EVALUACIÓN AUTOMÁTICA DE PRESERVACIÓN MEDIANTE LA UTILIZACIÓN DE DE TAREAS DE CURACIÓN EN DSPACE

Terruzzi Franco Agustín[1], Lira Ariel Jorge[2], De Giusti Marisa Raquel[3], Villarreal Gonzalo Lujan[4]

1. Estudiante avanzado de Licenciatura en Sistemas en la UNLP. Servicio de Difusión de la Creación Intelectual, Proyecto de Enlace de Bibliotecas, Universidad Nacional de La Plata (UNLP).
2. Licenciado en Informática. Servicio de Difusión de la Creación Intelectual, Proyecto de Enlace de Bibliotecas, Universidad Nacional de La Plata (UNLP).
3. Ingeniera en Telecomunicaciones y Profesora de Letras. Servicio de Difusión de la Creación Intelectual, Proyecto de Enlace de Bibliotecas, Universidad Nacional de La Plata (UNLP).Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CICPBA).
4. Doctor en Ciencias Informáticas. Servicio de Difusión de la Creación Intelectual, Proyecto de Enlace de Bibliotecas, Universidad Nacional de La Plata (UNLP).

Introducción

Muchos repositorios institucionales asignan parte de sus recursos a las actividades ligadas a la difusión y la preservación [1]. La difusión se realiza mediante una serie de acciones cuyo objetivo es lograr que los elementos del repositorio alcancen a la comunidad; la preservación se refiere a la persistencia en el tiempo de tales elementos, a fin de asegurar su acceso y usabilidad a largo plazo. Ambas actividades están influenciadas por los aspectos políticos, económicos y tecnológicos que rigen al repositorio. Las actividades de preservación están enmarcadas por las políticas del repositorio: el desarrollo de tales políticas deben ser el primer paso para garantizar las acciones de preservación [2]. Estas acciones implican generalmente la validación de los elementos del repositorio a partir de las restricciones indicadas en sus políticas, lo que puede realizarse tanto forma manual como con la ayuda de alguna herramienta que permite su automatización [3]. Típicamente, durante la evaluación de un recurso se toman algunas de sus propiedades y se las compara con parámetros esperados según se ha establecido previamente. Estos valores esperados surgen, en muchos casos, a partir de modelos estándares como la norma OAIS, lo que permite definir la generación de un plan de preservación basado en un contexto normalizado y acotado. No obstante, los cambios en las políticas y la evolución tecnológica constante hacen que estas validaciones deban revisarse permanentemente, dificultando la ejecución de las tareas de evaluación automáticas o semiautomáticas [4].

Por otro lado, las tareas de evaluación y control pueden efectuarse manualmente mediante comprobaciones realizadas desde el software en uso o incluso ejecutando consultas a la base de datos, y por medio del establecimiento de un flujo de trabajo para la gestión de los cambios sobre los elementos. Sin embargo, si bien esto permite alcanzar un alto nivel de precisión al evaluar los elementos del repositorio, esta labor demanda mucho tiempo y, como todo proceso manual e iterativo, suele ser propenso a fallos.

El SEDICI [5] es un caso típico de repositorio digital donde se presenta la situación descripta en los párrafos anteriores. Desde su creación en el año 2003, el repositorio institucional de la Universidad Nacional de La Plata [6] ha evolucionado tanto en las políticas como las tecnologías que utiliza [7]. En la actualidad reúne cerca de 35000 obras que provienen de toda la UNLP y con tipologías muy variadas. Este crecimiento sostenido y evolución contínua a lo largo de los años demanda grandes esfuerzos para realizar los controles y evaluaciones sobre los ítems, así como también actualizar los mecanismos para realizar dichos controles.
En este contexto, el presente trabajo propone una herramienta que permite evaluar e informar el "*estado actual de preservación*" de los elementos del repositorio de manera automática. La herramienta permite personalizar tanto los controles como las actualizaciones a realizar, y dispone de mecanismos para seleccionar los elementos del repositorio en donde aplicar dichos controles. Esto se enmarcó en el desarrollo de Dspace [8], para que se pueda compartir con la comunidad especializada en el tema.

Diseño

Se estableció un diseño que permite representar la variabilidad de las reglas de preservación según su alcance, y se implementó un método para construir nuevas reglas y ejecutarlas sobre elementos del repositorio especificados previamente. Cada regla a ejecutar forma parte de un contexto determinado, que define su nivel de abstracción así como también las reglas de validación aplicables a dicho contexto. En este sentido, se pensaron diferentes *niveles de especificación*[1] para la definición de las reglas: la norma OAIS como nivel más general, el nivel de Repositorio para definir elementos y restricciones típicas de los repositorios, el nivel Dspace donde se determinan aspectos propios de este software, y finalmente el nivel SEDICI que toma elementos definidos particularmente para este repositorio (ej. metadatos propios). Las características de cada nivel definen la reusabilidad de las reglas, su aplicabilidad (desde las más abstractas hacia las más específicas) y su relevancia. Para cada nivel se determinan 1) las restricciones a evaluar, 2) las propiedades a verificarse y 3) el contexto del cual se desprenden. Lógicamente, a medida que el nivel se vuelve más específico, también lo hacen sus restricciones a partir del repositorio donde se aplican las reglas y, al mismo tiempo, su implementación se vuelve más simple y factible. Esta variabilidad hace que las reglas de mayor abstracción sean aplicables a la mayoría de los repositorios digitales, pero también sean más costosas en tiempo y esfuerzo en el momento de su implementación.
En la Imagen 1 se muestran los niveles de especificación establecidos y los diferentes dominios y restricciones que presentan.

