Desarrollo del método automático para la actualización de repositorios institucionales basados en DSpace

Development of the automated method for upgrading of institutional repositories based on DSpace

Huriviades Calderón-Gómez [1], Danny Murillo [2]*
[1] Dirección de Investigación, VIPE, Universidad Tecnológica de Panamá, [2] Dirección General de Innovación y Tecnología Educativa, Universidad Tecnológica de Panamá
[1] huriviades.calderon@utp.ac.pa, [2] danny.murillo@utp.ac.pa

Resumen– *El principal objetivo de esta investigación es desarrollar un módulo automático que actualice el código fuente en los repositorios institucionales en versiones End-of-life (EOL) de DSpace. Se hace especial énfasis en el diseño de scripting y utilización de paquetes GNU para adecuar las subrutinas a las necesidades de los desarrolladores. Asimismo, el propósito es demostrar las etapas que conlleva desarrollar el módulo y las ventajas de ejecutarlo, basado en los resultados de esta investigación. También, dicho módulo permite crear variantes o mejoras al código, ya que se encuentra bajo licenciamiento Attribution-NonCommercial-ShareAlike 4.0 International. Esta iniciativa surge para mitigar los riesgos de ataques informáticos a las vulnerabilidades conocidas.*

Palabras claves– *Repositorio institucional, DSpace, biblioteca digital, actualización a DSpace, algoritmo, plataformas digitales.*

Abstract– *The main objective of this research is to develop an automatic module to update the source code in the institutional repositories in versions End-of-life (EOL) of DSpace. Special emphasis is placed on the design of scripting and use GNU packages to adapt the subroutines to the needs of developers. In addition, the purpose is to demonstrate the stages involved in developing the module and the benefits of running it, based on the results of this research. Also, the module allows you to create variations or improvements to the code, as it is under license Attribution-NonCommercial-ShareAlike 4.0 International. This initiative comes to mitigate the risks of cyber-attacks to known vulnerabilities.*

Keywords– *Institutional repositories, DSpace, digital library, upgrading DSpace, algorithm, digital platform.*

1. Introducción

En el ámbito, de las bibliotecas modernas ha sido considerada como un factor clave la construcción de sistemas de software (Biblioteca Digital) para la disponibilidad de los contenidos en formato digital, y como esta tendencia ha llegado a ser ampliamente implementado por diferentes organizaciones (gubernamentales y no gubernamentales) para gestionar sus contenidos (investigación, estadísticas, materiales de aprendizaje y otros) [1].

Existen varias categorías de Biblioteca Digital, como, por ejemplo: archivos digitales, museos digitales o repositorios institucionales [2], [3].

La presente investigación, se centra en los repositorios institucionales, debido a las propuestas que surge bajo la cobertura de la declaración de Butapest de acceso abierto en el 2002, donde se da el apoyo de visibilizar los resultados de investigaciones por parte de la comunidad científica. En esta declaración se presentan dos alternativas, acceso abierto dorado, que es el acceso abierto a través de revistas científicas digitales con revisiones por pares y el acceso abierto verde, que es el acceso abierto a través de repositorios, mediante el archivado de los artículos científicos (postprint o versión posterior a su evaluación) de acceso abierto en formato electrónico [4]–[6].

Actualmente, existen varios tipos de repositorios institucionales como DSpace, EPrints, Greenstone, Fedora y entre otros.

Por ello, se hizo el análisis de las plataformas más utilizadas a nivel internacional, según OpenDoar [7]. En datos estadísticos de estas plataformas el 44.1% de las instituciones utilizan DSpace, 13.7% Eprints, 4.8% Digital Commons. En el caso de Centroamérica (Costa Rica, El Salvador, Honduras, Nicaragua y Guatemala), se cuentan con diecinueve repositorios institucionales de Países, donde el 63.2% utilizan DSpace, 26.3% Eprints y 10.6% otras alternativas.

Atendiendo a estas consideraciones, se seleccionó a DSpace como repositorio institucional de la Universidad Tecnológica de Panamá (UTP) por sus características, como por ejemplo: escalabilidad, robusticidad, almacenamiento distribuido, sostenibilidad e interoperabilidad [8].

