III Conferecia Bibliotecas y Repositorios Digitales de América Latina (BIREDIAL '13)
VIII Simposio Internacional de Bibliotecas Digitales (SIBD '13)
"ACCESO ABIERTO, PRESERVACIÓN DIGITAL Y DATOS CIENTÍFICOS"
Ciudad de la Investigación, Universidad de Costa Rica, del 15 al 17 de octubre de 2013.
Ponencia

Control de integridad y calidad en repositorios DSpace

Marisa R. De Giusti
Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CICPBA)
Proyecto de Enlace de Bibliotecas (PREBI)
Servicio de Difusión de la Creación Intelectual (SEDICI)
Universidad Nacional de La Plata, Argentina
marisa.degiusti@sedici.unlp.edu.ar

Nestor F. Oviedo
Proyecto de Enlace de Bibliotecas (PREBI)
Servicio de Difusión de la Creación Intelectual (SEDICI)
Universidad Nacional de La Plata, Argentina
nestor@sedici.unlp.edu.ar

Ariel J. Lira
Proyecto de Enlace de Bibliotecas (PREBI)
Servicio de Difusión de la Creación Intelectual (SEDICI)
Universidad Nacional de La Plata, Argentina
alira@sedici.unlp.edu.ar

Gonzalo Luján Villarreal
Proyecto de Enlace de Bibliotecas (PREBI)
Servicio de Difusión de la Creación Intelectual (SEDICI)
Universidad Nacional de La Plata, Argentina
gonzalo@prebi.unlp.edu.ar

Resumen

En los últimos años se ha visto un incremento significativo en la creación de nuevos repositorios digitales, así como en la consolidación de los ya existentes. Esto puede atribuirse en gran medida al creciente interés por parte de las instituciones académicas y científicas en reducir las restricciones de acceso a su producción, tomando la decisión de compartirla públicamente bajo las políticas de acceso abierto, y buscando asegurar este libre acceso en el tiempo a través de diversas estrategias de preservación. Es así que el gran crecimiento de los repositorios digitales como puntos centralizados para el depósito, difusión y preservación del

material académico y científico de las instituciones se ha convertido en una estrategia a nivel institucional.
El crecimiento de los repositorios digitales se ve reflejado mayormente en la cantidad de recursos que los mismos son responsables de almacenar, compartir y preservar; en la medida que el volumen de recursos aumenta, también aumenta la necesidad de contar con mecanismos de control automático de metadatos y archivos, a fin de simplificar el trabajo de control de los administradores sobre la calidad de los metadatos descriptivos, administrativos y de preservación, buscando de esta manera contar con un repositorio eficiente y confiable.
El software para repositorios digitales DSpace, de amplio uso a nivel mundial, ofrece para estos fines un sistema de control semi-automático denominado *Curation Tasks* que permite evaluar y/o modificar cualquier característica deseada sobre todos los recursos del repositorio, o bien sobre un conjunto acotado a una comunidad o una colección específica.

Este trabajo describe 2 vías de extensión sobre el módulo de *curation* de Dspace. En primer lugar se describe un conjunto de *curation tasks* orientadas a analizar y reportar distintos aspectos asociados a la calidad de los datos y a brindar un soporte adicional a las tareas de preservación sobre el repositorio por medio de chequeos de integridad y de generación de nuevos metadatos. En segundo lugar se plantea la modificación de la estrategia de ejecución de *curation tasks* provisto por DSpace, en pos de minimizar su impacto en la performance de la aplicación, y flexibilizar los criterios de selección de recursos a procesar.

A continuación se mencionan *curation tasks* que serán consideradas en este trabajo:

- chequeo de enlaces web a documentos alojados en servidores externos al repositorio, configurable;
- chequeo de metadatos conectados con autoridades (o vocabularios controlados) dentro o fuera del repositorio, para chequeo de integridad en los datos;
- chequeo de archivos cargados en el repositorio, asegurando que todos los recursos cuenten con un archivo asociado bajo las normas y políticas del repositorio;
- control de metadatos obligatorios, según el tipo de documento;
- control del dominio de metadatos, de acuerdo a tipos primitivos como ser fecha, número, texto, etc;
- generación de metadatos de preservación a partir de los archivos asociados a los recursos (ej.: software con el que se realizó el archivo, con su correspondiente versión, versión del formato, nivel de compresión utilizado, etc)
- testeo y posterior reporte de recursos a partir de condiciones lógicas sobre metadatos y archivos, utilizando un lenguaje de expresión simple.