---

1 Utilizaremos este concepto en el trabajo, para referirnos al contexto de aplicación de la regla que se describe.

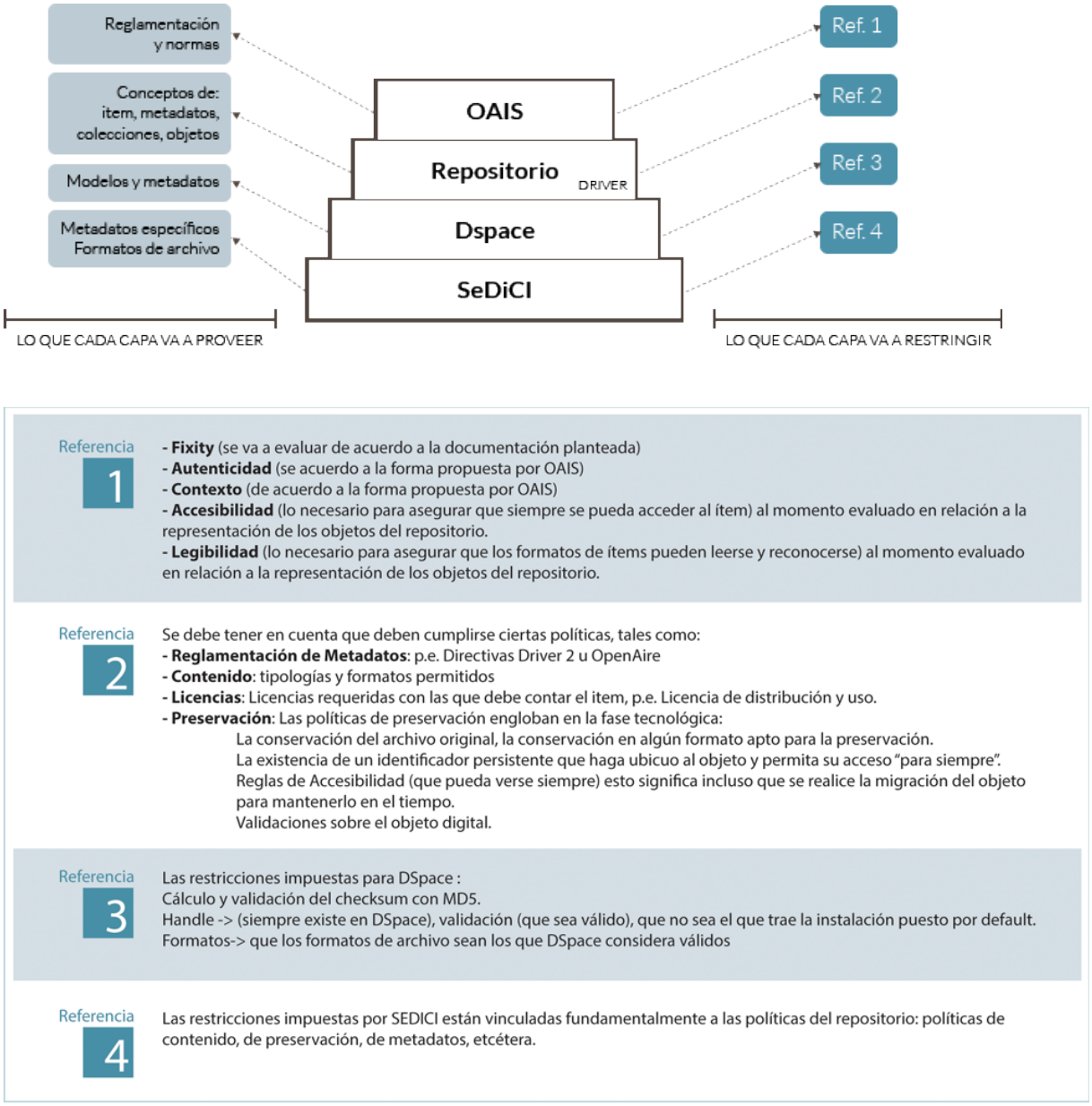

Figura 1: Niveles de especificación definidos: restricciones y propiedades de cada uno. Mientras más específico es el nivel, las propiedades y restricciones que ofrece se tornan más dependientes de la implementación y del repositorio en el que se esta trabajando.

Se trabajó en una primera iteración sobre la aplicación de reglas aisladas sobre conjuntos de ítems, y a posteriori sobre la posibilidad de añadir nuevas reglas, quitarlas y modificarlas. Por otro lado, se estableció un nuevo criterio para evaluar el *grado de preservación*[2] de un determinado recurso, lo que permite obtener una medida uniforme de comparación entre los elementos. Para esto la validación de cada regla retorna un valor numérico, que representa el grado con el que el elemento cumple con dicha regla. Dado que este nivel de adecuación a una regla

2 El grado de preservación es un concepto utilizado en este trabajo para establecer una medida cuantitativa de la posibilidad de preservar un determinado ítem o elemento en un repositorio.

puede variar de acuerdo con el contexto en que se la aplica, se incorporó un mecanismo que permite definir la importancia de cada regla junto con el orden y contexto de aplicación. De este modo, es posible preparar "*recetas*": conjuntos de reglas que poseen un peso y un orden de ejecución que depende de la receta, y que se calculan en el momento de ejecutar el validador. Este validador fue diseñado sobre un modelo de objetos que hace énfasis en el modelado de reglas y en la posibilidad de configurar el modo de aplicación de las mismas.