En este trabajo describimos los pasos que seguimos en la construcción del módulo automático para la actualización del código fuente de los repositorios basados en versiones EOL de DSpace, debido a la problemática que traen consigo, como, por ejemplo: inestabilidad, vulnerabilidades de seguridad o incompatibilidad entre las dependencias [9].

Con esa finalidad, se propone el desarrollo del módulo mencionado para ofrecer una gestión más amigable al personal de tecnología informática en las configuraciones avanzadas de los repositorios institucionales.

Cabe destacar, que la implementación del repositorio es parte del proyecto UTP-Ridda2 [10], que tiene el objetivo de mejorar la visibilidad y alcance de la producción científica-académica de la UTP.

Actualmente, existen muchos puntos a considerar sobre este concepto, debido a esto, hemos decidido delimitarlo a cuatro secciones, para así explicar la importancia del tema y las razones que justifican su estudio: La sección 2, se describen las etapas de la metodología para el desarrollo del módulo. En la sección 3, se muestran los resultados obtenidos por las diversas pruebas de softwares. Luego, la sección 4, se vinculan los resultados con la discusión. Por último, compartimos nuestra conclusión y contribución al realizar esta investigación.

2. Metodología

Para la realización de este estudio, se ha optado por la aplicación del diseño *Bash Script* porque permite la gestión de los componentes de software (C, C++, Python o Java) dentro de un marco de trabajo modular. En cuanto al impacto en el rendimiento del módulo, este depende de la cantidad de ciclos computacionales ejecutadas por cada subrutina [11]–[13].

Entre la serie de ventajas que proporcionan aplicar este diseño se encuentran; en primer lugar, flexibilidad de modificar el código cuando sea necesario; en segundo lugar, el rápido desarrollo de las instrucciones y por último la replicación de los resultados en diferentes entornos GNU/Linux [14].

Tradicionalmente, DSpace ha sido evaluado mediante una documentación digital (wiki) para su implementación en las diferentes universidades del mundo [15], [16].

Por ello diferentes autores [17]–[19], han utilizado diversos scripts para la implementación en las diferentes versiones que ofrece DSpace; sin embargo, la mayoría de estos scripts están sujetos a versiones obsoletas de DSpace o no cuentan con el soporte por parte de los desarrolladores. En consecuencia, existe el riesgos de ataques informáticos a las vulnerabilidades conocidas [9], que ya no se limitan a un determinado proyecto, pero a menudo afectan a cientos de sistemas diferentes, por ejemplo, Maven (gestor de dependencias de Java).

Como seguimiento de esta actividad se hace necesario el análisis de las políticas de soporte en las diferentes versiones DSpace implementadas en los scripts desarrollados [20].

Por ende, se optó el algoritmo autómata [21], como esquema para la adaptación de un módulo automático para las actualizaciones más recientes (6.3) en repositorios DSpace cerca del ciclo de vida útil o por correcciones de seguridad (5.2 o superior).

2.1 Dependencias

Previo comienzo del estudio se identificaron los componentes de terceros y las herramientas necesarias para ejecutar un repositorio DSpace bajo la versión 6.x [22].

De igual manera, se debe tener presente que DSpace está construido con software de código abierto (OSS), cuyo concepto es aplicado para la mayoría de aplicaciones, sistemas operativos, bases de datos y servidores web [23]; por consiguiente, al utilizar

dependencias no acordes a los códigos de clonación o configuraciones recientes por los desarrolladores conlleva al aumento de vulnerabilidades a nivel operativo.

