Proposta de Repositório para o Versionamento de Objetos de Aprendizagem Customizáveis

José Wallison F. da Silva[1], Cidcley T. de Souza[1], Maria de Fátima C. de Souza[2]

[1]Programa de Pós-Graduação em Ciências da Computação (PPGCC)
Instituto Federal do Ceará (IFCE) – Fortaleza, CE - Brasil

[2]Instituto Universidade Virtual (UFC Virtual)
Universidade Federal do Ceará (UFC) – Fortaleza, CE - Brasil

wallison.felix@ppgcc.ifce.edu.br, cidcley@ifce.edu.br,
fatimasouza@virtual.ufc.br

Abstract. *This paper describes the Versioning strategy of resources on the Customizable Learning Object VErsioning Repository (CLOVeR), repository to store, share and manage Customizable Learning Objects. The repository evaluation revealed that resources inclusion time changes linearly with executable and media files number. For resource get, the time stay in same range, regardless of resources aspects. The evaluation also demonstrated executable and media files are the elements that define the space necessary for storage each resource. And that CLOVeR optimize the use of this space in the same way on scenarios with few and scenarios with many media files.*

Resumo. *Este artigo descreve a estratégia de Versionamento de recursos do Customizable Learning Object VErsioning Repository (CLOVeR), repositório destinado ao armazenamento, compartilhamento e gerenciamento de Objetos de Aprendizagem Customizáveis. A avaliação deste repositório revelou que o tempo de inclusão de recursos varia linearmente com o número de executáveis e mídias do recurso e que, no caso da obtenção, o tempo se mantém em uma mesma faixa, independentemente do recurso. A verificação também evidenciou que são os executáveis e as mídias que determinam o espaço necessário para o armazenamento dos recursos e que o CLOVeR otimiza o uso deste espaço de forma semelhante em cenários com poucas e com muitas mídias.*

1. Introdução

Objetos de Aprendizagem (OAs) possibilitam aos docentes diversificarem a abordagem de conteúdos [Wiley 2000]. Muitas definições, baseadas tanto em aspectos educacionais quanto em aspectos tecnológicos, foram apresentadas para eles ao longo dos anos [Sinclair et al. 2013]. Para este artigo, a difundida conceituação de Wiley (2000) foi adotada, pois restringe OAs a recursos digitais. Ela os define como "quaisquer recursos digitais que possam ser reutilizados como suporte à aprendizagem" [Wiley 2000].

Apesar das diferenças entre os vários conceitos apresentados, os OAs possuem algumas características consideradas centrais e que são comuns a todas as definições. A *Reusabilidade* é uma delas [Ieiri e Braga 2015, Sinclair et al. 2013]. Porém, para que sejam reutilizados, esses recursos devem ser compartilhados de um modo que possam ser facilmente localizados [Cechinel e Ochoa 2014]. Os Repositórios de Objetos de Aprendizagem (ROAs) são as ferramentas pelas quais é feito esse compartilhamento. ROAs são bases digitais destinadas ao armazenamento, organização, classificação e disponibilização de Objetos de Aprendizagem que, por sua vez, são descritos e indexados por metadados [Ieiri e Braga 2015, Tzikopoulos et al. 2009].

DOI: 10.5753/cbie.sbie.2017.324

Mesmo com o compartilhamento, esses recursos nem sempre podem ser reutilizados no estado em que são disponibilizados. O reúso de OAs está mais relacionado à aplicabilidade deles a novos contextos de uso [Sinclair et al. 2013]. Logo, é comum que eles demandem modificações tanto para a correção de erros ou realização de atualizações quanto para a adaptação a outro cenário de uso [Silva e Souza 2016]. Para Sinclair et al. (2013), os próprios recursos têm de permitir, de modo simplificado, a execução de alterações por parte dos seus usuários. Para tanto, questões relacionadas à forma como são licenciados e estruturados devem ser consideradas durante a concepção [Sinclair et al. 2013]. Contudo, a maioria dos OAs ainda são produzidos como blocos monolíticos, limitando a realização de modificações apenas em nível de código-fonte, o que exige conhecimento técnico de programação [Souza et al. 2011, Souza et al. 2012a].

Buscando garantir maior autonomia aos professores para a adaptação de OAs às suas necessidades, Souza et al. (2012b) propuseram a estratégia de Customização Guiada (CG), que visa possibilitar aos usuários executarem customizações de recursos em nível de interface, sem demandar conhecimento técnico. Os OAs do tipo animação/simulação baseados nesta estratégia são denominados Objetos de Aprendizagem Customizáveis (OACs) [Souza et al. 2012b].

Os OACs oferecem aos seus usuários mecanismos que permitem a fácil e rápida customização de suas interfaces. Eles são formados por Cenas constituídas de Componentes, que, por sua vez, podem ser do tipo: *texto*; *botão*; *imagem*; *áudio* e *vídeo* [Souza et al. 2011, Souza et al. 2012a]. Cada Componente possui, de acordo com o tipo, um conjunto de atributos cujos valores determinam o seu estado na Cena. Os OAC são customizados por meio da alteração dos valores desses atributos [Silva et al. 2017, Souza et al. 2012a]. Por exemplo, um botão pode ser habilitado/desabilitado; uma imagem substituída; um texto traduzido; dentre outros procedimentos.