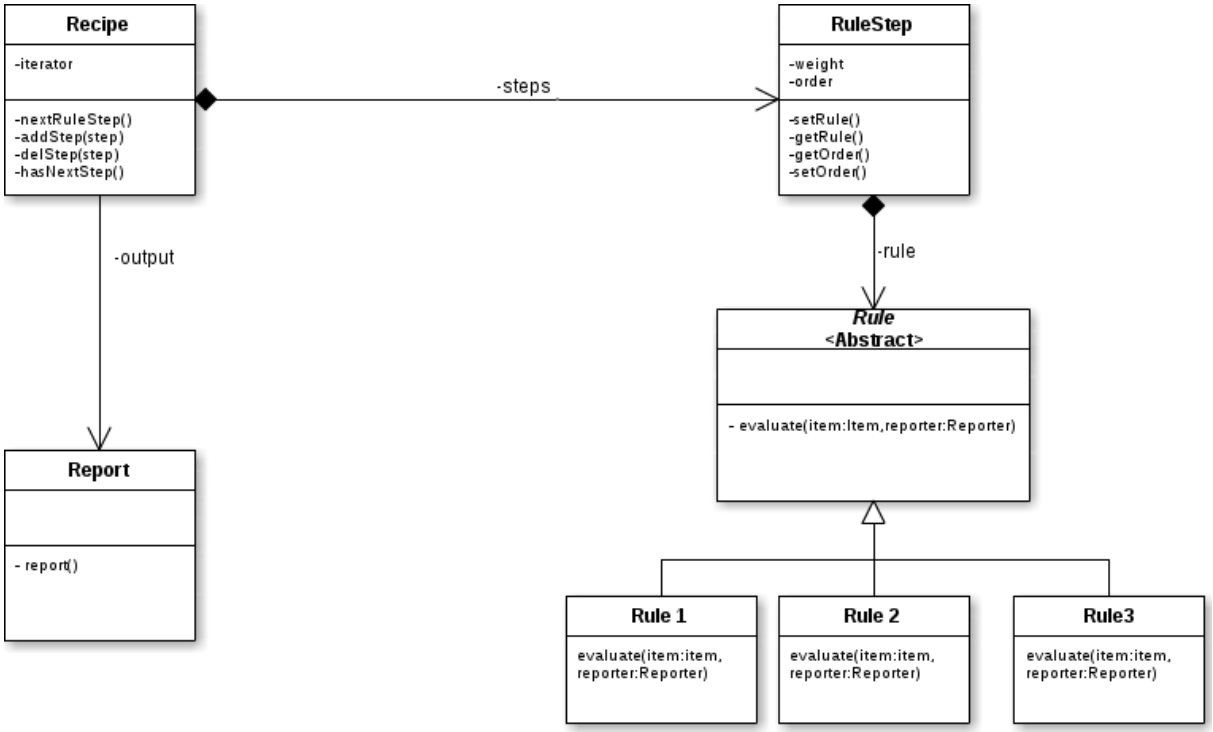

Figura 2: Diagrama UML del validador propuesto. Las reglas (Rule) conforman una jerarquía fácilmente extensible, y se organizan en etapas con un orden y un peso (RuleStep). Estas etapas componen recetas (Recipe), cuyo resultado es reflejado en un reporte (Report).

Se estableció así un mecanismo normalizado de definición de reglas, en donde la administración del repositorio puede definir una regla fácilmente, a partir de una definición semántica de la misma. En el SEDICI, se concilió[3] una estandarización para la especificación de las reglas de validación, que se escriben en base a una estructura como se muestra en el ejemplo a continuación:

Enfoque: DSpace
Nombre: "Verificar validez de Handle"
Restricciones: Handle siempre existe en DSpace, puede ser el handle predefinido
Regla: El ítem contiene un handle válido y no se trata del Handle por defecto (123456789)

3 Se estableció de común acuerdo entre el personal técnico, administrativo y los expertos la temática.

Metadato(s) Asociado: dc.identifier.uri (sedici.identifier.handle)
Respuesta esperada: True => Válido
False => Inválido

Cada regla se implementa mediante una clase JAVA [9] como subclase de la clase abstracta Rule. El uso del patrón de diseño *Template Method* [10] simplifica la generación de nuevas reglas, requiriendo la implementación de la regla de evaluación (método *evaluate*). El resultado de la ejecución de cada receta, con sus etapas y reglas, es enviado hacia un reporte. La clase Report permite encapsular la información que se desea generar en el reporte, alterar el grado de granularidad (nivel de especificación) de la misma, e incluso adaptar el formato de salida según los requerimientos del usuario (archivo de texto plano, csv, salida en pantalla, documento HTML, etc.)