Para asegurarnos de que la replicación del módulo se ha llevado a cabo, es necesario realizarlo bajo las siguientes condiciones controladas:

- Software
  - Ubuntu Server 16.04 LTS.
  - PostgreSQL 9.5.x.
  - OpenJDK 8.x.
  - Apache Tomcat 7.x.
  - Apache Ant 1.9.x.
  - Apache Maven 3.3.x.
  - Npm 3.5.x.
  - Apache HTTP 2.4.x.
  - NodeJS 4.x o 6.x.
  - Git 2.x.
  - XMLUI (Mirage 2).
- Hardware (gama media para producción)
  - 4GB de RAM reservados para DSpace (2GB para Tomcat y 2GB para PostgreSQL).
  - 6GB de RAM para el sistema operativo (SO).
  - Cualquier procesador moderno para servidores (Intel Xeon o AMD Opteron).
  - 350 GB de almacenamiento HDD o SSD.

2.2 Segmentación del Módulo

Una dificultad a la que nos hemos enfrentado en este estudio es la profunda comprensión que se requiere en la estructura de la base de datos y la organización de las carpetas con el fin de actualizar solamente el código fuente de DSpace, sin afectar a los metadatos, archivos pertenecientes y prefijos de Handle existentes de los objetos recolectados. Por ello, se estableció las instrucciones de las subrutinas para un control de flujo en la actualización de los repositorios DSpace a la versión más reciente (6.3), basándonos principalmente en los múltiples resultados obtenidos por las pruebas [24]–[26].

2.2.1 Subrutina: PM_Prerequisites

Esta primera subrutina tiene como función la verificación de todas las dependencias para DSpace en el SO; de lo contrario se utilizará el Advanced Package Tool (APT) para obtener las dependencias faltantes [21].

Asimismo, a las dependencias obsoletas que se actualizarán a las señaladas previamente (por ejemplo, Tomcat 6 a Tomcat 7) es necesario seguir con las directrices de configuración para el funcionamiento óptimo del repositorio. Para ello, se deberá referir a los módulos de configuración de PostgreSQL, OpenJDK, Tomcat y Apache HTTP del algoritmo autómata [27] o consultar la documentación de cada componente individual para detalles completos y actuales [22]. En la figura 1 se muestran las tareas específicas a realizar por orden de prioridad dentro de la subrutina.

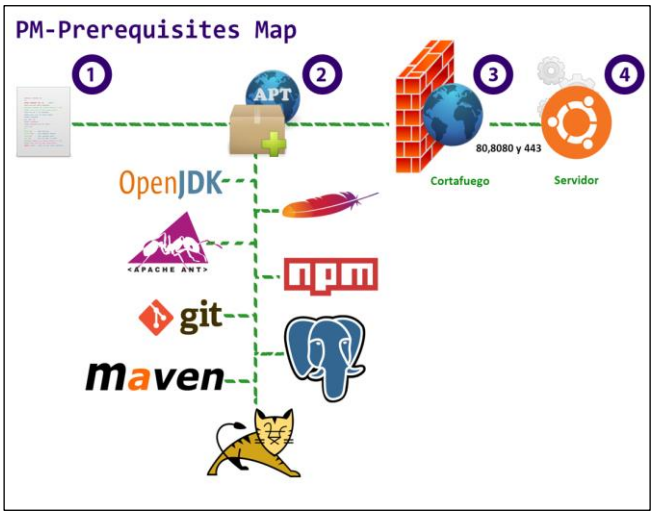


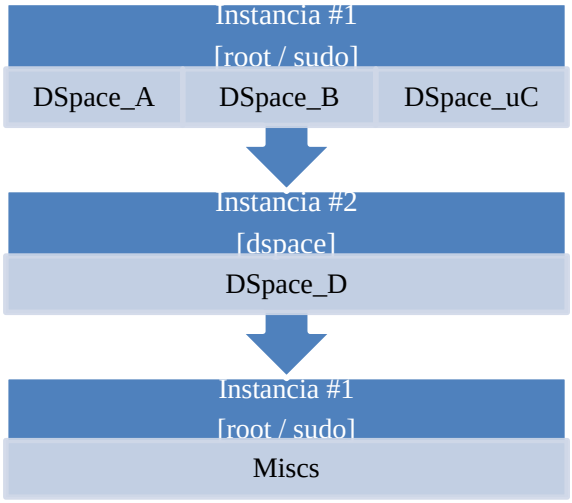
Figura 1. Diagrama de control de versiones para las dependencias (software) en las configuraciones en Ubuntu Server 16.04 LTS, por ejemplo, habilitar los puertos (80, 8080 y 443) en el cortafuego (UFW).