Assim como os OAs em geral, para que sejam reutilizados, os OACs necessitam ser compartilhados por meio de ROAs [Silva et al. 2017]. Porém, como os *ROAs Tradicionais* funcionam basicamente como depósitos para o armazenamento e a obtenção de OAs, onde os recursos são persistidos em forma de pacotes que encapsulam todos os seus elementos, tais como executáveis e mídias (imagens, áudios, vídeos) [Santos et al. 2014, Tarouco et al. 2011]. Armazenar OACs e suas respectivas Versões Customizadas (VCs) nesses repositórios resultaria em inúmeras replicações na base, porque a criação de uma VC a partir da customização de um único atributo demandaria a persistência da nova versão como um pacote completo, incluindo arquivos e dados inalterados [Silva et al. 2017]. Neste contexto, Silva et al. (2017) propuseram o *Customizable Learning Object VErsioning Repository* (*CLOVeR*), um repositório que considera os aspectos estruturais e funcionais dos OACs e de suas VCs e realiza o devido Controle de Versão desses recursos, organizando e gerenciando as diversas versões e evitando a replicação de dados [Silva et al. 2017].

Este artigo descreve em detalhes a estratégia de Versionamento de Recursos proposta para o *CLOVeR* e apresenta os resultados obtidos ao longo de sua validação. Nela foi verificado o tempo de resposta para inclusão e obtenção de recursos com a estratégia proposta e analisada as situações em que o *CLOVeR* otimiza o armazenamento, já que não replica conteúdo comum entre versões de um mesmo OAC.

Na próxima seção são descritos os aspectos que as estratégias de Controle de Versão de Recursos em ROAs devem possuir e como elas são comumente implementadas. O artigo segue com uma seção em que o *CLOVeR* é apresentado e a sua estratégia de Versionamento é detalhada. A quarta seção é destinada à descrição da

metodologia e dos resultados da avaliação do Controle de Versão proposto. Na quinta e última seção, são apresentadas a conclusão da pesquisa e uma listagem de trabalhos futuros que podem aprimorar o *CLOVeR*.

2. Versionamento de Recursos em Repositórios de Objetos de Aprendizagem

Os ROAs em geral devem possuir soluções para o Controle de Versão dos recursos armazenados, dado que os usuários comumente necessitam gerar versões de OAs mais adequadas ao seu contexto de uso [McNaught 2007, Silva e Souza 2016, Tate e Hoshek 2009, Theilmann e Altenhofen 2003]. Além disso, o Versionamento destes recursos contribui para o reúso, uma vez que possibilita e estimula o compartilhamento de versões modificadas dos mesmos [Tate e Hoshek 2009].

Tate e Hoshek (2009) e Theilmann e Altenhofen (2003) listam requisitos para estratégias de Controle de Versão de OAs. Dentre eles, está o fato delas terem que estabelecer relacionamentos entre os recursos e suas respectivas versões; manter o histórico das alterações que originaram cada versão; e garantir, apesar dos relacionamentos, a independência entre as várias versões de um recurso, ou seja, a inclusão/remoção de uma versão não deve impactar as demais. Brooks et al. (2003) também destacam que soluções para o Versionamento de OAs devem armazenar a *Sintática* e a *Semântica* das ações de modificação. Eles definem *Sintática* como sendo as diferenças entre as versões em um formato adequado para o processamento por sistemas, normalmente chamada de *Delta*, enquanto que a *Semântica* representa estas mesmas diferenças, mas em um formato compreensível por humanos, assim como *Logs*.

Silva e Souza (2016) executaram uma *Revisão Sistemática da Literatura* a fim de identificar como tecnicamente é realizado o Controle de Versão de recursos em ROAs. O estudo concluiu que o Versionamento se dá por meio da alteração de padrões de metadados ou definição de modelos de dados próprios para o repositório. Ambas as formas visam possibilitar a definição de relacionamentos entre os recursos, registrando a identificação das relações e as diferenças entre as versões [Silva e Souza 2016].

3. Versionamento de Recursos no *CLOVeR*

A estratégia de Versionamento de recursos proposta para o *CLOVeR* organiza e gerencia os OACs e suas VCs com base em um Modelo de Dados definido para o repositório, método comumente adotado pelos ROAs, como evidenciado no estudo de Silva e Souza (2016). O modelo estabelecido para o *CLOVeR* armazena os elementos que compõem os OACs (executáveis, arquivos de mídia, estado dos Componentes e metadados) de modo fragmentado e relaciona as suas versões em um formato hierárquico.

