Um Estudo de Mapeamento Sistemático Sobre o Padrão OSLC

Bruno Marcelo S. Ferreira[1], Rafael S. Torres[1], Fábio P. Basso[1], Diego Kreutz[1], Elder Rodrigues[1], Maicon Bernardino[1], Rafael Z. Frantz[2]

[1]Universidade Federal do Pampa (UNIPAMPA), PPGES
Av. Tiarajú, 810 - Bairro Ibirapuitã - Alegrete, RS

[2]Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ)
Rua do Comércio, 3000 - Universitário, Ijuí - RS

{brunoferreira.aluno, rafaeltorres.aluno, fabiobasso,
diegokreutz, elderrodrigues}@unipampa.edu.br;
bernardino@acm.org; rzfrantz@unijui.edu.br

Abstract. *The software industry invests in modern tools throughout the software development lifecycle. However, there are challenges to achieve an end-to-end integrated environment such as data integration and artifact traceability. To mitigate these challenges, many approaches have been proposed for integration. In this context, Open Services for Lifecycle Collaboration (OSLC) is an open standard for tool interoperability, which allows data federation throughout Software Engineering (SE) application lifecycles. This study presents a systematic mapping study about OSLC, analyzing 59 primary studies and addressing integration issues.*

Resumo. *A indústria de software investe em ferramentas modernas ao longo de todo o ciclo de desenvolvimento de software. No entanto, existem desafios para alcançar um ambiente integrado de ponta a ponta, como por exemplo estabelecer a rastreabilidade dos artefatos. Para mitigar esses desafios, diversas abordagens foram propostas para integração de ferramentas de software. Nesse contexto, o Open Services for Lifecycle Collaboration (OSLC) é um padrão aberto para interoperabilidade de ferramentas, que permite a federação de dados ao longo do ciclo de vida de aplicações de Engenharia de Software (ES). Este artigo apresenta um estudo de mapeamento sistemático sobre OSLC, analisando 59 estudos primários e abordando questões de integração.*

1. Introdução

Ferramentas de software são usadas para apoiar times a concluir atividades ao longo do ciclo de vida de desenvolvimento de software. No entanto, essas ferramentas geralmente são projetadas sem o suporte necessário para serem integradas a outras ferramentas de software. Além disso, possuem configurações, funcionalidades, fabricantes e manipulam diversos tipos de artefatos. Esse cenário adiciona desafios para a obtenção de um ambiente integrado de ponta a ponta, como por exemplo estabelecer e manter relacionamentos entre artefatos através de toda cadeia de ferramentas.

Ambientes integrados de forma automatizada são um fenômeno raro na indústria devido ao número de características envolvidas no processo de interoperabilidade de ferramentas [Wicks and Dewar 2007]. Além disso, cenários de integração de ferramentas

de software (por exemplo, projetos de larga escala) geralmente adicionam desafios extras para alcançar um ambiente integrado de ponta a ponta, como rastreabilidade entre artefatos de ferramentas, caracterizando um problema de interoperabilidade complexo.

A solução industrial Open Services for Lifecycle Collaboration (OSLC) [OSLC 2020] define um modelo de representação comum para artefatos produzidos ao longo do projeto, potencializando a integração e interoperabilidade de ferramentas. A especificação do OSLC consiste em regras e padrões para o uso de tecnologias, padrões como o RESTful e protocolos como o HTTP. Há domínios de aplicação que estendem a especificação OSLC através de regras específicas de contexto, como em Gerenciamento de Requisitos, Gerenciamento de Qualidade, Gerenciamento de Configuração e Mudanças. Como resultado, a especificação do OSLC potencializa a interoperabilidade em diferentes domínios de desenvolvimento de software sem requerer o conhecimento específico sobre como as ferramentas operam internamente.

O OSLC é baseado nos princípios de Dados Ligados e no funcionamento da *web*. Além disso, cada artefato é representado por arquivos RDF/XML estruturados com base em domínios OSLC. Os relacionamentos entre os artefatos são obtidos através de uma estrutura composta por três elementos: sujeito, predicado e objeto. Por exemplo, em um ambiente formado pelas ferramentas de gerenciamento de requisitos (Ferramenta A) e gerenciamento de testes (Ferramenta B), um requisito na Ferramenta A pode ser validado por um caso de teste na Ferramenta B por meio do OSLC.

