# *CLOVeR*: An Optimized Repository for Customizable Learning Objects

José Wallison F. da Silva [1], Cidcley T. de Souza [2]
Graduate Program in Computer Science
Federal Institute of Ceará - IFCE
Fortaleza, Brazil
wallison.felix@ppgcc.ifce.edu.br [1], cidcley@ifce.edu.br [2]

Maria de Fátima C. de Souza [3]
Institute Virtual University
Federal University of Ceará - UFC
Fortaleza, Brazil
fatimasouza@virtual.ufc.br [3]

***Abstract*—Despite reusability being a Learning Objects (LO) core feature, sometimes they demand modifications to fit the new use context. This adaptation process must be easy and rapid. However, several LO are produced in a way that changes need to be done at source code, requiring technical knowledge. Guided Customization (GC) aims at enabling the user to execute adaptations without this requirement. It allows changes at interface level, which simplifies customization of some resource aspects. LO with GC strategy are named Customizable Learning Objects (CLO). However, when CLO and their Customized Versions (CV) are stored like a package of files in a traditional Learning Object Repository (LOR), duplications are scattered throughout the repository, since different CV of same CLO have common unchanged files. This paper presents a proposal of repository that avoids this replication. Considering that CLO executable and media files (images, audios and videos) are the ones which consume more space in repository, repository evaluation demonstrates that in "worst case scenario" (all media files are replaced on customization) the proposal behaves like a traditional LOR concerning disk space consumption. In the "best case scenario" (no media file is replaced) the proposed approach proves more efficient, whereas it shares unchanged media.**

***Keywords-learning object; guided customization; customizable learning object; learning object repository***

## I. INTRODUCTION

In the last years, many definitions have been proposed for Leaning Objects (LO), including definitions considering the learning aspect and the technological aspect [1]. Between them, one of the most widespread definition and used in this work is the Wiley's definition. It describes LO as "any digital resource that can be reused to support learning" [2]. Despite LO's definitions differ in some aspects, reusability is a LO core aspect listed in all definitions [1].

The reuse of LO affects education in two ways: it reduces costs in general and contributes to the increase in quality of resources [3]. However, for LO to be reused, it is necessary to publish them in repositories that allows potential users to find them. Named as Learning Object Repository (LOR), they are online databases to store, manage and share LO, where these resources can be found by metadata [4] [5].

However, the sharing this resource type is not sufficient for enabling its reuse. In order to be reused effectively, the resource needs to fit new use context or to allow adaptation [1]. Reference [6] said that educational resources are free to be used by others only if they enable users to execute four actions (known as "4Rs"). *Revise* (to adapt, adjust, modify, or alter the content) is one of them. Even so, there is no guarantee that users have technological and pedagogical capacity to execute the necessary adaptations [7]. Since several LO still are produced as monolithic blocks which modifications need to be done at source code level, requiring technical knowledge [8] [9].

In this context, Guided Customization (GC) was applied to learning objects of animation/simulation type, a procedure that gave rise to Customizable Leaning Objects (CLO) [9]. GC is a strategy that allows the CLO users to execute customizations at resource interface level without demand any technical knowledge [9]. This guarantees more autonomy to teachers and provides financial advantages, since the time for adaption stage is reduced [8] [9]. GC also limits customizations according to user's Degree of Freedom (DF), a permission level assigned based on user profile. It objectives to ensure that modifications do not remove the initial pedagogical goals of CLO [8] [9].

CLO are composed by a set of linked screens named as Scenes, which are formed by a group of components with pedagogical potential that can be of five types: *text*; *button*; *image*; *audio*; and *video*. Each one of these components have a set of attributes that values determine the component state [9]. CLO eliminates some barriers of cultural type that limits reuse of LO. With them, for example, users can translate all texts in a simple way and can replace media files (images/videos) by media files more familiar to leaners [9].

Since LO of animation/simulation type are composed of executable file, media files and control flow, it is not recommended to store CLO and their Customized Versions (CV) in traditional LOR, because these repositories store LO as a package of all files that compose them, working as a LO deposit [10]. In this case, many duplications are scattered throughout repository, since different CV of same CLO have common unchanged files that could be shared between them.