Para cada VC no *CLOVeR*, apenas são persistidas as diferenças entre o estado dos seus Componentes e o estado deles na versão inicial do recurso [Silva et al. 2017]. Isso torna as VCs independentes entre si, requisito listado por Tate e Hoshek (2009) e Theilmann e Altenhofen (2003) para o Controle de Versão de OAs. Além disso, as relações existentes entre as versões garantem a manutenção do histórico de modificações realizadas no recurso, algo também elencado como exigência por estes autores. Ademais, no *CLOVeR* são mantidas a *Sintática* e a *Semântica* das alterações [Brooks et al. 2003] que originaram cada versão dos OACs.

O Controle de Versão proposto para o *CLOVeR* resulta na otimização do espaço para o armazenamento de recursos, pois evita a replicação de arquivos e dados inalterados entre as versões de um mesmo OAC. Nele são persistidos apenas o estado dos Componentes e as mídias modificadas durante as customizações [Silva et al. 2017].

3.1. Modelo de Dados do *CLOVeR*

Assim como em *ROAs Tradicionais*, o envio e a obtenção de OACs/VCs no *CLOVeR* ocorrem por meio de pacotes que encapsulam todos os seus elementos. Quando um usuário realiza a inclusão de um novo recurso na base, todas as informações contidas no Pacote de Implantação submetido são adicionadas ao repositório de forma fragmentada [Silva et al. 2017], simplificando a reutilização de OAs, conforme destacado por Junqueira e Lóscio (2014). As informações relacionadas aos recursos (metadados e estado dos Componentes) são persistidas em um banco de dados *MongoDB*, uma base não-relacional orientada a documentos [MongoDB Inc. 2014a]. Já os arquivos executáveis e de mídia são armazenados em um diretório no próprio sistema de arquivos do servidor em que o *CLOVeR* estiver hospedado, sendo referenciados pelas suas localizações (*paths*) [Silva et al. 2017].

Como detalhado por Silva et al. (2017), para cada novo OAC, três documentos são criados no *MongoDB*: um *RDescritor* com os metadados do OA; um *EFDescriptor* com os metadados do arquivo executável para cada executável que o recurso possuir; e um *CDescriptor* com o estado dos Componentes no executável para cada *EFDescriptor*. Já no caso da inclusão de uma nova Versão Customizada, um *VDescriptor* é criado para representar a versão na base. Neste documento são registradas as diferenças do estado dos Componentes da versão em relação ao estado inicial [Silva et al. 2017].

Para adicionar uma VC ao *CLOVeR*, o usuário deve, além de submeter um Pacote de Implantação de Versão Customizada (*Customized Version Deployment Package – CV-DP*), conforme descrito por Silva et al. (2017), informar um conjunto de metadados para descrever a nova versão. São exigidos os seguintes elementos da Categoria *General* do padrão *IEEE Learning Object Metadada* (*IEEE LOM*): *Título* (*Title*); *Idiomas* (*Languages*); e *Descrição* (*Description*). O campo *Descrição* destina-se à *Semântica* das alterações [Brooks et al. 2003] que originaram a versão. Os usuários descrevem nele a motivação e as modificações executadas no processo de customização.

Já em relação à *Sintática* [Brooks et al. 2003], no *CLOVeR* ela é representada pelas diferenças no estado dos Componentes do recurso. O *Delta* é definido no momento da inclusão da VC e é equivalente às diferenças detectadas na comparação do novo estado dos Componentes após a customização, informação contida no *CV-DP*, com o estado inicial dos Componentes do recurso, contido no *CDescriptor*. A listagem a seguir apresenta um exemplo de *Delta* persistido no *customizations* de um *VDescritor*:

```
"customizations": [
{ "scene": "initial_scene",
  "components": [
  { "source":"./file/hist_rias_quase_fant_sticas/swf/components/1/mrB_2.png",
    "name": "mrBook" },
  { "label": "Atividade 1",
    "name": "activity1Button" },
  { "enabled": false,
    "label": "Atividade 2",
    "name": " activity2Button" } ]
} ]
```

No caso deste exemplo, três Componentes da Cena denominada *initial_scene* têm o estado diferente de quando o OAC foi adicionado ao repositório. O Componente do tipo Imagem *mrBook* teve o seu arquivo de mídia alterado, enquanto que os Botões *activity1Button* e *activity2Button* tiveram as suas legendas (atributo *label*) traduzidas para o português, sendo que o *activity2Button* também foi desabilitado.

Como recomendado por Tichy (1985), o Controle de Versão do *CLOVeR* registra cada versão apenas como um bloco de modificações, não exigindo a replicação integral

do recurso. Dessa forma, a obtenção de versões se dá por meio da aplicação dessas alterações à versão original, ação denominada *Merge* [Tichy 1985]. No caso do *CLOVeR*, a recuperação de uma VC demanda a execução de um único *Merge*, que ocorre entre o estado dos Componentes na VC e o seu estado inicial. Esta estratégia foi inspirada no Git, difundido sistema de versionamento de arquivos, que, criando arquivos de referência, minimiza o tempo e a complexidade para se obter uma determinada versão, uma vez que evita a execução de múltiplos *Merges* [Chacon e Straub 2014].