Actualmente las *curation tasks* se ejecutan sobre todos los ítems de una colección, una comunidad o incluso el repositorio completo, sin interrupciones y de manera secuencial. Esta estrategia de selección y ejecución genera una elevada demanda de recursos sobre el servidor que aloja el repositorio durante todo el tiempo de ejecución de los procesos de *curation tasks*, degradando la performance del mismo. Además, cuando se incluye más de una tarea en una misma orden de ejecución, éstas se ejecutan de forma secuencial, es decir, una tarea no puede iniciar su ejecución hasta tanto la tarea anterior no haya finalizado completamente. De aquí que

en este trabajo se propone una nueva estrategia para la selección de los recursos a procesar y dos nuevas estrategias de ejecución de *curation tasks*:

- estrategia de selección de recursos a procesar en base a una expresión lógica configurable (ej.: seleccionar recursos según el valor de su metadato *dc.type*);
- estrategia de ejecución por lotes incrementales en pos de disminuir el impacto de la ejecución de las *curation tasks* sobre el sistema;
- cambio en la forma de ejecución secuencial a fin de obtener un avance uniforme a nivel de recursos en el procesamiento, en lugar de un avance a nivel de *curation tasks*.

Hacia el final de este trabajo se mencionan otras posibles *curation tasks*, haciendo especial hincapié en las dificultades de implementación y la utilidad de las mismas en lo que respecta a la calidad de los metadatos y archivos, y a la preservación de los recursos. Entre estas tareas se pueden destacar un mecanismo de diagnóstico de archivos plausibles de ser o no preservados, un proceso de detección de recursos duplicados, una tarea de inferencia de relaciones entre recursos, de extracción de bibliografía a partir del texto completo, entre otras.

Palabras clave: preservación, calidad, integridad, curation system, curation task, dspace

1. Introducción

El significativo incremento visto en los últimos años respecto de la cantidad de repositorios digitales nuevos, así como la consolidación de muchos de los existentes, es una muestra del interés que las instituciones han puesto en la difusión y preservación en el tiempo de su producción académica y científica. En línea con esta tendencia de crecimiento y consolidación, sumado a los múltiples desarrollos orientados a promover la interoperabilidad entre repositorios, éstos últimos se han visto en la necesidad de contar con mecanismos automatizados para el control de sus datos y para la detección temprana de potenciales problemas de integridad, en pos de mejorar la calidad de la información y servir como soporte adicional para las tareas de preservación digital.
En este trabajo se describen un conjunto de tareas automáticas e independientes, en el marco de un repositorio digital implementado bajo el software DSpace, haciendo uso de su *Curation System* para el control de los datos existentes, generación de reportes y generación de nuevos datos orientados a la preservación digital.

2. Curation System en DSpace

El software para repositorios digitales DSpace provee un módulo denominado *Curation System*, cuyo objetivo es servir como un mecanismo automático para la ejecución de controles y validaciones periódicas sobre el contenido del repositorio [2]. Esta funcionalidad está disponible desde la versión 1.7 de este software y se destaca su capacidad para ser extendido y configurado según las necesidades de cada repositorio. En particular, permite que cada

institución desarrolle sus propios controles y validaciones, permitiendo un simple acoplamiento de los mismos.
El *curation system* de DSpace es responsable de la selección, organización y ejecución de múltiples *curation tasks*: módulos de menor tamaño, desarrollados para realizar tareas acotadas y concretas sobre un conjunto de objetos del repositorio, o sobre todo el repositorio.
Entre las *curation tasks* disponibles en una instalación básica de DSpace se encuentran:

- *Format profiling*: recorre los ítems del repositorio y elabora una tabla que contiene todos los formatos de bitstreams encontrados, junto con la frecuencia de ocurrencia de cada uno.
- *Required metadata*: analiza cada ítem a fin de determinar si cuenta con todos los metadatos configurados como obligatorios, según la configuración del formulario de envíos.
- *ClamScan*: ejecuta un análisis en busca de virus sobre los bitstreams de cada ítem utilizando el software antivirus ClamAV[1]
- *Microsoft translator*: utiliza la *API Microsft Translator* para realizar traducciones automáticas de los metadatos.
- *Link checker*: analiza los metadatos en busca de URLs y realiza conexiones a las mismas a fin de determinar el código de respuesta.