2.2.2 Subrutina: PM_DSpace [A, B, uC &D]

Esta segunda subrutina se encarga de la actualización del código fuente Dspace en el servidor a través de cuatros funciones: la primera, verifica la versión más reciente del DSpace usando el repositorio GitHub del proyecto, según el control de cambios de versiones [28] y asigna un directorio de construcción (dspace-source) para llevar a cabo su compilación; la segunda, crea un directorio de ejecución (dspace) para la instalación y ejecución del repositorio. Además, permite restablecer el directorio dspace con una copia de respaldo; la tercera, define las variables básicas (por ejemplo, las credenciales de la base de datos o la interfaz de usuario) al archivo de configuración del DSpace para su posterior compilación; la cuarta, compila y despliega todos los archivos de configuración (dspace.cfg, page-

structure.xml y entre otros), *command line scripts* (oai, index-discovery, harvest o generate-sitemaps) y las aplicaciones webs (oai, solr y xmlui) del repositorio. En la figura 2 se observa la secuencia lógica de estas subrutinas.

Figura 2. Diagrama de control de acceso no interactivo para la ejecución de las secuencias de comandos en las diferentes instancias y el intercambio de usuarios con diferentes niveles acceso de lectura y escritura en el repositorio.



2.2.3 Subrutina: PM_Misc [A & uB]

Esta última subrutina, proporciona los permisos necesarios para los archivos recién creados de los usuarios (dspace y tomcat7) que gestionarán al repositorio. En vista a que la construcción o actualización (mvn y ant) de las dependencias se debe realizar con usuario sin privilegios (non-root), cuya función es prevenir errores en los paquetes (por ejemplo, Bower). Por ello, se establece el valor octal 775 para los permisos de los archivos, como se muestra en la figura 3.

```
PM_Misc_uB(){
   source $DSCONF
   chmod -R 775 $DSPACE_PATH/dspace
   chmod -R 775 $DSPACE_PATH/dspace/var/oai/
   chmod -R 775 $DSPACE_PATH/dspace/log/
}
```

Figura 3. Integración del modelo ACL, para la gestión de control de acceso de los usuarios dentro del grupo dspace.

2.3 Implementación del Módulo

Para lograr la actualización exitosa del código fuente del repositorio DSpace a la más reciente, es necesario tener presente tres requisitos:

- Las versiones (5.2 o superior) de repositorios DSpace, se encuentran dentro del rango de soporte para la actualización.
- Es necesario contar con todas las dependencias (software y hardware) cumplidas.
- Es indispensable seguir con la normalización dada por la documentación de cada componente instalado [15] o por la técnica del algoritmo autónomo [21].

Atendiendo a estas consideraciones, es deber obligatorio para el interesado evitar incurrir en faltas, por tal razón este debe tener precaución al ejecutar este módulo y tener presente dichos requisitos. Para realizar la implementación, se debe ejecutar dicho script como usuario con privilegios y utilizar el comando de ejecución (**. ./scritp.sh**), por ejemplo, en la figura 4 se muestra la ejecución del módulo.

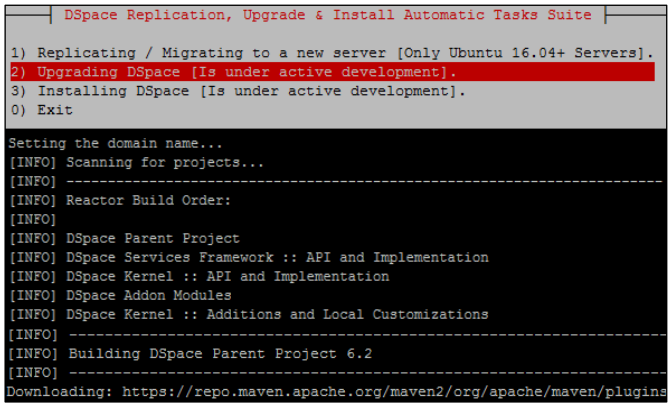


Figura 4. Ejecución del módulo para la actualización de un repositorio DSpace de la versión 5.2 a la 6.3 en un Ubuntu Server 16.04.4 LTS con 1,994 metadatos indexados.