O conjunto de atributos que cada usuário pode customizar em um OAC é limitado conforme o Grau de Liberdade (GL) do mesmo. GL é um grau de permissão atribuído de acordo com o nível de conhecimento e de interesse dos utilizadores. Ele busca garantir a manutenção, mesmo com as customizações, das relações pedagógicas incorporadas aos recursos no momento da concepção [Souza et al. 2012b]. Durante a inclusão de uma nova VC ao *CLOVeR*, o GL do usuário é validado, sendo verificado se ele possui permissão para adicionar uma versão originada a partir das respectivas modificações. No caso de uma *Versão de 1º Nível* (criada a partir do OAC original), os atributos alterados dos Componentes (*Delta*) são analisados. Já no caso das *Versões de 2º Nível* em diante (geradas a partir de outras versões), primeiro é necessário realizar o *Merge* do estado dos Componentes da versão de origem com o estado inicial deles, para, em seguida, calcular o *Delta* da nova versão em relação a sua versão de origem. Desse modo se obtêm quais alterações foram de fato realizadas no processo de customização, possibilitando a validação do Grau de Liberdade do usuário.

Cada Versão Customizada no *CLOVeR* possui um *Número de Versão*, que é baseado na posição da VC na hierarquia de versões de um OAC. Ele é registrado no campo *version* do *VDescriptor*. Os novos arquivos de mídia definidos para as versões durante os processos de customização são armazenados juntamente com os executáveis e as mídias originais do OAC, mas em uma estrutura de subdiretórios que reserva um local para os novos arquivos de cada VC conforme o seu *Número de Versão* (Figura 1). Com isso, o executável e as mídias são persistidas uma única vez na mesma hierarquia.

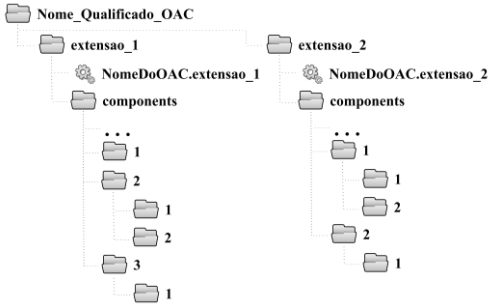


Figura 1. Exemplo da estrutura de diretórios destinada a Executáveis e Mídias

Os relacionamentos existentes entre os vários documentos criados no *MongoDB* resultam em uma *Hierarquia de Versionamento* em formato de árvore para cada OAC persistido no *CLOVeR*, conforme ilustrado na Figura 2 [Silva et al. 2017]. Apesar de ser uma base de dados não-relacional, o *MongoDB* dá suporte às estruturas de dados do tipo árvore, propiciando a indexação das referências estabelecidas nelas [MongoDB Inc. 2014b]. Essa organização em forma de hierarquia simplifica a identificação da versão de origem e de versões derivadas no grupo de VCs de um OAC.

Mesmo relacionando-se entre si, cada VC é independente das demais, requisito para o Versionamento de OAs [Tate e Hoshek 2009, Theilmann e Altenhofen 2003], já que apenas são registradas as diferenças entre o estado dos Componentes da versão e o estado inicial. Dessa forma, elas podem ser acessadas e customizadas por múltiplos

usuários em paralelo, cabendo ao *CLOVeR* determinar em que posição na *Hierarquia de Versionamento* cada nova versão deve ser adicionada.

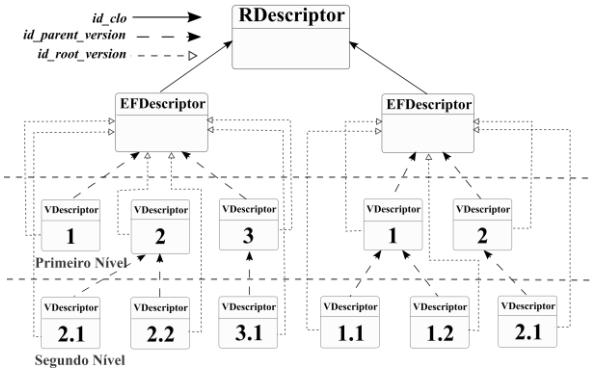


Figura 2. Exemplo da *Hierarquia de Versionamento* de um OAC

3.2. *Customizable Learning Object Web Platform*

O *CLOVeR* pode ser acessado e gerenciado por meio de uma plataforma *web responsiva* que disponibiliza publicamente os OACs e suas Versões Customizadas, a *Customizable Learning Object Web Platform* (*CLOWebPlatform*) [Silva et al. 2017]. Esta plataforma também possui uma interface *RESTful* para a inclusão e a obtenção de recursos por meio de requisições *HTTP*, permitindo a integração do repositório com outras aplicações, tais como Ambientes Virtuais de Aprendizagem [Silva et al. 2017].