El *curation system* de DSpace ofrece múltiples formas de ejecutar las *curation tasks*: desde la interfaz de usuario o desde la línea de comandos; de forma individual o indicando un conjunto de tareas (que se ejecutarán en forma secuencial); asociadas a algún paso en el workflow de revisión, etc. Además permite definir colas de *curation tasks* a fin de invocarlas de forma diferida usando normalmente el *cron* del sistema.

3. Curation Tasks para el control de integridad y calidad

A continuación se describen un conjunto de *curation tasks* pasibles de ser implementadas como extensiones en el software DSpace a modo de herramientas adicionales en lo que respecta al control de los datos y su integridad, y por ende funcionarán como elementos adicionales en pos de mejorar la calidad de la información.

3.1. *ConfigurableLinkChecker*: Metadatos con enlaces externos

DSpace ofrece en su instalación por defecto dos *curation tasks* para el control de los enlaces web contenidos dentro de los metadatos. Estas son:

BasicLinkChecker: selecciona sólo los metadatos cuyo calificador es *"uri"* (sin importar el esquema ni el nombre del metadato) y considera que el contenido de estos metadatos corresponden a URLs que deben ser verificadas.

[1] http://www.clamav.net

MetadataValueLinkChecker: selecciona todos los metadatos asociados a un ítem y en cada caso verifica si su contenido comienza con las cadenas “http://” o “https://”. En caso afirmativo, considera que dicho metadato contiene una URL para verificar.

En ambos casos, la verificación se basa en realizar una conexión HTTP (o HTTPS) utilizando la URL a chequear, y controlar que el código de estado retornado por el servidor se encuentre en la familia 2XX.

Si bien éstas *curation tasks* ofrecen lo necesario para chequear que los enlaces externos sigan siendo válidos, presentan una importante falencia: no es posible configurar los metadatos sobre los cuales estas tareas realizarán sus funciones.
En el caso de MetadataValueLinkChequer, el hecho de recorrer todos los metadatos de cada ítem sólo para determinar cuáles de éstos contienen un enlace pasible de ser chequeado es un trabajo muy costoso. A menos que se trate de un esquema de metadatos en el que se requiere la carga de múltiples metadatos distintos con enlaces como contenido, el trabajo será en su mayor parte un gran desgaste de recursos. Asimismo, en el caso de BasicLinkChecker, la restricción respecto de que el metadato a verificar debe obligatoriamente denominarse de forma tal que su calificador sea “uri” representa una limitación muy importante en la personalización del esquema de metadatos a utilizar, restringiendo así una de las principales características del software.

Esto evidencia la necesidad de contar con una *curation task* configurable en al menos dos aspectos: el conjunto de metadatos que serán considerados para la evaluación, y el tipo de recurso esperado como resultado en cada caso.
La configuración de los metadatos a revisar puede realizarse simplemente agregando una *property* en un archivo de configuración específico para esta *curation task*, ya que DSpace ofrece un módulo especial para la recuperación de estos parámetros de forma simple. Además, las restricciones respecto de los tipos de respuesta esperados según el metadato revisado también puede ser configurado de esta forma.

3.2. *AuthorityChecker*: conexión con autoridades

DSpace ofrece un mecanismo de control de autoridades que por medio de directivas de configuración permite especificar que el contenido de ciertos metadatos deben ser obtenidos desde vocabularios controlados o bien desde servicios ajenos al software o al repositorio. De esta forma se permite que metadatos tales como autores, áreas temáticas, instituciones, etc., puedan ser recuperados desde bases de datos externas, creando una asociación explícita y unívoca entre las instancias de estos metadatos y los valores del vocabulario controlado o base de datos. Aún cuando existe la posibilidad de utilizar variantes en el texto del metadato, la asociación explícita se mantiene, ya que la misma se basa en un código de identificación único para cada término.

