions, many applications do need to make these assertions. Even relational databases support the proper syntax to select elements that do not have a particular relationship to other elements. There is simply no analogous query in present RDF query facilities.

There are several useful cases in which such an assertion could be valuable. In the history system, it may be desirable to formulate a query that would request the end of a chain of Events and Situations. In order to form this query, though, a request must be made for an Event that has no preceding Situation.

Use of Inferencing

The query mechanism must be careful not to inundate the client with unnecessary inferred statements and to answer the query precisely.

Bombarding the client with extra statements inherently limits the scalability of the query mechanism without providing any value to the client. Specifically, inferred statements should be used by the query mechanism, but the statements that actual exist in the model should be included in the results of the query.

One possible application of negative existential assertions involves tuning the query engine to trim the set of result statements after applying inferencing rules. This technique would make it possible to request the most general or most specific inferred statement be the result of a given query by finding either end of the inheritance chain.

Conclusion

The DSpace History System is a simple subsystem that provides data serialization and event maintenance to DSpace as well as historical query capbilities to clients of DSpace. Even though the job performed by the subsystem is simple, consisting primarily of serializing instances of DSpace object and connecting them via Harmony ABC, there are many integration points that needed to be considered when implementing the system. A prior integration with the DSpace Content Management system was enhanced to provide a means to incorporate multiple objects into a single DSpace event. New and public RDF Schemata were integrated to improve interoperability with clients that recognize these schemata. And, an RDF-specific query mechanism was incorporated to provide remote clients with an easy way to navigate the data collected by the History System. The result is a quiet distiller of information that, hopefully, will grow valuable over time and, because of heavy use of standards, will continue to be usable well into the future.

References

1. Tansley, et. al. 2004. The DSpace institutional digital repository system: current functionality.in Proceedings of the third ACM/IEEE-CS joint conference on Digital libraries. 87-97.
2. Tansley, Robert, Mick Bass, MacKenzie Smith: DSpace as an Open Archival Information System: Current Status and Future Directions. ECDL 2003. 446-460.
3. Bass, Michael J., et. al. The DSpace™ Institutional Repository System: Status, Roadmap, Research, Governance, and Community Building.
4. SIMILE Project. <http://web.mit.edu/simile/www/>.
5. Trusted Digital Repositories: Attributes and Responsibilities. An RLG-OCLC Report. May 2002. <http://www.rlg.org/longterm/repositories.pdf>.
6. Resource Description Framework, <http://www.w3.org/RDF/>.
7. Jena Open-Source Semantic Web Toolkit. <http://jena.sourceforge.net>.
8. HP Labs Semantic Web Research Activity. <http://www.hpl.hp.com/semweb/>.
9. RDF Schema. <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>
10. The Harmony Project. <http://metadata.net/harmony>.
11. Lagoze, Carl and Jane Hunter, The ABC Ontology and Model. <http://metadata.net/harmony/JODI_Final.pdf>
12. Dublin Core Metadata Element Set. <http://dublincore.org/documents/dces/>.

Biographies

Jason Kinner investigates the integration of digital media systems at Hewlett-Packard Laboratories. He was the architect and implementer of the current DSpace History System. Over the past several years, he has participated in various efforts involving content management, archiving, and distribution. Jason holds a B.S.E. in Computer Science and Engineering from the University of Pennsylvania.

Mick Bass manages research in digital media systems at Hewlett-Packard Laboratories. Mick was the HP project manager for the HP/MIT DSpace development effort, and currently leads the SIMILE research project. Prior to DSpace, he led a team within HP's Enterprise Computing organization to create and deploy management methods and supportive software tools to effectively manage large and complex development efforts. His previous background was in hardware and software design contributing to HP's Precision Architecture microprocessors. Mick has fourteen years experience with HP, holds an MS in Management of Technology from MIT Sloan, and a BS in Computer Engineering from the University of Illinois.