Em uma cadeia de ferramentas OSLC, uma ferramenta pode ser classificada como um provedor ou consumidor. As ferramentas provedoras são projetados para armazenar e prover dados, permitindo as ferramentas consumidoras fácil acesso para navegar, criar e recuperar dados [Aichernig et al. 2014]. Existem três abordagens para prover suporte OSLC em ferramentas de software: abordagem nativa, *plugin* e adaptador. A abordagem mais recomendada é o desenvolvimento de adaptadores, pois não é preciso possuir conhecimento das tecnologias que envolvem a implementação da ferramenta.

O desenvolvimento de interfaces adaptadoras OSLC são relatadas na literatura como a alternativa mais utilizada e recomendada para integrar ferramentas com OSLC. Entretanto, ainda existem alguns desafios como reduzir o custo elevado de desenvolvimento para que esse objetivo torne-se viável. O processo para o desenvolvimento dos adaptadores OSLC é realizado de forma semi-automática, sendo necessário a implementação de código-fonte que implica em esforço manual para recuperar e manipular os dados mantidos pelas ferramentas de software.

Neste trabalho apresentamos um Estudo de Mapeamento Sistemático (SMS) que analisa 59 estudos primários relatando o uso do OSLC e discutimos a análise dos resultados destes estudos focados nos desafios encontrados sobre o desenvolvimento de adaptadores OSLC na indústria. Os resultados mostram que se for possível reduzir o trabalho para desenvolver ou reutilizar os adaptadores será possível explorar o OSLC em outros ambientes além dos sistemas críticos de segurança.

O restante do artigo está estruturado como segue. O protocolo do estudo empírico e as análises e resultados são apresentados nas Seções 2 e 3, respectivamente. Na Seção 4, apresentamos e discutimos as ameaças à validade enquanto. A partir das análises e resultados, resumimos as principais na Seção 5. Por fim, apresentamos as considerações

finais e trabalhos futuros na Seção 6

2. Estudo de Mapeamento Sistemático

Nesta seção descrevemos o processo aplicado em nosso estudo de mapeamento sistemático. Para realizar o SMS, seguimos o protocolo proposto por Petersen *et al.* [Petersen et al. 2015]. O protocolo completo deste trabalho está disponível no Google Drive[1].

2.1. Questões de Pesquisa

Definimos as seguintes Questões de Pesquisa (QP) para o estudo de mapeamento sistemático:

QP1. *Em quais fases do ciclo de vida do software o trabalho utiliza o padrão OSLC como solução?* Os estudos de integração de aplicações voltados para contextos de ES são normalmente associados a determinados ciclos de vida da aplicação. Assim, buscamos caracterizar os estudos que contribuem para determinadas disciplinas de ES, fornecendo um mapa de cobertura que permite inferir se os processos de desenvolvimento de software são automatizados.

QP2. *Quais são as facetas da contribuição do estudo?* Nosso objetivo é caracterizar as contribuições da área de acordo com sua faceta: seja com ferramentas, modelagem, processos e metodologias.

QP3. *Quais os tipos de pesquisa mais frequentes sobre OSLC?* Por fim, para compreender a maturidade dos estudos do ponto de vista empírico, também buscamos classificar os estudos de acordo com o tipo de avaliação adotada.

2.2. *Strings* de Busca

As buscas pelos trabalhos relacionados ao SMS foram realizadas em cinco bibliotecas digitais frequentemente utilizadas pela comunidade científica, incluindo ACM Digital Library[2], IEEEXplore[3], Springer Link[4], Science Direct[5] e Scopus[6]. As *strings* de busca foram elaboradas com o objetivo de obter uma ampla cobertura de estudos sobre o padrão OSLC, como mostra a Tabela 1.

2.3. Critérios de Inclusão e Exclusão

Os critérios de inclusão (CI) e critérios de exclusão (CE) são definidos e utilizados para filtrar os artigos relacionados ao padrão OSLC. Os CI e CE foram definidos e aplicados com base na leitura do título, resumo e palavras-chave. Os artigos devem satisfazer todos os CIs para serem selecionados. Porém, se o artigo estiver associado a pelo menos um CE, é excluído do SMS. Os critérios são os seguintes:

[1]Google Drive: `https://drive.google.com/drive/folders/14Fus_yehjlRxRXP53_V1ATxgzsdfo7Bd?usp=sharing`