Implementación

El diseño antes propuesto debía acoplarse al contexto de SEDICI, principalmente al software que sustenta este repositorio. Por este motivo se buscó adaptar el modelo hacia una estructura compatible con un módulo de Dspace. Esto permite reutilizar todo el modelo de datos y de objetos del repositorio y, de este modo, es posible implementar rápidamente una solución orientada a la reutilización.
Dspace ofrece la posibilidad de desarrollar módulos tradicionales, pero también permite gestionar tareas de curación [11] [12], lo que resulta de suma utilidad para esta solución. Se diseñó entonces una *tarea de curación* específica capaz de ejecutar la validación correspondiente y que conozca cómo y cuáles reglas aplicar. Esto llevó a que en una segunda iteración, se construya el diagrama UML [13], mostrado en la imagen 2.

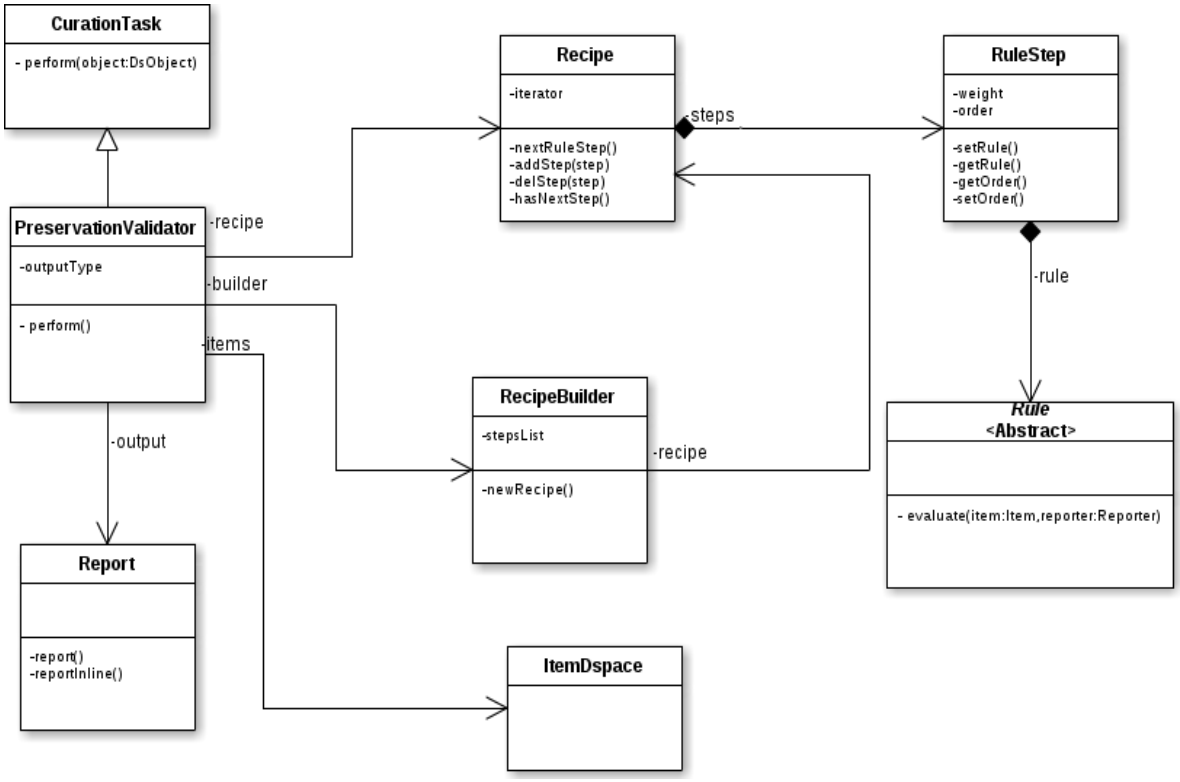

Figura 3: Diagrama UML del validador propuesto a partir de la adecuación de la solución a una tarea de curation en Dspace. El validador es en sí mismo una tarea de curación y se utiliza como tal. Con cada ejecución, aplica una serie de reglas (Rules) ordenadas en recetas de diferente tipo, a un conjunto específico de elementos del repositorio (ItemDspace).

A partir de este diseño, se implementaron las clases JAVA correspondientes para que la tarea de curación pudiera ejecutarse. Un requisito importante de este desarrollo es poder incorporar y modificar recetas, basadas en reglas preexistentes, sin la necesidad de modificar el código fuente de la aplicación (lo que llevaría a recompilar, *deployment*, etc). Se buscó entonces un método de definición de recetas, que detecta las configuraciones en tiempo de ejecución. Esto permite intercambiar las recetas, reglas y etapas de manera muy simple y sin necesidad de conocer la implementación del validador. Para lograr esto, el desarrollo se basó en las especificaciones de las tareas de curación en Dspace, que permiten definir y cargar una serie de *propiedades* para cada tarea, un método por el cual una tarea puede conocer un valor de una configuración en un archivo determinado en tiempo de ejecución.