Las bases de datos externas suelen ser gestionados por herramientas de software completamente ajenas al software del repositorio, ya que incluir en DSpace las funciones necesarias para la administración de los tesauros, sistemas de clasificación, autores, instituciones, etc., puede resultar en desarrollos extensos y costosos. Debido a esto, en principio, la única forma de conexión que existe entre estos servicios ajenos al repositorio y el repositorio mismo son los *plugins* para el control de autoridades.
DSpace incluye en su instalación por defecto un plugin que es capaz de leer y proveer términos desde un vocabulario controlado, alojado en forma local en el servidor bajo un formato XML, y con un esquema determinado. Este módulo es suficiente cuando el vocabulario puede obtenerse en forma de archivo (y transformarse en caso de ser necesario) en el formato adecuado; y también cuando se trata de un vocabulario que requiere muy pocas (o nulas) actualizaciones. Sin embargo, si se trata de nombres de autores o instituciones utilizados en el proceso de catalogación dentro del repositorio este *plugin* no resulta suficiente, ya que el esquema requerido por el módulo de autoridades podría no ser adecuado para la información que se requiere almacenar, y además es probable que se requieran actualizaciones con elevada frecuencia (cada vez que se agrega, elimina o modifica un elemento).
Es así que los *plugins* para el control de autoridades se limitan a consultar y recuperar información desde los servicios o bases de datos externas, y cada uno de estos servicios cuenta con su propio mecanismo o herramienta de gestión (incluyen la gestión manual de archivos en XML), dando lugar a problemas en la integridad de los datos: instancias de metadatos cuyos códigos corresponden a términos que fueron eliminados en el vocabulario controlado (o modificados) de forma unilateral por la herramienta de gestión correspondiente.

Con el fin de detectar los problemas de integridad introducidos por las modificaciones realizadas desde las herramientas de gestión, se propone la creación de una *curation task* que se ejecute periódicamente, verifique y reporte los casos en los cuales la clave almacenada en un metadato controlado por un *plugin* de autoridades ha sido eliminada del servicio externo. Además, sería deseable comprobar si existen diferencias entre el texto del metadato y el valor retornado por el servicio externo, incluyendo esto en el reporte final.

3.3. *fileChecker*: Validación de archivos

Cada repositorio establece sus políticas de contenidos y por ende, las condiciones que un ítem debe reunir para ser considerado como parte del acervo. Entre estas condiciones generalmente suele haber restricciones respecto de la obligatoriedad en la existencia de los archivos a fulltext, junto con los formatos aceptados y las características de los mismos.
En un ambiente en el cual el origen de los datos y archivos es variado (catalogación manual, ingesta mediante OAI-PMH, depósito remoto vía SWORD, etc), es necesario controlar que las políticas del repositorio respecto de los archivos almacenados se vean respetadas.
Con este fin se propone la creación de una *curation task* que analice cada ítem y verifique la existencia de al menos un bitstream, o en su defecto, que exista al menos un metadato cuyo contenido sea una URL que enlace al archivo alojado en un servidor externo. Además sería deseable ser capaz de especificar el formato de archivo esperado según el tipo de documento (ej.: valor del metadato *dc.type*).

Como resultado de la ejecución de esta tarea se generará un reporte en el cual se incluirán aquellos ítems que no cumplan con las restricciones definidas: que contenga un archivo en un formato aceptado, o bien que contenga un enlace al archivo.

3.4. *mandatoryMetadata*: Metadatos obligatorios

En una instalación por defecto DSpace cuenta con una *curation task* que permite controlar todos los ítems, a fin de reportar el conjunto de metadatos obligatorios ausentes. Para esto analiza la configuración de los formularios de envío, determina cuales son los metadatos requeridos, y verifica la existencia de dichos metadatos en cada ítem procesado. Como resultado se genera un reporte en el que se listan todos los ítems que no cumplen con la validación.
Desde la versión 3.0 de DSpace, la configuración del formulario de envíos fue extendida para soportar la carga de datos en base al valor de un metadato, en particular el metadato *dc.type*. Sin embargo esta nueva función no ha sido aplicada sobre la *curation task* de control y por lo tanto existen ítems que se consideran como inválidos erróneamente.
Se propone extender esta *curation task* a fin de incluir la nueva forma de carga basada en el valor del metadato *dc.type* como un elemento adicional a considerar al momento de determinar si un metadato es requerido. Esto es, antes de validar si el metadato es obligatorio, es necesario verificar si el mismo es adecuado (según la configuración) para el tipo de documento que se está controlando (según el valor del metadato *dc.type*).