3. Resultados

Para evaluar la eficiencia del módulo se utilizó pruebas de software como prueba de integración y prueba del sistema. Basándonos en estos resultados, se realizaron las adecuaciones pertinentes entre las versiones “Alpha” y “Candidata a Definitiva (RC)” para la estabilidad del algoritmo, cuyo proceso demoró seis meses de desarrollo en conjunto con pruebas de regresiones. La siguiente figura ilustra la actualización dada a un repositorio.

Figura 5. Actualización dada a un repositorio DSpace a la versión más reciente, utilizando el módulo automático. La parte superior es un repositorio con metadatos bajo la versión 5.2 en un servidor local, y la parte inferior es la actualización dada al mismo repositorio bajo la versión 6.3.

3.1 Prueba de Integración

Generalmente, la integración es el proceso de depuración de los fragmentos o componentes de códigos (subrutinas) son agregados para crear el algoritmo principal (módulo), cuya función es exponer los problemas que pueden surgir en esta etapa de integración [25]. Sin embargo, las subrutinas son aceptables cuando se prueba individualmente en forma aislada, de hecho, podrían resultar en un comportamiento incoherente o erróneos cuando se combinan con el fin de construir algoritmos complejos.

Por ello, se realizaron varias pruebas preliminares: la primera, son el uso de script estáticos para la comprobación de la sintaxis del lenguaje (paso a paso), coherencia e integridad, así como la adhesión a convenciones establecidas o controladas (por ejemplo, utilizar *shellcheck*); y la segunda, es el uso de script dinámicos para llevar a cabo simulación, tiempo de análisis y de prototipado de la data ingresada [25].

Por consiguiente, la estrategia debe tener un equilibrio entre el número de entradas y el tiempo-esfuerzo dedicados a fines de realizar estas pruebas. Además, se recomienda utilizar *chroot (change root)* para el intercambio de los ficheros de la raíz y sus procesos actuales en ejecución (sistema operativo) a un directorio (Chroot Jail) determinado para ejecutar los scripts en un entorno controlado [29], y así evitar causar daños irreversibles al sistema. La figura 6 ilustra el uso de estas pruebas.

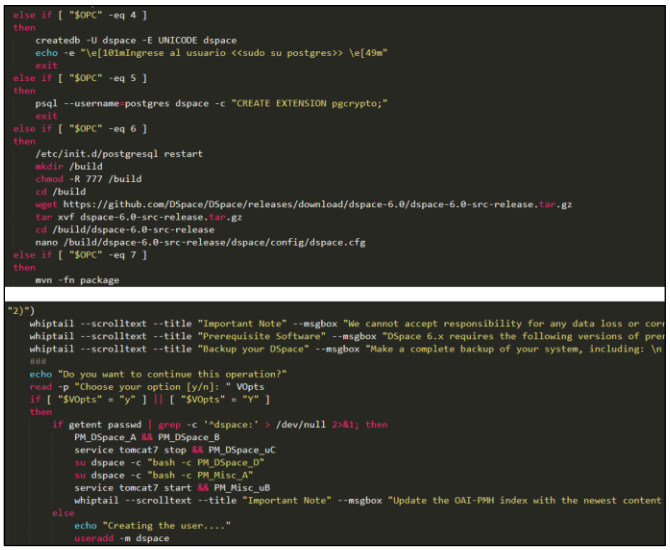


Figura 6. Ejecución exitosa de las pruebas de integración a las diferentes versiones del módulo. La parte superior se basa en la versión Alpha (0.9) con pruebas estáticas bajo una secuencia establecida, y la parte inferior se basa en la versión RC (1.5) con pruebas dinámicas bajo la verificación de parámetros, subrutinas e intercambios de instancias.