Se definió una clase *builder* [14] que es ejecutado por la tarea de curación y que instancia las recetas con las propiedades definidas. Estas propiedades se cargan en tiempo de ejecución, y son leídas desde el archivo de configuración específico del validador, que se encuentra con el nombre de *evaluatePreservation.cfg,* con la estructura que se muestra en el ejemplo a continuación:

```
#----------------------------------------------------------------#
#-------------RULE VALIDATOR Curation Task Configurations------#
```

```
#-------------------------------------------------------------#
#-------------- Configuration properties used solely-----------#
#----------------by the rule validator task ------------------#
#-------------------------------------------------------------#
#------------------@author Franco Agustín Terruzzi--------------#
validator.rules=\
ar.edu.unlp.sedici.dspace.curation.preservationHierarchy.preservationRules.HandleValidationRule =
r1,\
ar.edu.unlp.sedici.dspace.curation.preservationHierarchy.preservationRules.Md5ValidationRule =
r2,\
ar.edu.unlp.sedici.dspace.curation.preservationHierarchy.preservationRules.LicenceValidationRule =
r3,\
ar.edu.unlp.sedici.dspace.curation.preservationHierarchy.preservationRules.ExistMetadataValidation
Rule = r4

#rules properties
#rules properties must be written starting with rulename.propertyName and must be used like
Strings
#the rules properties values depends directly of the recipes configurations, hence it must be
specified in the recipe configuration file
# example
        # r1.predefined.Handle=1234567/8
# see the recipe configuration file for more details

#recipes must be written in the next format: recipe.rules = rule1, weigth1; rule2, weigth2; rule3,
weigth3
#recipes must be written in a distinct cfg file see HandleValidation.cfg
recipe.name = Empty Recipe
recipe.rules= []
```

Ejemplo 1: Configuración típica del archivo evaluatePreservation.cfg. Las tareas de curación en Dspace permiten configurar ciertas propiedades específicas de una tarea, mediante la formulación de un archivo de texto plano de configuración que posea el mismo nombre que la tarea.

La información especificada en el archivo de configuración permite al validador definir qué reglas aplicar, en qué orden aplicarlas y establecer el peso para cada una. Para permitir la aplicación de distintas recetas, se utilizó un *sistema de herencia* [15] de tareas de curación, en donde se define en el archivo de configuración correspondiente no sólo la tarea de curación en sí sino también la receta que dicha tarea aplica. Estos archivos de configuración fueron denominados *archivos de configuración de recetas* y deben ser definidos por los diseñadores de las reglas y de las recetas al momento de ejecutarlas. Esto permite, adicionalmente, que la ejecución del validador con una nueva receta sólo requiera añadir una nueva entrada en el archivo de configuración correspondiente que haga referencia al *archivo de configuración* específico de la nueva receta (*curate.cfg*). Esto se visualiza en el siguiente ejemplo:

```
#-------------------------------------------------------------#
#--------------CURATION SYSTEM CONFIGURATIONS-------------------#
#-------------------------------------------------------------#
# Configuration properties used solely by the curation system   #
```

```
#---------------------------------------------------------------#

### Task Class implementations

### Task Class implementations

plugin.named.org.dspace.curate.CurationTask = \
   org.dspace.ctask.general.NoOpCurationTask = noop, \
   org.dspace.ctask.general.ProfileFormats = profileformats, \
   org.dspace.ctask.general.RequiredMetadata = requiredmetadata, \
   org.dspace.ctask.general.ClamScan = vscan, \
   org.dspace.ctask.general.MicrosoftTranslator = translate, \
   org.dspace.ctask.general.MetadataValueLinkChecker = checklinks, \
   ar.edu.unlp.sedici.dspace.curation.LinkControlledMetadata = linkMetadata, \
   ar.edu.unlp.sedici.dspace.curation.CopyMetadata = copyMetadata, \
   ar.edu.unlp.sedici.dspace.curation.FixControlledMetadata = fixControlledMetadata, \
   ar.edu.unlp.sedici.dspace.curation.EvaluatePreservation = evaluatePreservation, \
   ar.edu.unlp.sedici.dspace.curation.EvaluatePreservation =
evaluatePreservation.HandleValidation, \
   ar.edu.unlp.sedici.dspace.curation.EvaluatePreservation =
evaluatePreservation.Md5Validation, \
   ar.edu.unlp.sedici.dspace.curation.EvaluatePreservation =
evaluatePreservation.ExistMetadataValidation, \
   ar.edu.unlp.sedici.dspace.curation.EvaluatePreservation =
evaluatePreservation.LicenceValidation
# add new tasks here
```

Ejemplo 2: configuración definida en el archivo curate.cfg. Para añadir una tarea de curación nueva, debe añadirse una línea en este archivo de configuración de Dspace. Cada receta que se quiera aplicar y configurar en tiempo de ejecución, debe añadirse en este archivo como una nueva tarea de curación.

En una instalación normal de Dspace, los archivos de configuración se encuentran en <directorio_instalación>/config/modules/NombreCuration.cfg.

De la misma forma se pueden definir un conjunto de propiedades de cada regla en una receta en específico. Esto surgió de la idea de que una regla puede variar levemente desde un contexto de aplicación a otro (por ejemplo, yendo de un ámbito más general a uno más específico), sin la necesidad de que tenga que implementarse por completo. Por este motivo en el archivo de configuración que representa a la receta correspondiente, es posible agregar algunas propiedades que estarán disponibles para las reglas durante su ejecución. Para esto, se utiliza la siguiente notación:

```
#---------------------------------------------------------------#
#--------------RULE VALIDATOR Curation Task Configurations------#
#---------------------------------------------------------------#
#-------------- Configuration properties used solely------------#
#-----------------by the rule validator task -------------------#
#---------use this configuration to set the choosen recipe name-#
#------------------@author Franco Agustín Terruzzi--------------#
recipes.name = ExistMetadataValidation
```

```
recipes.rules = r4,1;

#metadata to validate in this recipe
r4.metadata.validate=\
dc.description.provenance,\
dc.identifier.uri,\
sedici.identifier.uri,\
mods.location
```

Ejemplo 3: notación de las propiedades de una regla de validación en el archivo evaluatePreservation.ExistMetadataValidation.cfg. Las propiedades para una regla inician con el nombre de la misma y siguen el formato mostrado NombreDeRegla.Propiedad = valor1, valor2,valor3 (para valores con saltos de línea se debe agregar \ ).