Quando um usuário deseja baixar a versão inicial de um OAC, o *back-end* da *CLOWebPlatform* busca na base o *EFDescritor* referente ao recurso requisitado e, a partir desse documento, obtém o estado dos Componentes para o executável em questão, informação contida no *CDescriptor* referenciado. Todas estas informações são empacotadas no formato de um Pacote de OAC (*Customizable Learning Object Package – CLO-P*), descrito por Silva et al. (2017), e enviadas ao solicitante.

Já no caso do *download* de uma VC, primeiramente o *back-end* da plataforma encontra o *VDescritor* que a representa no *CLOVeR*, para em seguida obter o *EFDescriptor* referenciado por tal documento. O *CDescriptor* com o estado inicial dos Componentes é obtido a partir do *EFDescriptor* e é executado um *Merge* do seu conteúdo com o valor registrado no campo *customization* do *VDescritor*. O estado dos Componentes resultante do *Merge* é empacotado, juntamente com as demais informações referentes à VC, em forma de um *CLO-P* e enviado ao usuário requisitante.

4. Avaliação do Versionamento de Recursos no *CLOVeR*

O *CLOVeR* foi submetido a um processo de avaliação que buscou verificar o tempo de atendimento a operações de inclusão e obtenção de recursos, dado que estas ações exigem a execução de processamentos, devido à estratégia de Versionamento armazenar os recursos de forma fragmentada. A avaliação também buscou averiguar em quais cenários o repositório otimiza o consumo de espaço utilizado para a persistência de recursos, já que não replica conteúdo inalterado entre as versões de um mesmo OAC.

Todas as *Rotinas de Teste* descritas a seguir foram executadas com o *CLOVeR* e a *CLOWebPlatform* hospedados em um servidor *Windows 10* (*64 bits*), utilizando o sistema de arquivos *NTFS*, com 6 GB de memória *RAM* e processador *Intel Core i7-3537U* de 4 núcleos e 2,00 GHz de frequência. Todas as operações (inclusão/obtenção) foram realizadas por meio da interface *RESTful* da *CLOWebPlatform*. É importante destacar também que os tempos aferidos dizem respeito ao efetivo tempo de

processamento, sendo desconsiderado o tempo de envio das requisições e das respostas, pois dependem da taxa de transmissão da rede utilizada pelos usuários.

4.1. Comportamento em Relação ao Tempo

Primeiramente, foram cronometrados os tempos para processar requisições de inclusão e obtenção de recursos. Para tanto, foram definidos dois *Cenários de Teste*:

(i) ***Variando a Quantidade de Arquivos de Mídia*** – Cada OAC com 100 Componentes, variando de 10 a 100 (10 em 10) a quantidade dos que referenciam arquivo de mídia e têm esse arquivo substituído durante a customização;

(ii) ***Variando a Quantidade de Componentes*** – Cada OAC com 10 Componentes referenciando arquivo de mídia, variando de 10 a 100 (10 em 10) a quantidade total de Componentes e dos que são customizados, fixando-se em 10 as mídias substituídas.

O primeiro *Cenário de Teste* objetivou analisar o impacto no tempo total de execução da variação do número de mídias contidas no OAC e substituídas para cada nova VC, enquanto que o segundo buscou avaliar o impacto da mudança do número total de Componentes. Em ambos os cenários, os OACs foram padronizados em: 5 Cenas; 1 Arquivo Executável (5 MB); e todas as Mídias com 1 MB. Cada *Rotina de Teste* consistiu na adição de 1 OAC e de 4 VCs (do 1º ao 4º nível) ao *CLOVeR*.

Os resultados demonstraram que, no caso das inclusões, o tempo para adicionar executáveis e mídias ao *filesystem* do servidor de hospedagem do *CLOVeR* é o maior gargalo do processamento. O gráfico apresentado na Figura 3 demonstra a representatividade desse tempo em relação ao tempo total de processamento no primeiro cenário, no qual se variou a quantidade de mídias. É possível notar, inclusive, que essa representatividade se eleva juntamente com o número de arquivos de mídia nos Pacotes de Implantação de OAC/VC, aproximando-se de 100% na maioria dos casos.