3.5. *domainChecker*: validación de datos según un dominio específico

En las versiones actuales de DSpace no existe ningún mecanismo de definición y validación de restricciones respecto del dominio de datos al que cada metadato pertenece. Esto, sumado a que el orígen de los datos puede ser muy diverso (catalogación manual mediante formulario de carga, importación mediante OAI-PMH, depósito remoto vía SWORD, etc), implica que existe un alto riesgo de que se generen datos erróneos, inconsistentes, o inadecuados para un determinado metadato.
Por esta razón se propone la creación de una *curation task* responsable de verificar que se cumplan las restricciones de dominio sobre cada metadato, según los parámetros de configuración definidos para los mismos.
Es importante destacar que actualmente DSpace no contempla esta clase de restricciones sobre los elementos del formulario, por lo cual es necesario extender la lógica de configuración a fin de permitir registrar las mismas de forma simple y clara. En principio, se podría considerar incluir estas extensiones como parte de la configuración de los *schemas* de metadatos que serán utilizados por la aplicación (*registries* de DSpace), y así dejar establecido cuál será el dominio de cada metadato para toda la aplicación. Sin embargo, esta aproximación pierde robustez cuando se considera además que la definición de los formularios de carga permite definir distintas características para cada metadato (obligatoriedad, controlado por *value-pair*, visibilidad según la etapa de carga, etc), según distintos contextos de carga: carga basada en el metadato *dc.type*. Esto sugiere que un mismo metadato podría ser requerido en un contexto y opcional en otro. Asimismo, un metadato podría estar restringido a un dominio particular en

un contexto dado. Por esta razón es más adecuado extender la configuración de los formularios de envío a fin de soportar esta nueva forma de restricción (ver ejemplo en figura 1).

A continuación se listan distintas restricciones de dominio que podrían considerarse:

- Fechas: el valor ingresado debe corresponder con una fecha válida, según algún estándar.
- Autoridades: el valor debe contener una clave de autoridad y un nivel de confianza adecuado para ser considerado como seleccionado desde un plugin de autoridades.
- Texto de una línea: el valor no debe presentar caracteres de salto de línea
- Valor de value-pair: el valor debe corresponder con alguna de las opciones del *value-pair* definido en la configuración del formulario.
- Código de idioma: el valor debe corresponder a un código de idioma según algún estándar
- Boolean: el valor debe ser “TRUE” o “FALSE” obligatoriamente
- Link: el valor debe ser un enlace válido
- Expresión regular: dada una expresión regular, el valor del metadato debe coincidir con la misma

```
<field>
        <dc-schema>dc</dc-schema>
        <dc-element>identifier</dc-element>
        <dc-qualifier>uri</dc-qualifier>
        <label>Enlace externo</label>
        ...
        <domain-validators>link</domain-validators>
</field>
```

Figura 1. Ejemplo de configuración de formulario de envíos

A fin de contar con un mecanismo flexible y extensible para la configuración de los validadores encargados de controlar cada metadato, se propone además agregar una nueva propiedad de configuración al sistema, a través de la cual se definirán las clases que implementarán cada validación, junto con un nombre (corto, del tipo *slug*) a utilizar en la configuración del formulario de carga como modo de referenciarlas (ver ejemplo en figura 2).

```
plugin.named.org.dspace.content.validators.ValidatorManager = \
 ar.edu.unlp.sedici.dspace.validators.DateValidator = date, \
 ar.edu.unlp.sedici.dspace.validators.LinkValidator =link, \
 ar.edu.unlp.sedici.dspace.validators.SingleLineValidator = singleline
```

Figura 2. Ejemplo sobre definición de validadores

El resultado de la ejecución de esta *curation task* será un reporte de los ítems y metadatos que no superaron satisfactoriamente la ejecución de sus correspondientes validadores.

3.6. *metadataExtractor*: generación de metadatos de preservación

La preservación digital forma parte del conjunto de características pilares asociadas a los repositorios digitales, por consiguiente es muy importante proveer de funcionalidad al repositorio a fin de realizar esta tarea de la forma eficiente y confiable.
Para asegurar la disponibilidad de los recursos digitales en el tiempo, se requiere no sólo contar en el recurso protegido bajo políticas de backup y demás estrategias que aseguren que el mismo no se pierda, sino que es igualmente importante contar con la información necesaria a fin de ser capaces de accederlo y poder visualizar/utilizar su contenido [1].
Esta información normalmente se registra utilizando metadatos, denominados "de preservación" [1], y su principal función es registrar todos los datos asociados tanto al recurso digital en sí, como al contexto necesario para acceder dicho recurso en el tiempo (plataforma, versiones de software, dispositivos de almacenamiento, dependencias, etc).
El objetivo de la *curation task* que aquí se propone es el de automatizar la generación de parte de estos metadatos de preservación, con la intención de simplificar el trabajo del personal humano por un lado, al tiempo que se asegura que todos los recursos cuenten con información de preservación precisa y confiable. A estos efectos existen múltiples aplicaciones y librerías que permiten extraer información desde archivos. Un caso muy utilizado es Apache Tika[2], una librería de acceso abierto con la capacidad de leer y extraer información útil a los fines de la preservación. Por ejemplo, entre los datos que se podrían extraer de un archivo PDF se pueden mencionar:

- tamaño físico de las páginas (en milímetros o pulgadas)
- software, y versión del mismo, con el que se creó el archivo fuente
- software, y versión del mismo, con el que se generó el PDF a partir del archivo fuente
- cantidad de páginas
- versión del estándar PDF en el que el archivo está basado

De la misma forma, Apache Tika permite obtener datos de este tipo desde múltiples formatos de archivos, como ser imágenes (JPEG, PNG, etc), audio y video (MPEG, AVI, etc), MS Office (DOC, XLS, etc), entre muchos otros.