El desarrollo aquí descrito permite ejecutar una tarea de curación que valide una serie de ítems, lo que puede resolverse directamente desde un intérprete de comandos y con solo un comando:

./dspace curate -r - -t evaluatePreservation.*<RecetaAValidar>* -i *<handleItem | handleComunidad | handlColección | all>* -v

Donde:

- -r define el tipo de reporte para gestionar la salida
- - le indica que la salida será la estándar
- -t permite definir el tipo de tarea que se va a ejecutar
- -i permite definir los tipos de ítems en donde se aplicará la tarea
- -v hace más comunicativa la salida de la tarea

A partir de estas primeras reglas se pudieron definir otras que se desprendían de los *niveles de especificación* ya mencionados, y se aplicaron recetas compuestas por varias reglas, con configuraciones, órdenes y pesos distintos. En este contexto, se construyó una receta que, para cada ítem, verifica una serie de propiedades que representan su *grado de preservación* con respecto a lo establecido como *Información Descriptiva de Preservación (PDI)* [16]. Se establecieron reglas para verificar:

- **Procedencia:** la información de procedencia es información que Dspace completa de manera predeterminada para cada ítem al momento de instanciarlo. Se comprueba que esta información exista y sea válida. Para esto se consulta el metadato estándar de dublin-core *dc.description.provenance* [17].
- **Contexto:** se verifica el contexto externo del item, validando que la información de contexto exista y esté asociada correctamente al ítem. Para hacer esto, se consultan los metadatos correspondientes a la localización física (mods.location) y electrónica (sedici.identifier.uri) asignados al ítem y verifica su existencia y validez.
- **Referencia:** Se verifica a través de la existencia de identificadores persistentes asignados. En este caso, la regla valida que el Handle que fue asignado para el item y que figura en el metadato dc.identifier.uri sea válido. Se considera a un Handle como válido cuando no es el que viene predefinido por Dspace (12345678) y cuando el contenido alojado en el medatador

dc.identifier.uri es igual al guardado en la tabla Handle de la base de datos de Dspace.

- **Integridad:** se evalúa utilizando el checksum, el cual es calculado al momento de guardar el item y volviendo a calcularlo durante la validación del mismo. En este caso, se verifica que para cada *bitstream* de cada *bundle* [18] del ítem correspondiente, se mantenga su integridad. Para esto recurre al módulo *checker* [19] que brinda Dspace, que se utiliza para verificar el checksum correspondiente a cada item, comparándolo contra un checksum esperado almacenado en la base de datos. El ítem sólo es válido cuando todos los bitstreams relacionados con él han arrojado "CHECKSUM_MATCH" como resultado del chequeo. Cabe mencionar que para el caso de Dspace, el checksum se verifica y calcula usando el algoritmo MD5 [20].
- **Derechos:** todo ítem debe tener una licencia asignada, por lo que el metadato correspondiente a la Licencia CC debe existir siempre. Además, la URI de la Licencia debe también existir y debe ser la correspondiente con la Licencia elegida. En SEDICI, esto se valida chequeando la existencia y veracidad de los metadatos sedici.rights.license y sedici.rights.uri.

Con la definición de estas reglas se pudo aplicar una receta que verifique la PDI de cada elemento del repositorio, obteniendo un reporte automatizado del estado de preservación del repositorio en el momento de la ejecución.

Ejecución

Se utilizó la herramienta por primera vez, con una única regla que consistía en verificar si los elementos escogidos tenían o no un Handle [21], y si este Handle no era el predefinido para dspace (123456789).

En una primer prueba se ejecutó en un solo elemento, el 10915/10672[4], el comando de consola para realizar esta tarea quedó con el formato siguiente:

```
./dspace curate -r - -t evaluatePreservation.HandleValidation -i 10915/10672 -v
```

Se obtuvo un resultado correcto, cuyo reporte se muestra a continuación:

4 En el SEDICI, así como en la mayoría de los repositorios que utilizan Dspace, los elementos se identifican de manera unívoca con el handle que tienen asignado. Este handle es un tipo de identificador persistente, es decir, que ya el hecho de utilizarlo es en sí mismo un mecanismo de preservación.