This paper presents a proposal of repository most appropriate for CLO. It includes a versioning strategy, which defines the relation between CV, facilitating its management and avoiding duplication of CLO unchanged files.

## II. CLOVER REPOSITORY

The proposed repository was named as *CLOVeR* (Customizable Learning Object VErsioning Repository). It is accessible and manageable from *CLOWebPlatform*, an open and responsive web platform, like recommended in [7].

The platform has functionalities to store, search, browse (browsing in CV tree of a CLO) and download CLO and CV. These operations are considered core functions related to LO aspect in a LOR [11]. In relation to metadata aspect, it is possible to store and view resources metadata, core functions for this aspect in a LOR [11]. The platform also provides a RESTful API, whereby it is possible to add new CLO/CV in the *CLOVeR* and to get specific CLO/CV stored in repository by their URI. It enables communication (add/get resources) in an automatic way, allowing to link *CLOVeR* with other applications.

The repository also stores information about users that can sign up to receive a DF to customize CLO and access authorization for *CLOWebPlatform* restricted operations.

The resources submit to *CLOVeR* and the resource retrieval from it happens in package format (Fig. 1). These packages encapsulates elements that compose resources, such as executable and media (images, audios and videos) files. Packages also includes JSON files with information about the resource, as metadata based on IEEE LOM Standard [12] and components states in the resource version.

### A. *CLO Deployment Package* (CLO-DP)

Illustrated in Fig. 1(a), it is the package format to include a new CLO in *CLOVeR*. It contains: *LOM.json*, a file with CLO metadata, but with a structure flexible; *MANIFEST.MF* file, a text file with information for the CLO-DP processing; and *executable_files* directory, a directory with one subdirectory to each CLO executable file, whereas the same CLO can need different executables to distinct execution environments. Subdirectories in *executable_files* are named as the executable file extension and is composed by: executable file itself; *CLOName_extension.json* (executable file metadata); *CLOName.json* (all components grouped by scenes); and *components* subdirectory (media files).

### B. *CLO Package* (CLO-P)

Illustrated in Fig. 1(b), it is the package format that CLO and CV are downloaded from repository. It packages: CLO executable file required; *CLOName.json* (file containing the state of all component on desired version); *components* directory (media files); and a *token.txt* (file to control resource customization and CV inclusion in repository).

### C. *CV Deployment Package* (CV-DP)

Illustrated in Fig. 1(c), it is the package format used to add a new CV in *CLOVeR*. It is formed by: *CLOName.json* (file with the new components states after customization); *components* directory (new media files if any was replaced on customization); and the *token.txt* from CLO-P source package.

## III. Implementation of CLOVeR Repository

The *CLOVeR* is composed of two databases, a model adopted by some LOR [13]. CLO and their CV are stored in a non-relational document oriented database, the MongoDB, which stores documents in a JSON binary format [14]. Information about registered users and access authorization to *CLOWebPlatform* are persisted in a PostgreSQL database.

### A. *CLOVeR Repository Data Model*

When a new CLO is included in *CLOVeR*, three JSON file types packaged in the CLO-DP are stored like three different MongoDB documents types. *Root Descriptor*

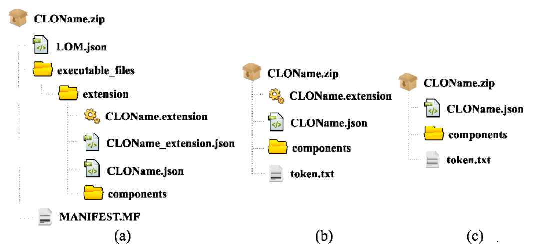

Figure 1. Package types structure: (a) CLO-DP; (b) CLO-P; (c) CV-DP.

(RDescriptor) saves the CLO metadata defined in *LOM.json*. One *Executable File Descriptor* (EFDescriptor) is created for each CLO executable file to store metadata defined in *CLOName_extension.json*. The components state is persisted as *Components Descriptor* (CDescriptor), one for each CLO executable file. Still on CLO inclusion, executable and media files are persisted in server filesystem in a directory structure intended for CLO. Descriptors references them by their path.

In respect of the CV inclusion, each new version is stored like as a *Version Descriptor* (VDescriptor) in MongoDB. This descriptor are composed by: version number; a small number of metadata describing the version; reference to EFDescriptor of hierarchy wherein is inserted; reference to version from it was generate; and component attributes modified in relation to initial components state.