3.2 Prueba del Sistema

En particular, este tipo de prueba intenta revelar errores que no pueden ser atribuidas a componentes como tales, a las incoherencias entre los componentes, o a las interacciones de componentes y otros objetos, por ejemplo, descubrir los fallos que se manifiestan sólo a nivel del sistema y por lo tanto no fueron detectados durante las pruebas de integración o unidad [25].

Como complemento, la prueba del sistema incluye un conjunto de pruebas especializadas como:

- Prueba de Rendimiento.
- Prueba de Seguridad.
- Prueba de Fiabilidad.
- Prueba de Estrés.
- Prueba de Compatibilidad.
- Prueba de Recuperación.

Para este caso se utilizó las pruebas de compatibilidad y de estrés, bajo un entorno de servidor virtual privado (KVM-QEMU) con diferentes emulaciones, cuyo objetivo es definir un perfil operacional necesaria para el análisis estadístico de la confiabilidad del módulo.

3.2.1 Prueba de Compatibilidad

Esta primera prueba tiene como objetivo la verificación de las funcionalidades del repositorio en diferentes versiones DSpace sin afectar a las dependencias recomendadas. A partir de los datos de la tabla 1, se evidencia el rango de soporte para la actualización exitosa.

Tabla 1. Rango de versiones soportadas para la actualización utilizando el módulo automático.

Versión DSpace	Soportado		Tiempo Estimado (minutos)
	Sí	No	
6.1			15 ~ 25
6.0			15 ~ 25
5.8			25 ~ 30
5.7			25 ~ 30
5.6			30 ~ 35
5.5			30 ~ 35
5.4			35 ~ 45
5.3			35 ~ 45
5.2			35 ~ 45
5.1			---
5.0			---
4.x			---
3.x			---
1.x			---

La tabla 1 proporciona una visión general del tiempo estimado que se tomaría en la actualización del DSpace, se evidencian un aumento en el tiempo en versiones EOL (5.x), debido a que se requiere una etapa adicional en actualizar las dependencias de software. También, hay que tener presente que el tiempo estimado puede variar por diversos factores como: gama del hardware, banda ancha de Internet y al volumen de metadatos almacenado en el repositorio.

3.2.2 Prueba de Estrés

Esta última prueba tiene como objetivo la evaluación de las funcionalidades del repositorio, después de su actualización a la versión más reciente con el módulo.

Para ello, se tomó como muestra a 50 usuarios interactuando con la plataforma durante 5 minutos. La figura 7 presenta un resumen de los resultados obtenidos en un análisis con los softwares Apache JMeter y Load Impact, cuyo resultado demuestra una correlación positiva entre el rango estipulado de aceptación (menos de 1500 ms) [30], y el tiempo promedio de respuesta (762.5 ms ~ 1300 ms) del repositorio.

Figura 7. Resultado de la prueba de estrés con el máximo de usuarios (50) conectados al repositorio actualizado. Las líneas de colores representan varios parámetros como: la línea azul, es el tiempo de respuesta de cada usuario; la línea verde, son los usuarios activos en el repositorio; la línea morada, son las solicitudes por segundo de los usuarios; la línea naranja, es la carga de memoria generada por los usuarios.

4. Discusión

Este estudio se propuso con el objetivo de evaluar la importancia de desarrollar un módulo automático para la actualización del código fuente en los repositorios DSpace, por lo que hemos identificado las principales tareas adoptado por la comunidad de usuarios DSpace y el uso de algoritmo autónomo para llevar a cabo esta actividad [3], [9].