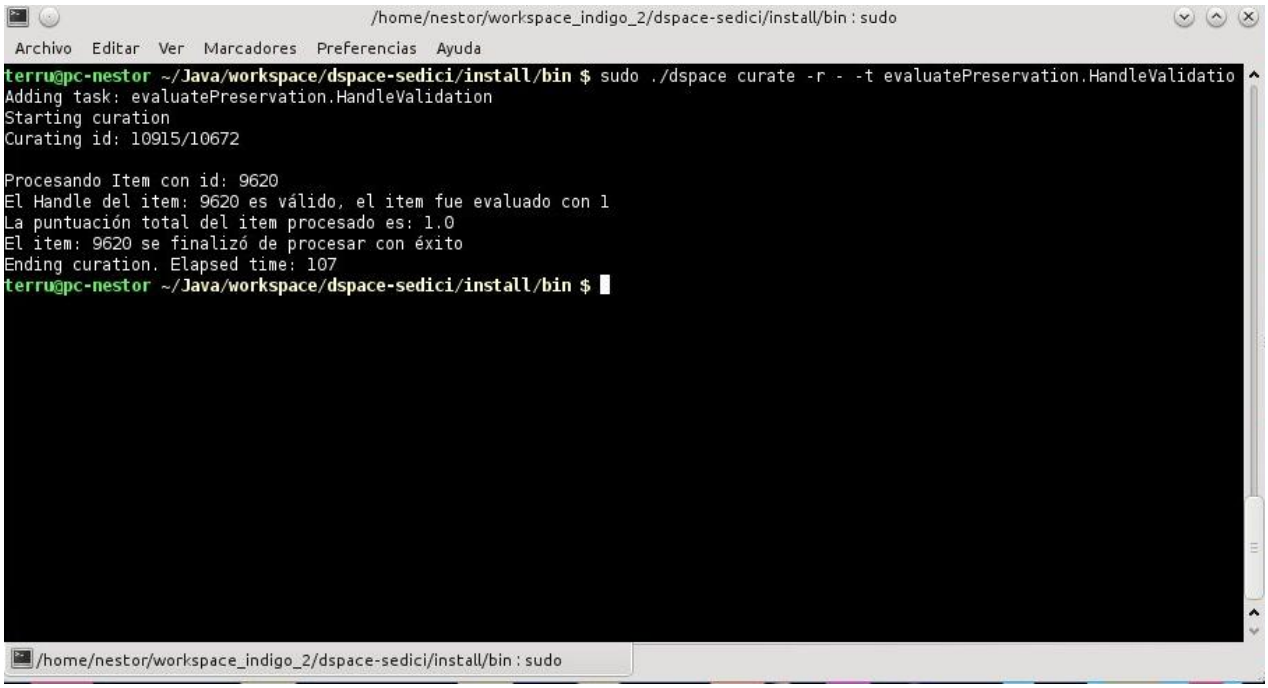

Figura 4 : consola con resultados de la ejecución para el Ítem 10915/10672. Puede verse que la validación fue exitosa debido a que se corrobora para el Ítem especificado, que el handle asignado es un handle válido.

Se probó entonces, la ejecución para una colección particular completa, identificada en SEDICI con el Handle 10915/123 (http://sedici.unlp.edu.ar/handle/10915/123) y se aprovechó el redireccionamiento[5] de la salida de la consola, para guardar el reporte en un archivo. Se obtuvo también un resultado satisfactorio de la evaluación. Se utilizó la siguiente sentencia:

./dspace curate -r - -t evaluatePreservation.HandleValidation -i 10915/123 -v > result.txt

Por último se ejecutó esta única regla para todo el repositorio, con el objetivo de evaluar su desempeño tanto en tiempo de ejecución y en utilización de memoria como en eficacia. Dado que el reporte era demasiado extenso, se lo almacenó nuevamente en un archivo en disco, para poder procesar posteriormente los resultados. La sentencia de ejecución se indica a continuación (observar el uso de la palabra clave "all" para identificar a todos los ítems):

./dspace curate -r - -t evaluatePreservation.HandleValidation -i all -v > result.txt

Los resultados para la validación del Handle en todos los elementos del repositorio fueron exitosos. La tarea de curación puede ejecutarse en un breve período de

---

5 El redireccionamiento es un método para dirigir la salida estándar de la consola hacia un archivo, con el objetivo de que esta salida pueda guardarse para análisis posteriores. Existe mucha información en línea de esta técnica, puede verse una explicación bien fundada en : http://www.robvanderwoude.com/battech_redirection.php

tiempo y utiliza la memoria de manera eficiente, esto se puede apreciar en la Figura 4, donde (aunque para un solo elemento) el tiempo estimado fue de sólo 107 ms y donde el proceso no generó una carga demasiado alta en el rendimiento general del sistema en donde se ejecuta.

Conclusiones

En varias situaciones en donde se utilizó la herramienta sobre el repositorio SEDICI, se pudieron identificar elementos que no superaron alguna validación y, por consiguiente, implementar un flujo de trabajo para realizar las correcciones pertinentes sobre los elementos. Este mecanismo de automatización de tareas de control de calidad, ha tenido resultados satisfactorios no sólo a la hora de su utilización, sino también como herramienta para evaluar posibles situaciones de mejora en las políticas de preservación del repositorio.

La visualización amigable de los resultados, ha permitido a la administración realizar pruebas y validaciones personalizadas, incluso ha sido útil para realizar algunos experimentos de evaluación del estado de preservación del repositorio [22].