El problema que surge inmediatamente es la ausencia de una estructura de datos que permita asociar información adicional a cada Bitstream[3] en DSpace. En concreto, los Bitstreams no poseen metadatos asociados. Una posible solución a esto puede ser considerar agregar un nuevo tipo de Bundle de preservación que albergará un nuevo Bitstream por cada archivo procesado por la *curation task*, el cuál contendrá la información para preservación extraída.
Como alternativa podría considerarse extender el modelo de datos de DSpace para permitir el uso de metadatos asociados a los Bitstreams, aunque si bien esto puede implicar múltiples ventajas incluso desde varios aspectos del software, también requiere un detenido análisis sobre los posibles puntos de conflicto y las extensiones al software requeridas, cuestiones que escapan al alcance de este trabajo.

[2] http://tika.apache.org/
[3] Para mas información sobre el modelo de datos ver https://wiki.duraspace.org/display/DSDOC3x/Functional+Overview#FunctionalOverview-DataModel

3.7. *expressionFilter*: generación de reportes en base a expresiones

En las versiones actuales del software no existe ningún componente que permita generar reportes personalizados en base a características arbitrarias definidas por los administradores del sistema. Una funcionalidad de este tipo resultaría de especial interés dado que permitiría perfilar distintos aspectos referidos al contenido del repositorio y detectar problemas de forma temprana.
Para estos fines se propone la implementación de una *curation task* que permita a los administradores del sistema escribir expresiones lógicas simples (que evalúen a *verdadero* o *falso*) a fin de incluir o descartar un ítem para su inclusión en el reporte final.
En una primera aproximación, este lenguaje simple de expresiones lógicas podría contar con el soporte de un *ValueStack* simple y acotado desde donde se hará referencia a elementos como: un ítem y todos sus componentes (metadatos, bundles, bitstreams, etc); una comunidad o colección y toda su información (descripción, items, permisos, etc); acceso a los parámetros de configuración; entre otros. Además sería deseable hacer uso de un conjunto de funciones simples para permitir la iteración y agregación sobre colecciones de elementos. Tanto los elementos del *ValueSatck* como las funciones especiales deberán ser referenciadas mediante palabras clave unívocas.
Cuando esta *curation task* sea ejecutada, cada uno de los objetos de DSpace a procesar (un Item, una Colección o una Comunidad) debería ser almacenado en el ValueStack y la expresión lógica evaluada. Como resultado de esta ejecución, los objetos para los cuales la expresión evaluada retorne verdadero serán incluídos en el reporte final.

Las palabras clave para referenciar el Item, la Colección o la Comunidad en proceso podrían ser *item*, *col* y *com*, respectivamente; y cada una estará disponible sólo cuando el respectivo objeto de DSpace esté en proceso. Posibles palabras clave para referenciar la iteración sobre una colección de elementos son *count* y *exists*. La tabla 1 describe su significado junto algunos ejemplos de uso.

count	contabiliza y retorna la cantidad de elementos Ejemplos: - cantidad de bitstreams en el Bundle ORIGINAL de un ítem item.bundle('ORIGINAL').bitstream().count() - cantidad de metadatos dc.title de un ítem item.metadata('dc.title').count() - cantidad total de metadatos de un ítem item.metadata().count()
exists	evalúa que al menos uno de los elementos cumpla una condición *Ejemplos:*

	- que tenga al menos un metadato dc.title en español item.metadata('dc.title').exists(o.lang = 'es')