[2]ACM DL: `https://dl.acm.org/`

[3]IEEEXplore: `https://ieeexplore.ieee.org/`

[4]Springer: `https://link.springer.com/`

[5]Science Direct: `https://www.sciencedirect.com/`

[6]Scopus: `https://www.scopus.com/`

Tabela 1. *Strings* de Busca

Base de Dados	*String* de Busca
ACM DL	(“Open Services for Lifecycle Collaboration” OSLC) AND (Integration Interoperability “Linked Data” Lifecycle Traceability)
IEEE Xplore	(“Open Services for Lifecycle Collaboration” OR OSLC) AND (Integration OR Interoperability OR “Linked Data” OR Lifecycle OR Traceability)
Springer Link	(“Open Services for Lifecycle Collaboration” OR OSLC) AND (Integration OR Interoperability OR “Linked Data” OR Lifecycle OR Traceability)
Science Direct	(“Open Services for Lifecycle Collaboration” OR OSLC) AND (Integration OR Interoperability OR “Linked Data” OR Lifecycle OR Traceability)
Scopus	TITLE-ABS-KEY ((“Open Services for Lifecycle Collaboration” OR OSLC) AND (Integration OR Interoperability OR “Linked Data” OR Lifecycle OR Traceability))

- **CI1:** O estudo apresenta no título, resumo, palavras-chave ou na introdução os termos “OSLC” ou “Open Services for Lifecycle Collaboration”.
- **CI2:** O estudo fornece um exemplo de uso, método, ferramenta, métrica ou processo que emprega o padrão OSLC como solução.
- **CE1:** O estudo não é escrito em inglês.
- **CE2:** O estudo não é primário.
- **CE3:** O estudo contém menos de quatro páginas.
- **CE4:** O estudo não é da área da Ciência da Computação.
- **CE5:** O estudo foi publicado anterior à 2008.
- **CE6:** O estudo não está disponível para download.
- **CE7:** O estudo é duplicado.

2.4. Extração de Dados

Para responder às questões de pesquisa e analisar adequadamente os resultados, utilizamos o formulário de extração de dados, conforme Tabela 2. Este formulário contém dez itens que devem ser identificados pelos pesquisadores nos artigos, após uma leitura completa do texto. Outra informação que é resultante dessas leituras são as respostas às questões de pesquisa, apresentadas na Seção 2.1.

Tabela 2. Formulário de Extração de Dados

Item	Questão de Pesquisa
Id	-
Título	-
Fase do ciclo de vida do software	QP1
Ferramentas	QP1
Processo de desenvolvimento de software	QP1
Vantagens	QP1
Desvantagens	QP1
Contexto da organização	QP1
Faceta da contribuição	QP2
Tipo de pesquisa	QP3

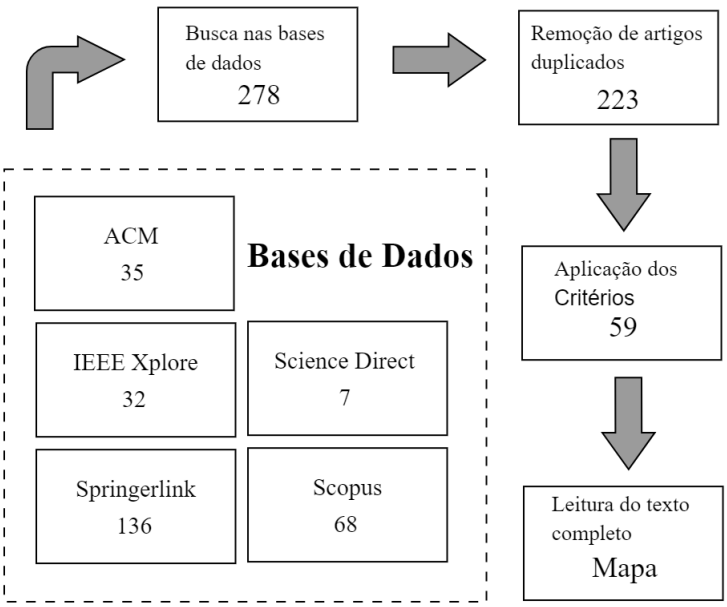

Figura 1. Processo de Triagem dos Estudos

3. Análise dos Resultados

Os resultados são baseados nos artigos selecionados durante o processo de triagem, conforme ilustrado na Figura 1. Na primeira etapa, as*strings* de busca foram utilizadas nas bases de dados e foram retornados 278 artigos. Em seguida, os estudos duplicados entre as bases de dados foram desconsiderados, restando 223 artigos. Assim, após aplicar os critérios de inclusão e exclusão com base no título, resumo, palavras-chave e introdução desses artigos, foram considerados 59 estudos de acordo com o tema de pesquisa.