All relationships result on a CLO *Versioning Hierarchy*. Fig. 2 exemplifies it. Despite it is not recommended defining relationships on non-relational databases, tree data structure is well supported by MongoDB [15].

The versioning proposed guarantees control version as advised for LO, capturing the syntactical and the semantic changes [16], relating versions and preventing that a version impact the use of any other version [17].

One of the versioning strategy aspect aims at avoiding data replication is that only differences between version and initial components states are stored in VDescriptor. In this case, just one merge is necessary to obtain a CV regardless of level in the hierarchy. This was inspired on Git, a control version system that aims to minimize merge amount required to retrieve a file version [18]. The other aspect is related to executable and media files, which consume more space in repository. These files are saved in a directory structure where executable file is kept in subdirectory named as executable file extension, while media files are kept in *components* subdirectory. They are stored once to all *Versioning Hierarchy*. For each new version, only media

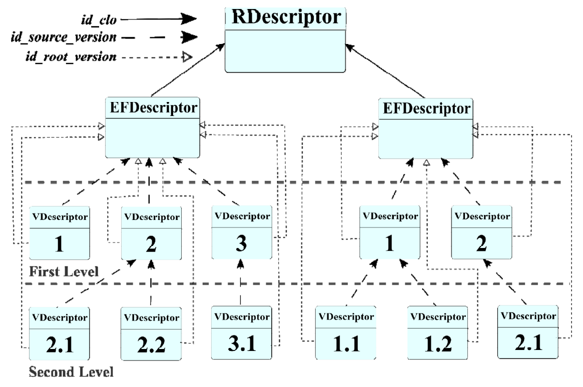

Figure 2. Example of CLO *Versioning Hierarchy*.

files replaced on customization process are persisted. This approach avoids the replication of unchanged media files between Customized Versions of a CLO.

## IV. Evaluation and Results

The *CLOVeR* efficiency was evaluated comparing Disk Space Consumption (DSC) with a traditional LOR, wherein LO are stored as a package of files [10]. The evaluation was realized using a CLO with 1 executable file (5MB) and 100 components distributed in 5 scenes, being 55 among them of media type (1MB each). With compression strategy used, it resulted in a CLO-P of 54,121KB.

Three test cases were executed. CLO was customized four times on each one, but with a different amount of replaced media files. Since executable and media files are responsible for more space occupation in the repository, how much more media file are replaced on customization process, more space are allocated to new version:

- *Worst Case Scenario*: fifty-five media files replaced for others of the same size (1MB).
- *Medium Case Scenario*: twenty-eight media files replaced for others of the same size (1MB).
- *Best Case Scenario*: no media file replaced.

DSC verifications were executed in an environment running Windows 10 OS with NTFS filesystem. The values were obtained in *kilobytes* adding the space occupied in MongoDB (shell command `db.stats().dataSize`) and the filesystem space used to store executable and media files.

If a traditional LOR was used, on three test cases a new CLO-P would be inserted in integral form for each version. Considering that CLO-P used on test cases had 54,121KB, it was possible to make a projection of DSC. Fig. 3 illustrates the test cases results and shows the *CLOVeR* consumption behavior in comparison with a traditional LOR.

In *Worst Case Scenario*, *CLOVeR* behaves like a traditional LOR about DSC, having a growth rate slightly higher. However, in the two other scenarios (*Medium* and *Best*) the results demonstrate that *CLOVeR* is substantially efficient when compared with traditional LOR, whereas unchanged media files are not replicated. These results are meaningful, as the *Worst Case Scenario* tend to occur rarely.

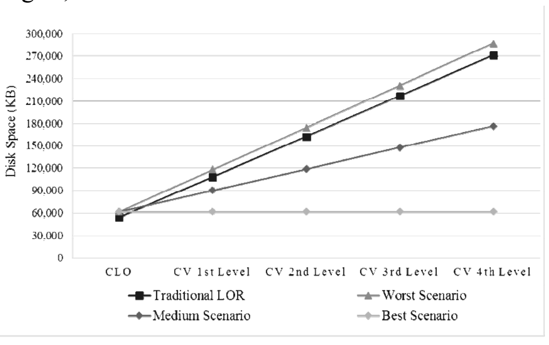

Figure 3. *CLOVeR* and traditional LOR disk space consumption.