Tabla 1. Palabras clave de iteración y agregación

4. Estrategias de selección y ejecución de tareas de curation propuestas

Actualmente la única estrategia de selección de recursos para la ejecución de *curation tasks* que ofrece DSpace se basa en la selección de un elemento dentro de la estructura jerárquica de comunidades y colecciones. En múltiples ocasiones es necesario realizar revisiones periódicas y/o cambios masivos sobre un conjunto de ítems que cumplan con ciertas características. Por ejemplo, es viable pensar en una validación sobre los archivos asociados a los ítems de tipo Artículo dentro de una comunidad en particular, considerando que dicha comunidad puede contener múltiples tipos de documento. Otro ejemplo puede ser una verificación respecto de la presencia de un metadato opcional dentro de un conjunto selecto de ítems: un metadato con referencias geográficas dentro de ítems cuya materia sea la geología.
Tanto en el caso de los ejemplos mencionados, como en muchos otros, es posible que los ítems de interés se encuentren dispersos en múltiples colecciones y comunidades, las cuales además pueden contener ítems con diferentes características. A priori, las posibilidades son incontables dado que cada repositorio establece la estructura de comunidades y colecciones acorde a sus necesidades, así como sus propios esquemas de metadatos.
Dado el modo de funcionamiento actual, la ejecución de las *curation tasks*, cualquiera fuera su objetivo, deben obligatoriamente procesar todos los ítems dentro de la comunidad o colección especificada (o incluso sobre todo el repositorio), debiendo incluir las restricciones de inclusión/exclusión de ítems como parte de la lógica de procesamiento de la tarea. Esto por supuesto implica la pérdida de objetividad de la *curation task* y la consiguiente dificultad en la reutilización de la misma sobre otros conjuntos de ítems.

Aquí se propone considerar una nueva estrategia para la selección de los ítems que serán procesados por la *curation task*, tomando como criterio de selección una expresión lógica que permita realizar evaluaciones simples sobre las características de un ítem. Así, de forma análoga a lo expuesto anteriormente cuando se describió la *curation task expressionFilter*, se plantea el uso de una expresión como un parámetro adicional al momento de la invocación, cuya evaluación sobre cada ítem dentro del conjunto preseleccionado de ítems (en base a la estrategia de selección original) determinará si un ítem debe procesarse o no. La figura 3 presenta un ejemplo de invocación por línea de comandos de una curation task utilizando una expresión que establece que sólo serán procesados los ítems cuyo metadato *dc.type* contiene el valor "Artículo".

```
dspace curate -e admin@dspace.org -i all -t fileChecker -f "item.metadata('dc.type') = 'Article'"
```

Figura 3. Ejemplo de estrategia de selección en base a una expresión

Respecto de las estrategias de ejecución de *curation tasks*, DSpace solo ofrece la posibilidad de realizar ejecuciones secuenciales sobre todo el conjunto de ítems a considerar. Esto es, cuando en una invocación se indican más de una tarea a ejecutar, cada tarea comienza su trabajo sólo cuando su predecesora ha terminado el propio; y una tarea se da por terminada cuando ha completado su procesamiento sobre todos los ítems considerados (puede ser toda una comunidad, colección o el repositorio completo).
Cuando se trata de controles continuos y periódicos sobre todos los ítems del repositorio, es importante tener presente que cada ejecución puede representar una carga de procesamiento importante, así como un prolongado tiempo de ejecución. Por ejemplo, las *curation tasks* asociadas con el control de enlaces requieren ser ejecutadas periódicamente sobre todo el conjunto de ítems, ya que un enlace externo puede quedar fuera de servicio en cualquier momento. Esto significa que no es suficiente revisar un ítem sólo una vez, sino que el mismo debe ser revisado con cierta frecuencia.
Este tipo de tareas requieren una estrategia de ejecución distinta a la propuesta por DSpace, ya que sus tiempos de ejecución, cargas de procesamiento o acceso a base de datos, etc., suelen ser valores significativos.

Aquí se propone entonces una estrategia de ejecución de *curation tasks* por bloques. Esto es, dado todo el conjunto de ítems a procesar, el mismo se divide en porciones equivalentes en cantidad de ítems, cada una de las cuales será procesada de forma independiente y en invocaciones independientes.

Esta estrategia de ejecución permite aumentar la frecuencia de ejecución de las curation tasks, reduciendo la demanda de recursos durante el tiempo de actividad, y además ofrece el potencial para detectar estados indeseados y problemas de manera temprana: con la estrategia tradicional, si la ejecución de la *curation task* es semanal y la misma acaba de ejecutarse, un problema producido en este momento no será detectado hasta la próxima semana; en cambio, con esta nueva estrategia es posible reducir la frecuencia a una vez por día, utilizando un bloque de mucho menor tamaño, en cuyo caso si el problema recientemente generado corresponde a un ítem en este bloque, el mismo sería detectado con un día de retraso como máximo.