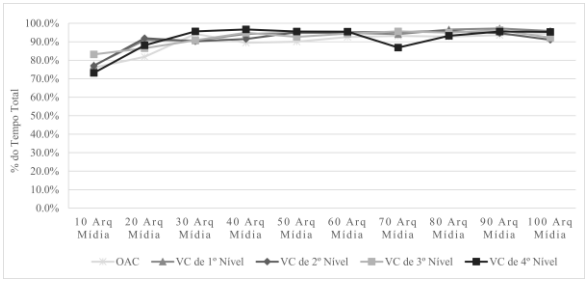


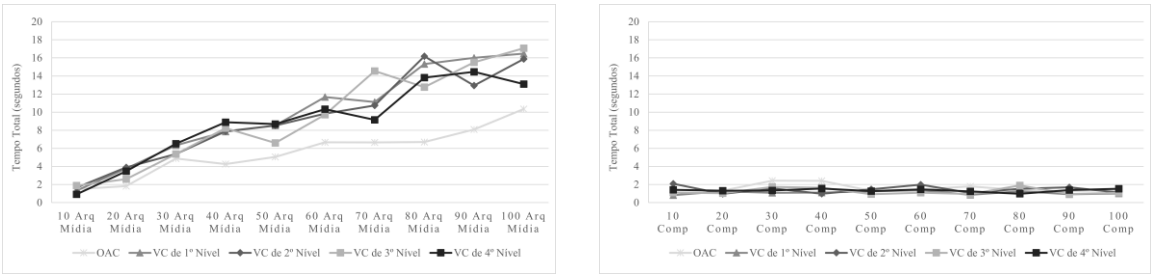
Figura 3. Representatividade do tempo de escrita de arquivos em *filesystem* em relação ao tempo total de inclusão do recurso

Em relação à inclusão de Versões Customizadas, os tempos do *Cálculo do Número de Versão* e do *Cálculo do Delta* para a versão a ser inserida também foram registrados, uma vez que são execuções importantes para a estratégia de Versionamento proposta. Os resultados indicaram que eles têm impacto mínimo. Desconsiderando o tempo de escrita das mídias, eles representam, somados, menos de 5% do tempo total.

Os gráficos da Figura 4 demonstram a evolução do tempo para a inclusão de recursos nos dois *Cenários de Teste*. Os resultados apresentados evidenciam que esse tempo varia de forma linear com o quantitativo de arquivos escritos em *filesystem* (Figura 4.a), mantendo-se constante quando este número não se altera, independentemente da posição do recurso na *Hierarquia de Versionamento* (Figura 4.b).

Como parte da avaliação, foi realizado o cálculo do tempo médio para a adição de uma mídia com 1 MB no sistema de arquivos. Para o ambiente utilizado, a média foi de *0,137 segundos/mídia*. Portanto, é possível estimar que, no ambiente da avaliação, a

inclusão de uma Versão Customizada com 1.000 mídias de 1 MB (algo em torno 1 GB) ocorre em aproximadamente 2 minutos e 17 segundos.



(a) (b)

Figura 4. Tempo total de inclusão: (a) *Primeiro Cenário* (b) *Segundo Cenário*

Em relação às aferições dos tempos de resposta às requisições para a obtenção de OAC/VC, os resultados obtidos demonstraram que, apesar do número de Componentes e arquivos de mídia no recurso e de suas posições na *Hierarquia de Versionamento*, esse tempo se mantém na faixa de *0,05 a 0,60 segundos*. O fato dos executáveis e das mídias serem lidos do *filesystem* apenas no momento do *download* do Pacote de OAC e uma única operação de *Merge* ser demandada para a definição do estado dos Componentes para cada Versão Customizada contribuem para a manutenção do tempo nessa faixa.

Os resultados indicam que, mesmo com os processamentos exigidos pela estratégia de Versionamento, o *CLOVeR* possui adequado tempo de resposta ao atender requisições para a inclusão e a obtenção de recursos. E, no caso das inclusões, o tempo total varia de forma linear com características que impactam o tamanho dos Pacotes de Implantação (quanto mais e maiores as mídias = maior tempo de inclusão), algo que tende a ser naturalmente aceito pelos seus usuários.

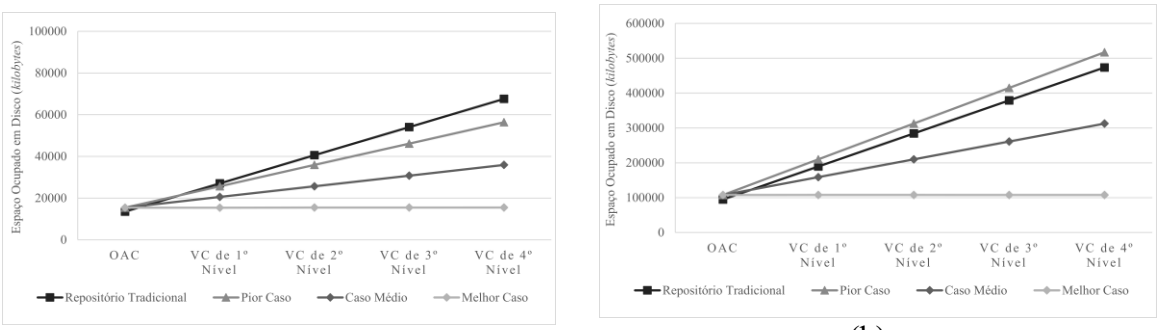
4.2. Utilização de Recursos

Considerando que o *CLOVeR* otimiza o espaço para o armazenamento de recursos, já que evita a replicação de arquivos e dados inalterados entre as versões de um mesmo OAC [Silva et al. 2017], no segundo momento da avaliação, a fim de verificar se a otimização ocorre de forma semelhante em cenários de recursos com poucas e com muitas mídias, foram realizadas as medições do espaço ocupado (em *kilobytes*) pelos recursos no repositório em dois *Cenários de Teste*: (1º) 100 Componentes, sendo 10 deles com referência a arquivo de mídia; (2º) 101 Componentes, sendo 100 deles com referência a arquivo de mídia. Cada cenário foi subdividido em três *Rotinas de Teste*:

(i) **"*Pior Caso*"** – Cada VC oriunda da troca de todas as mídias do recurso;
(ii) **"*Caso Médio*"** – Cada VC oriunda da troca de metade das mídias do recurso;
(iii) **"*Melhor Caso*"** – Cada VC sem nenhuma mídia do recurso alterada.