3.1. QP1. Em quais fases do ciclo de vida do software o padrão OSLC é utilizado?

Para responder a QP1, mapeamos as fases do ciclo de vida do software em que a solução do estudo foi empregada. Para isso usamos as disciplinas do *Rational Unified Process* (RUP). De acordo com os resultados do SMS, a maioria das soluções propostas são para os domínios de Gerenciamento de Requisitos, Análise & Projeto, Gerenciamento de Configuração e Mudanças e Testes, como mostra a Tabela 3. Todas as ferramentas no conjunto de ferramentas foram consideradas parte da solução OSLC. Portanto, os estudos podem ser categorizados em mais de uma disciplina.

As abordagens OSLC se empenham em resolver problemas comuns entre os artigos. No domínio de requisitos, o OSLC é proposto para estabelecer a rastreabilidade entre os requisitos e o restante do projeto. Além disso, as ferramentas de requisitos de software são frequentemente usadas nos exemplos de conjuntos de ferramentas como um ponto de partida para o fluxo de trabalho.

A partir de conceitos de Dados Ligados, artigos como [VanZandt 2015], [Buffoni et al. 2017] apresentam abordagens para rastrear o histórico das mudanças por meio das propriedades do OSLC, incluindo as partes interessadas que o modificaram e tarefas nas quais os artefatos estão associados.

Tabela 3. Estudos Distribuídos pelas Disciplinas do RUP

Disciplina	Quantidade de Artigos
Modelagem de Negócios	1
Requisitos	21
Análise & Projeto	29
Implementação	6
Teste	25
Implantação	0
Gerenciamento de Configuração e Mudanças	14
Gerenciamento de Projetos	6
Ambiente	2
N/A	11

OSLC é uma alternativa para ambientes que precisam implementar rastreabilidade. Rastreabilidade é a capacidade de estabelecer links (ou rastros) entre artefatos de origem e artefatos de destino [Gotel and Mäder 2012]. A capacidade de implementar rastreabilidade em todos os ambientes está em linha com as motivações do ALM. Um dos problemas para atingir esse objetivo é a dificuldade em manter a consistência dos dados. OSLC pode ser uma alternativa para automatizar ambientes ALM devido à sua capacidade de manter esses relacionamentos em uma estrutura chamada tripla, incluindo não apenas os artefatos, mas também as partes interessadas e atividades relacionadas.

Os resultados do SMS reforçam a necessidade de integrar diferentes domínios de desenvolvimento de software desde suas fases iniciais, principalmente na disciplinas de Requisitos. Portanto, podemos citar ferramentas de Engenharia Orientada a Modelos (MDE), que são citadas na fase de Análise & Projeto como soluções para minimizar esses esforços de integração. Nesse contexto, o Eclipse Lyo [El-khoury 2016] propõe um gerador de código-fonte aberto para interfaces de ferramentas em conformidade com OSLC no ambiente eclipse, que foi citado por artigos que foram usados no desenvolvimento de adaptadores OSLC. Ainda, há trabalhos que propõem abordagens para apoiar a integração ou transformação de modelos heterogêneos como BPMN, UML, SysML, SoaML e EMF [d. Martino et al. 2016], [Lu et al. 2018], [Arnould 2018], [Mustafa and Labiche 2017], [Zhang and Møller-Pedersen 2013], [Biehl et al. 2012], [Biehl et al. 2012].

Outra observação é o uso de soluções aplicadas à disciplina de Teste devido aos contextos em que o OSLC é normalmente aplicado. Por se tratarem de sistemas críticos de segurança, os conjuntos de ferramentas são compostos por ferramentas de V&V. A maioria dos estudos (74.57%) encontrados são contextualizados em ambientes de sistemas críticos, como sistemas médicos, sistemas embarcados, indústria automotiva e indústria aeronáutica. Esses sistemas são caracterizados por conterem atividades de verificação e validação, como por exemplo simulações e diferentes níveis de testes. Em outros 12 trabalhos (20,33%), o contexto do ambiente não é citado pelo autor. A distribuição dos estudos pelos contextos é apresentado na Tabela 4.

3.2. QP2. Quais são as facetas da contribuição do estudo?

Identificar quais facetas de contribuição são mais exploradas em uma área de pesquisa é importante para encontrar lacunas de pesquisa ou avaliar tendências na área. Os trabalhos podem possuir mais de uma faceta como contribuição. Em relação à faceta de

Tabela 4. Estudos Distribuídos pelo Contexto

Contexto	Quantidade de Estudos
Medicina	2
Sistemas Embarcados	10
Academia	2
Automotivo	9
Sistemas Críticos de Segurança	11
Internet das Coisas / Sistemas Ciberfísicos	7
Aeronáutico	4
Realidade Aumentada	1
Naval	1
N/A	12

contribuição dos estudos, estabelecemos quatro categorias: modelo, ferramenta, método e processo.