En línea con lo que respecta a las estrategias de ejecución, una consideración importante a tener en cuenta es cómo se organiza la ejecución de múltiples *curation tasks* invocadas en una misma orden. En DSpace, la organización es secuencial: cada *curation task* comienza su ejecución sólo después de que la tarea anterior finalizó completamente. Esto lleva a una situación en la que un mismo objeto, para cada *curation task* a ejecutar, debe recuperarse de la base de datos e instanciarse para su lectura y/o modificación, para finalmente persistir los cambios sobre la base de datos nuevamente. Esta forma de ejecución genera un *overhead* importante cuando se procesa un gran número de ítems y cuando la lista de *curation tasks* a ejecutar es también extensa.
Por otro lado, la ejecución secuencial de las *curation tasks* puede resultar poco natural ya que, al inicio de la ejecución de múltiples tareas invocadas desde una misma orden, es lógico asumir que cada ítem será procesado por todas las tareas indicadas antes de pasar a procesar el siguiente ítem. Sin embargo, ese ítem será procesado por sólo una *curation task*, y su recuperación y procesamiento por parte de la siguiente tarea no se realizará hasta tanto no llegue su turno en la ejecución.

En base a estas consideraciones, se propone una nueva estrategia de ejecución de múltiples *curation tasks*, a fin de mejorar la performance general del *curation system* y de transformar la secuencia de ejecución para que esté más acorde al comportamiento esperado. En base a esto, la nueva estrategia propone que, una vez instanciado el objeto a procesar se ejecuten en forma secuencial todas las curation tasks requeridas sobre el mismo, para luego realizar una única actualización de la base de datos que reúna todas las modificaciones pertinentes. Luego de esto, se procede a realizar las mismas actividades sobre el siguiente objeto a procesar.

Comentarios finales

En este trabajo se mencionaron algunas de las múltiples posibles extensiones sobre DSpace en pos de mejorar la calidad e integridad de la información, así como también la información disponible para los fines de preservación digital. En particular las *curation task* son la herramienta que DSpace ofrece pensando en estas importantes necesidades por parte de los repositorios digitales. La realización de chequeos y correcciones de forma periódica, así como la constante generación de reportes, permiten detectar potenciales problemas en forma temprana.

Además de las extensiones mencionadas previamente, y a modo de trabajos a futuro, es interesante considerar la implementación de una *curation task* que sea capaz validar la información de preservación asociada a los archivos del repositorio con respecto a lo registrado en, por ejemplo, el servicio remoto PRONOM[4].

Por otro lado, la existencia de múltiples orígenes para la información de un repositorio (catalogación, OAI-PMH, SWORD, etc.) generan la necesidad de contar con algún mecanismo de deduplicación de documentos. El desarrollo de una *curation task* para este fin requiere una gran discusión respecto de cuáles serán los aspectos a considerar para asumir que un recurso se encuentra duplicado, o bien de como calcular una medida de similitud entre documentos a fin de establecer un umbral de duplicidad, etc.

Asimismo, un repositorio digital ofrece uno de los contextos más propicios para la extracción de conocimiento ya que el uso de metadatos como forma de estructuración de los datos aporta un nivel semántico sobre la información almacenada. A partir de esto, se viable pensar en *curation tasks* que analicen esta información semántica a fin de inferir relaciones inicialmente implícitas entre los documentos, autores, temáticas, etc. Este tipo de tareas por supuesto requiere un gran trabajo de análisis, no sólo en lo que respecta a la información disponible y los pasos a seguir para llegar a una estrategia de inferencia eficiente, sino también respecto del desarrollo de modelos y ontologías para el almacenamiento de dichas relaciones, y su posterior recuperación.

Referencias

1. De Giusti, M.R., Lira, A.J., Villarreal, G.L., Texier, J.D., Oviedo, N.F.: Las actividades y el planeamiento de la preservación en un repositorio institucional. BIREDIAL - Conferencia

[4] http://www.nationalarchives.gov.uk/PRONOM

Internacional Acceso Abierto, Comunicación Científica y Preservación Digital. Barranquilla, Colombia. 2012. Disponible en http://sedici.unlp.edu.ar/handle/10915/26045
2. DSpace Manual. Disponible en https://wiki.duraspace.org/display/DSDOC3x
3. Knight, S.: Preservation Metadata: National Library of New Zealand Experience. Library Trends. Vol 54, No. 1. (91-110). 2005