Assim como para a análise do *Comportamento em Relação ao Tempo*, os OACs de ambos os *Cenários de Teste* foram padronizados em: 5 Cenas; 1 Arquivo Executável (5 MB); e todas as Mídias com 1 MB. E cada rotina consistiu na inclusão de 1 OAC e de 4 diferentes VCs. As medições foram realizadas somando-se o espaço usado pelo *MongoDB* (comando `db.stats().dataSize` no *shell* do banco) com o espaço ocupado pelos executáveis e pelas mídias adicionadas ao sistema de arquivos.

Na Figura 5 são apresentados os gráficos com os resultados obtidos nos dois cenários, juntamente com a projeção de espaço ocupado se um *ROA Tradicional* fosse utilizado. Foi possível realizar esta projeção por se conhecer o tamanho dos Pacotes de OAC: 13.516 KB no *Primeiro Cenário* e 94.708 KB no *Segundo Cenário*. Um *ROA Tradicional* demandaria um novo *CLO-P* de igual tamanho para cada VC incluída.



(a) (b)

Figura 5. Espaço para o armazenamento de recursos no *CLOVeR* e em um *ROA Tradicional*: (a) *Primeiro Cenário* (b) *Segundo Cenário*

No *"Pior Caso"* dos dois cenários, o repositório proposto possui a mesma tendência de crescimento de um *ROA Tradicional*, mas com um consumo ligeiramente menor no *Primeiro Cenário*, pois, como os recursos possuem poucas mídias, a não replicação do executável de 5 MB pelo *CLOVeR* influencia consideravelmente a evolução do consumo. Ele é superficialmente superior no *Segundo Cenário*, porque os *CLO-Ps* são persistidos de forma compactada nos *ROAs Tradicionais*, enquanto que executáveis e mídias são armazenadas diretamente no *filesystem* do *CLOVeR*, sem nenhuma estratégia de compactação. Isto impacta significativamente o consumo em cenários de recursos com muitas mídias. Já nas rotinas de *"Caso Médio"* e *"Melhor Caso"*, o *CLOVeR* possui expressiva economia de espaço em comparação com um *ROA Tradicional*, já que não replica arquivos comuns entre as diversas versões de um OAC.

Os resultados obtidos também evidenciaram que são os arquivos executáveis e as mídias dos recursos que determinam prioritariamente o espaço necessário para o seu armazenamento, uma vez que adicionar *VDescriptors* ao *MongoDB* afeta minimamente o consumo de espaço em disco. Cada nova versão nas rotinas de *"Melhor Caso"* (nenhuma mídia modificada) resultou em um consumo adicional de apenas 16 KB.

5. Conclusão

Neste artigo foi descrita a estratégia de Versionamento de recursos proposta para o *CLOVeR*, sendo detalhado como os elementos que compõem os OACs e suas respectivas VCs são organizados no repositório e quais os processamentos efetuados para a inclusão e a obtenção desses recursos, dado que eles são persistidos de forma fragmentada. A validação realizada demonstrou que o tempo de resposta do *CLOVeR* para inclusão de recursos varia linearmente com a quantidade e o tamanho dos executáveis e das mídias definidos nos Pacotes de Implantação. No caso da obtenção, o tempo se mantém dentro de uma determinada faixa, independentemente das características do recurso e de sua posição na *Hierarquia de Versionamento*.

Os resultados da validação também evidenciaram, em complemento ao estudo de Silva et al. (2017), que a otimização do espaço consumido pelo *CLOVeR* para o armazenamento de recursos ocorre de forma semelhante tanto em cenários com poucos quanto em cenários com muitos arquivos de mídia nos recursos. Eles também revelaram que são os arquivos adicionados ao sistema de arquivos (executáveis e mídias) o que de fato define a quantidade de espaço necessário para o armazenamento.

Contudo, algumas ações futuras podem ser realizadas para o aprimoramento do repositório, tais como o compartilhamento de arquivos de mídia entre OACs diferentes, otimizando ainda mais o armazenamento, e a pública disponibilização de todas as mídias persistidas, para que possam ser reutilizadas. Além disso, fica definida a

necessidade de estudo futuro para mitigar o gargalo gerado pelo tempo de escrita em *filesystem*. Uma possibilidade é a realização da escrita de forma assíncrona, após o envio da resposta ao usuário que incluiu o recurso no repositório.