## V. Conclusion and Future Work

The proposed repository has a versioning of resource, an important aspect for LOR to stimulate LO reuse [5], that guarantee efficiency storage of CLO and its CV and enables a flexible resource management. Therefore, it is more recommended use *CLOVeR* as repository for sharing these resources than using a traditional LOR. However, this efficiency could be enhanced if common media files were not replicated also between different CLO. For this reason, next proposed repository version will include a versioning of media file in order to control its versions and to centralize files referenced by resources.

## References

[1] J. Sinclair, M. Joy, J. Yau, S. Hagan, "A Practice-Oriented Review of Learning Objects," *IEEE Transactions on Learning Technologies*, vol. 6, no. 2, 2013, pp. 177-192.

[2] D. Wiley, "Learning Object Design and Sequencing Theory," PhD disertation, Dept. of Instructional Psychology and Technology, Brigham Young Univ., Provo, 2000.

[3] D. Sampson, P. Zervas, "A Workflow for Learning Objects Lifecycle and Reuse: Towards Evaluating Cost Effective Reuse," *Educational Technology & Society*, vol. 14, no. 4, 2011, pp. 64–76.

[4] R. Lehman, "Learning object repositories," *New Directions for Adult and Continuing Education*, vol. 2007, no. 113, 2007, pp. 57-66.

[5] R. McGreal, "A Typology of Learning Object Repositories," *Handbook on Information Technologies for Education and Training*, Springer, 2008, p. 5-28.

[6] D. Wiley, T. Bliss, M. McEwen, "Open Educational Resources: a review of the literature," *Handbook of Research on Educational Communications and Technology*, Springer, 2014, p. 781-789.

[7] N. Butcher, *A Basic Guide to Open Educational Resources (OER)*, UNESCO and Commonwealth of Learning, 2015.

[8] M.F. Souza, J.A. Castro-Filho, R. Andrade, "Model-driven Development in the Production of Customizable Learning Objects," *Proc. 10th IEEE International Conference on Advanced Learning Technologies* (ICALT 2010), IEEE, 2010, pp. 701-702.

[9] M.F. Souza, J.A. Castro-Filho, R. Andrade, "Applying Model-Driven Development for Building Customizable Learning Objects," *IEEE Technology and Engineering Education*, vol. 6, no. 1, 2011, pp.22-29.

[10] H. dos Santos, G. Carrillo, C. Cechinel, X. Ochoa, "Towards the use of Semantic Learning Object Repositories: evaluating queries performance in two different RDF implementations," *Bulletin of the IEEE Technical Committee on Learning Technology*, vol. 16, no. 4, 2014, pp. 6-9.

[11] G. Sampson, P. Zervas, "Learning Object Repositories as Knowledge Management Systems," *Knowledge Management & E-Learning: An International Journal*, vol. 5, no. 2, 2013, pp. 117-136.

[12] *IEEE Std 1484.12.1*, *Learning Object Metadata* (LOM), IEEE, 2002.

[13] X. Ochoa, G. Carrillo, C. Cechinel, "Use of a Semantic Learning Repository to Facilitate the Creation of Modern e-Learning Systems," *Proc. 15th International Conference on Human Computer Interaction* (INTERACCION 2014), ACM, 2014, pp. 92-98.

[14] *BSON Types*, MongoDB Inc.; https://docs.mongodb.com/manual /reference/bson-types/.

[15] *Model Tree Structures in MongoDB*, MongoDB Inc.; https://docs .mongodb.com/manual/tutorial/model-tree-structures/.

[16] C. Brooks, J. Cooke, J. Vassileva, "Versioning of Learning Objects," *Proc. 3rd IEEE International Conference on Advanced Learning Technologies* (ICALT 2003), IEEE, 2003, pp. 296-297.

[17] M. Tate and D. Hoshek, "A model for the Effective Management of Re-usable Learning Objects (RLOs): Lessons from a Case Study," *Interdisciplinary Journal of E-Learning and Learning Objects*, vol. 5, 2009, pp. 51-73.

[18] S. Chacon and B. Straub, *Pro git*, Apress, 2014, p. 497-501.