Adicionalmente, este desarrollo promueve la generación rápida de nuevas reglas de evaluación gracias a su diseño estructural. Esta facilidad en la generación de reglas expuso rápidamente la posibilidad de realizar mejoras sobre la herramienta, con el objetivo de mejorar su interfaz de usuario. En este contexto ha surgido la idea de implementar un mecanismo que permita a cualquier administrador del repositorio definir y ejecutar estas reglas de validación sin la necesidad de contar con conocimientos técnicos sobre la herramienta. La definición de estas reglas podría hacerse incluso desde la interfaz de administración del repositorio y podría darse, por ejemplo, mediante un lenguaje simple de utilizar que aplique las validaciones correspondientes y devuelva los reportes adecuados.

Referencias

1. Sánchez, S., & Melero, R. (n.d.). La denominación y el contenido de los Repositorios Institucionales en Acceso Abierto : base teórica para la “Ruta Verde.” Revisado Julio, 16, 2014 desde http://digital.csic.es/handle/10261/1487
2. Yuan, L. (2011). Institutional Repositories and Digital Preservation: Assessing Current Practices at Research Libraries, *17*(5/6).doi:10.1045/may2011-yuanli
3. De Giusti, M. R., Lira, A. J., Villarreal, G. L., & Texier, J. (2012, November 1). Las actividades y el planeamiento de la preservación en un repositorio institucional. *BIREDIAL - Conferencia Internacional Acceso Abierto, Comunicación Científica Y Preservación Digital*. Revisado Julio, 16, 2014 desde http://hdl.handle.net/10915/26045
4. Hamburger, S. (2003). The State of Digital Preservation: An International Perspective. *Library Collections, Acquisitions, and Technical Services*, *27*(3), 373–374. doi:10.1080/14649055.2003.10765943

5. SEDICI - Repositorio de la Universidad Nacional de La Plata . Revisado Julio, 16, 2014 desde http://sedici.unlp.edu.ar/.
6. Universidad Nacional de La Plata (UNLP). Revisado Julio, 16, 2014 desde http://unlp.edu.ar/
7. De Giusti, M. R. (2010). SeDiCI (Servicio de Difusión de la Creación Intelectual): un recorrido de experiencias (2003-2011). *Jornada Virtual de Acceso Abierto*. Revisado Julio, 16, 2014 desde http://hdl.handle.net/10915/27160
8. DSpace 4.x Documentation - DSpace 4.x Documentation - DuraSpace Wiki. (n.d.). Revisado Julio, 16, 2014 desde https://wiki.duraspace.org/display/DSDOC4x/DSpace+4.x+Documentation
9. Overview (Java Platform SE 7). Revisado Julio, 16, 2014 desde http://docs.oracle.com/javase/7/docs/api/
10. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design patterns: elements of reusable object-oriented software.
11. Extensions and Addons Work - DSpace - DuraSpace Wiki. (n.d.). Revisado Julio, 16, 2014 desde https://wiki.duraspace.org/display/DSPACE/Extensions+and+Addons+Work
12. Curation Task Cookbook - DSpace - DuraSpace Wiki. (n.d.). Revisado Julio, 16, 2014 desde https://wiki.duraspace.org/display/DSPACE/Curation+Task+Cookbook
13. Unified Modeling Language (UML). (n.d.). Revisado Julio, 16, 2014 desde http://www.uml.org/#UML2.0
14. Riehle, D. (1997). Cap 3.4. ”Builder”. A role-based design pattern catalog of atomic and composite patterns structured by pattern purpose (p. 14). Ubilab Technical Report 97.1. 1. Zürich, Switzerland: Union Bank of Switzerland.
15. Curation System - DSpace 4.x Documentation - DuraSpace Wiki. (n.d.). Revisado Julio, 16, 2014 desde https://wiki.duraspace.org/display/DSDOC4x/Curation+System
16. Reference Model for an Open Archival Information System (OAIS), CCSDS Magenta Book. Issue 1. . (n.d.). Revisado Julio, 16, 2014 desde http://public.ccsds.org/publications/archive/650x0m2.pdf
17. DCMI Metadata Terms. (n.d.). Revisado Julio, 16, 2014 desde http://dublincore.org/documents/dcmi-terms/
18. Chapter 2.1. DSpace System Documentation: Functional Overview. (n.d.). Revisado Julio, 16, 2014 desde http://dspace.org/sites/dspace.org/files/archive/1_5_2Documentation/ch02.html#N100A1
19. Validating CheckSums of Bitstreams - DSpace 4.x Documentation - DuraSpace Wiki. (n.d.). Revisado Julio, 16, 2014 desde https://wiki.duraspace.org/display/DSDOC4x/Validating+CheckSums+of+Bitstr

eams#ValidatingCheckSumsofBitstreams-AutomatedChecksumCheckers'Results
20. Rivest, R. (n.d.). The MD5 Message-Digest Algorithm. Revisado Julio, 16, 2014 desde http://tools.ietf.org/html/rfc1321
21. Sun, S., Reilly, S., Lannom, L., & Petrone, J. (n.d.). RFC 3652-Handle System Protocol (v2.1)- November 2003. Revisado Julio, 16, 2014 desde http://www.ietf.org/rfc/rfc3652.txt
22. De Giusti, M. R., Lira, A. J., Villarreal, G. L., Terruzzi, F. A., & Adorno, F. G. (2014, March 27). Preservación digital: un experimento con SEDICI-DSpace. *XX Asamblea General de ISTEC (Puebla, México, 2014)*. Revisado Julio, 16, 2014 desde http://hdl.handle.net/10915/34889.