Referências

Cechinel, C. e Ochoa, X. (2014). A Brief Overview of Quality inside Learning Object Repositories. In: *Proc. of 15th International Conference on Human-Computer Interaction (INTERACCIÓN)*, Puerto de la Cruz, p. 83-89.

Chacon, S. e Straub, B. (2014). Git Internals. In: *Pro Git: everything you need to know about Git*, 2. ed., Apress, Mountain View, p. 481-523.

Brooks, C., Cooke, J. e Vassileva, J. (2003). Versioning of Learning Objects. In: *Proc. of 3rd IEEE International Conference on Advanced Learning Technologies (ICALT)*, Athens.

Ieiri, A. Y. e Braga, J. C. (2015). Problemas de Usabilidade em Repositórios de Objetos de Aprendizagem a partir de Estudos Primários. In: *Anais do 26º Simpósio Brasileiro de Informática na Educação (SBIE)*, Maceió, p. 732-741.

Junqueira, R. P. e Lóscio, B. F. (2014). Repositórios de Objetos de Aprendizagem: uma análise comparativa com ênfase no reúso de conteúdos. In: *Anais do 25º Simpósio Brasileiro de Informática na Educação (SBIE)*, Dourados, p. 988-992.

McNaught, C. (2007). Developing Criteria for Successful Learning Repositories. In: *Web Information Systems and Technologies*, Edited by Filipe, J., Cordeiro, J. e Pedrosa, V., Springer, Berlim, p.8-18.

MongoDB Inc. (2014a). Introduction to MongoDB. Disponível em: < https://docs.mongodb.com/v3.2 /introduction/ >. Acesso: 05 jun. 2017.

MongoDB Inc. (2014b). Model Tree Structures in MongoDB. Disponível em: < https://docs.mongodb .com/v3.2/tutorial/model-tree-structures/ >. Acesso: 05 jun. 2017.

Santos, H. L., Carrillo, G., Cechinel, C. e Ochoa, X. (2014). Towards the use of Semantic Learning Object Repositories: evaluating queries performance in two different RDF Implementations. In: *Bulletin of the IEEE Technical Committee on Learning Technology*, v. 16, n. 4, p. 6-9.

Sinclair, J., Joy, M., Yau, J. Y. e Hagan, S. (2013). A Practice-Oriented Review of Learning Objects. In: *IEEE Transactions on Learning Technologies*, v. 6, n. 2, p. 177-192.

Silva, J. W. F. e Souza, C. T. (2016). Versionamento de Recursos em Repositórios de Objetos de Aprendizagem: uma revisão sistemática. In: *Novas Tecnologias na Educação*, v. 14, n. 2, p. 1-11.

Silva, J. W. F., Souza, C. T. e Souza, M. F. C. (2017). CLOVeR: an optimized repository for Customizable Learning Objects. In: *Proc. of 17th IEEE International Conference on Advanced Learning Technologies (ICALT)*, Timisoara, p. 76-78.

Souza, M. F. C., Castro-Filho, J. A. e Andrade, R. M. C. (2011). Applying Model-Driven Development for Building Customizable Learning Objects. In: *IEEE Multidisciplinary Engineering Education Magazine*, v. 6, n. 1, p. 22-30.

Souza, M. F. C., Castro-Filho, J. A. e Andrade, R. M. C. (2012a). Ampliando a Autonomia Docente com o Uso de Objetos de Aprendizagem Customizáveis. In: *Workshops do Congresso Brasileiro de Informática na Educação (CBIE)*, Rio de Janeiro.

Souza, M. F. C., Castro-Filho, J. A. e Andrade, R. M. C. (2012b). Customização Guiada: uma estratégia Orientada a Modelos para a produção de Objetos de Aprendizagem. In: *Anais do 23º Simpósio Brasileiro de Informática na Educação (SBIE)*, Rio de Janeiro.

Tarouco, L., Silva, C. e Grando, A. (2011). Fatores que Afetam o Reúso de Objetos de Aprendizagem. In: *Novas Tecnologias na Educação*, v. 9, n. 1, p. 1-10.

Tate, M. e Hoshek, D. (2009). A Model for the Effective Management of Re-Usable Learning Objects (RLOs): lessons from a Case Study. In: *Interdisciplinary Journal of E-Learning and Learning Objects*, v. 5, p. 51-73.

Theilmann, W. e Altenhofen, M. (2003). Versioning of E-learning Objects Enabling Flexible Reuse. In: *Proc. of IADIS International Conference WWW/Internet*, Algarve, p. 719-727.

Tichy, W. F. (1985). RCS - a system for version control. In: *Software: Practice & Experience*, v. 15, n. 7, p. 637-654.

Tzikopoulos, A., Manouselis, N. e Vuorikari, R. (2009). An Overview of Learning Object Repositories. In: *Selected Readings on Database Technologies and Applications*, Edited by Terry, H., IGI Global, London, p. 44-64.

Wiley, D. A. (2000). Learning Object Design and Sequencing Theory. Tese (Doutorado em Filosofia) - Department of Instructional Psychology and Technology, Brigham Young University, Provo.