Hay varias contribuciones significativas de este estudio: la primera, a diferencia de la técnica tradicional de repositorios DSpace para la actualización, nuestro enfoque permite replicar los resultados del análisis y conocimiento adquiridos para formar parte de un esquema automático en otros repositorios; la segunda, se demuestran los problemas de las vulnerabilidades de seguridad conocidas por la comunidad de usuarios DSpace (por ejemplo, en la versión 6.0 soportaba la autentificación con *Shibboleth* por REST API) [28], al utilizar componentes de terceros o versiones DSpace obsoletos. Esto se debe a que menudo son desconocidos por los desarrolladores, debido a la falta de documentación sobre las correcciones de errores o la falta de la herramienta para identificar estas dependencias directas e indirectas causadas por la reutilización de los componentes vulnerables [9]; la tercera, al utilizar el método de scripting, aunque preliminar, sugiere que este módulo pueda adaptarse a

la replicación o migración del repositorio a otros servidores sin afectar los handles indexados o cosechados por otros sistemas, bajo la condición de mantener el mismo dominio web.

Otro hallazgo importante fue que las versiones no soportadas (4.x, 3.x, 2.x y 1.x) para llevar a cabo la actualización a través del módulo, es necesario utilizar FlywayDB (actualiza la estructura de la base de datos acorde con la versión más reciente), redefinir las dependencias y los archivos de configuración o personalizados con la actualidad, y de realizar una actualización manual progresivamente hasta la versión 5.2 para una posterior actualización automática de versión.

Es importante tener en cuenta el posible riesgo (corrupción de los metadatos o tiempo de inactividad) de implementar el módulo directamente a un servidor de producción sin haber realizado una prueba preliminar.

Por ello, se recomienda siempre realizar una actualización al repositorio en un servidor de desarrollo o de prueba, y de realizar una copia de seguridad del repositorio (dspace, base de datos y archivos personalizados).

5. Conclusión

Con el desarrollo del presente estudio se logró completar el objetivo propuesto, obtener un módulo estable y modificable, según las necesidades del entorno para la actualización de repositorios DSpace dentro del rango de soporte.

Al mismo tiempo, se logró demostrar todos los pasos involucrados en la metodología para el desarrollo de las subrutinas utilizadas en el módulo, lo cual permite a los usuarios desarrollar sus propios módulos o para crear sus variantes para la gestión del repositorio (por ejemplo, replicación o integración de subcomponentes), ya que el módulo está bajo licencia Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

Como trabajos futuros se tiene contemplado el soporte para CentOS y Arch Linux. Además, sería de interés implementar otras funcionalidades como, por ejemplo: manipular dinámicamente la ruta de instalación del repositorio, incluir el soporte a OracleDB y de adaptar el módulo para futuras versiones DSpace (7.x).

Como complemento a los resultados de esta investigación, compartiremos la URL (https://github.com/bankashi/Scripts-for-DSpace) del proyecto que aloja la versión RC del módulo.

6. Referencias

[1] D. Hahn, “Acceso Abierto y Bibliotecas Digitales: Bibliotecas de Ciencias Sociales en Acción [Open Access and Digital Libraries: Social Science Libraries in Action], Lynne M. Rudasill, Maria Elena Dorta-Duque (Eds.), IFLA Publications (2013), (Nr 158. 346 pp.), ISBN: 978-3-11-028102-6,” *J. Acad. Librariansh.*, vol. 40, no. 3–4, pp. 420–421, May 2014.

[2] H. Chen and Y. Zhang, “Functionality Analysis of an Open Source Repository System: Current Practices and Implications,” *J. Acad. Librariansh.*, vol. 40, no. 6, pp. 558–564, Nov. 2014.

[3] S. Bangani, “The History, Deployment, and Future of Institutional Repositories in Public Universities in South Africa,” *J. Acad. Librariansh.*, vol. 44, no. 1, pp. 39–51, Jan. 2018.

[4] J. Berrocoso Valverde, “El acceso abierto al conocimiento científico,” *Reuni+D*, p. 56, 2013.

[5] T. H. Peréz, D. R. Mateos, and G. B. D. la Fuente, “Open Access: el papel de las bibliotecas en los repositorios institucionales de acceso abierto.,” *An. Doc.*, vol. 10, pp. 185–204, 2008.

[6] F. M. Sánchez-Martín, F. Millán Rodríguez, and H. Villavicencio Mavrich, “La Iniciativa Open Access (OAI) en la literatura científica,” *Actas Urológicas Españolas*, vol. 33, no. 7, pp. 732–740, Jan. 2009.

[7] OpenDoar, “Repositorios Centroamericanos,” 2017. .

[8] R. S. Project, “Repository software survey,” 2010. .

[9] S. S. Alqahtani, E. E. Eghan, and J. Rilling, “Tracing known security vulnerabilities in software repositories - A Semantic Web enabled modeling approach,” *Sci. Comput. Program.*, vol. 121, pp. 153–175, 2016.

[10] UTP-Ridda2, “¿Qué es UTP-Ridda2?,” *UTP-Ridda2*, 2017. [Online]. Available: http://ridda2.utp.ac.pa/que-es-utp-ridda2. [Accessed: 23-Jul-2018].

[11] D. M. Beazley, “Automated scientific software scripting with SWIG,” *Futur. Gener. Comput. Syst.*, vol. 19, no. 5, pp. 599–609, Jul. 2003.

[12] P. Ralph, “The two paradigms of software development research,” *Sci. Comput. Program.*, vol. 156, pp. 68–89, May 2018.

[13] M. L. Scott and M. L. Scott, “Scripting Languages,” in *Programming Language Pragmatics*, Elsevier, 2009, pp. 649–724.

[14] G. Speake and G. Speake, “Using BASH,” in *Eleventh Hour Linux+*, Elsevier, 2010, pp. 61–77.

[15] The DSpace Developer Team, “DSpace 6.x Documentation,” no. July, 2017.

[16] Stellenbosch University Library, “SUNScholar/Install

DSpace/S04/5.X," *Libopedia*, 2016. [Online]. Available: http://wiki.lib.sun.ac.za/index.php?title=SUNScholar/Install_DSpace/S04/5.X.

[17] D. T. Palmer, A. Bollini, S. Mornati, and M. Mennielli, "DSpace-CRIS@HKU: Achieving Visibility with a CERIF Compliant Open Source System," *Procedia Comput. Sci.*, vol. 33, pp. 118–123, Jan. 2014.

[18] P. Rodrigo, "Dspace Auto Install." Github, 2015.

[19] Libopedia contributors, "SUNScholar/Rebuild DSpace." Libopedia, 2016.

[20] T. Donohue, "DSpace Software Support Policy," 2017.

[21] H. Calderón-Gómez and D. Murillo, "Algoritmo autómata para la instalación estructurada Dspace en Ubuntu, utilizado en la implementación del repositorio institucional de la Universidad Tecnológica de Panamá," in *XX Congreso Internacional EDUTEC – 2017*, 2017.

[22] The DSpace Developer Team, "Prerequisite Software," 2017. [Online]. Available: https://wiki.duraspace.org/display/DSDOC6x/Installing+DSpace#InstallingDSpace-PrerequisiteSoftware.

[23] S. Kim and H. Lee, "Software systems at risk: An empirical study of cloned vulnerabilities in practice," *Comput. Secur.*, Feb. 2018.

[24] M. P. Prado and A. M. R. Vincenzi, "Towards cognitive support for unit testing: A qualitative study with practitioners," *J. Syst. Softw.*, vol. 141, pp. 66–84, Jul. 2018.

[25] F. Lonetti and E. Marchetti, "Emerging Software Testing Technologies," *Adv. Comput.*, vol. 108, pp. 91–143, Jan. 2018.

[26] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, and R. Tumeng, "Test case prioritization approaches in regression testing: A systematic literature review," *Inf. Softw. Technol.*, vol. 93, pp. 74–93, Jan. 2018.

[27] H. Calderón-Gómez, "DSpace 6.x Scripts (RC)." Bitbucket, Panama, 2017.

[28] N. Pascal, "Changes in 6.x." 2017.

[29] H. C. . van Tilborg and S. Jajodia, "Chroot," in *Encyclopedia of Cryptography and Security*, 2011.

[30] X. S. Wang, A. Krishnamurthy, and D. Wetherall, "Speeding up Web Page Loads with Shandian," *Ndsi*, p. 15, 2016.