Os resultados mostram que a maioria dos artigos contribui para a faceta de Ferramentas (50,84%). Esta categoria está relacionada à implementação de adaptadores de ferramenta OSLC, serviços da web OSLC ou ferramentas para visualizar artefatos de dados. Esses trabalhos propõem exemplos de conjuntos de ferramentas que usam adaptadores OSLC como uma solução de integração. Em relação à categoria Modelo (33,89%), trabalhos como: [Zhang and Møller-Pedersen 2014], [Gürdür et al. 2018], [Alvarez-Rodríguez et al. 2018], [Gallina et al. 2016], [Mustafa and Labiche 2017], cobrem a proposta para estender as especificações de domínios OSLC. Outro tópico representativo na categoria de modelo são abordagens para gerar especificações OSLC em ambientes baseados em MDE. Nos exemplos da categoria Processo (5,08%) encontrados, o OSLC é usado para automatizar atividades que envolvem a rastreabilidade de artefatos, conforme mostrado em trabalhos de [Baumgart and Ellen 2014], [Regan et al. 2015] e [Regan et al. 2014].

Em outros trabalhos, são propostos Métodos (33,89%) ou novas abordagens para o uso de OSLC. Assim, foi possível identificar que as contribuições dos estudos estão voltadas para a resolução de problemas em um domínio específico, como uma atividade ou um tipo de indústria. É um indicativo da lacuna de pesquisa de abordagens para soluções globais sobre OSLC, em processos especialmente planejados para ES.

3.3. QP3. Quais os tipos de pesquisa mais frequentes sobre OSLC?

A maturidade de uma pesquisa pode ser determinada pela forma como o estudo afeta a prática na área. Por esse motivo, adotamos as categorias propostas pelo estudo de Wieringa *et al.* [Wieringa et al. 2005], mapeando os tipos de pesquisa empregados por cada pesquisa. Assim, são considerados os seguintes tipos de estudos: *Evaluation Research*, *Validation Research*, *Solution Proposal*, *Experience Report*, *Opinion Paper* e *Philosophical Paper*.

A maioria dos artigos é classificada como *Solution Proposal* (61,01%), reforçando a ideia de que o OSLC possui implementações como prova de conceito e protótipos em ambientes reais. Alguns trabalhos sugerem a realização de estudos de caso, embora sejam apresentados dados insuficientes, levando a um pequeno número de estudos empíricos, in-

cluindo *Validation Research* (5,08%) e *Evaluation Research* (8,47%). Os artigos classificados como *Philosophical Papers* (20,33%) discutem como usar o OSLC para solucionar problemas, mas não apresentam sua implementação. Existem também diretrizes, opiniões e relatórios baseados em experiências profissionais na indústria, resultando em artigos de *Experience Report* (8,47%) e *Opinion Paper* (1,69%), afirmando que o OSLC é utilizado em ambientes reais.

Um grande número de empresas forneceram a validação das especificações OSLC no setor, envolvendo um número significativo de funcionários; um exemplo é a montadora Scania. Devido a requisitos impostos por normas governamentais ou ISOs para a operação desses sistemas, o OSLC é proposto como uma alternativa para estabelecer integrações na camada de dados dos aplicativos que os formam ecossistemas, especialmente em ambientes de teste.

Embora o OSLC atenda às necessidades desses ambientes, os autores mostram que integrar ambientes não é uma tarefa trivial. Além disso, a integração de tais ambientes requer profissionais especializados, o que pode aumentar o custo de sua adoção. Um dos opções que as empresas procuram é adotar abordagens como MDD e MDE para a geração de interfaces integradoras. Este cenário pode ser uma alternativa para meios e pequenas empresas a adotarem OSLC em seus ambientes. A distribuição dos estudos pelos tipos de pesquisa é apresentada na Tabela 5.

Tabela 5. Estudos Distribuídos pelo Tipo de Pesquisa

Tipo de Pesquisa	Quantidade de Estudos
Evaluation Research	5
Validation Research	3
Solution Proposal	36
Philosophical Paper	12
Experience Paper	5
Opinion Paper	1

4. Ameaças à Validade

Os trabalhos empíricos devem ser capazes de conter informações suficientes para que seja possível replicá-lo apenas com o acesso ao protocolo. Entretanto, durante sua execução existem alguns fatores que podem interferir nos resultados, sendo tratados como ameaças à validade. A seguir seguem algumas ameaças deste SMS:

Validade interna: Esse tipo de ameaça refere-se à validade da análise realizada, validando se as conclusões derivadas dos dados são válidas internamente. Nesse sentido, perguntas de pesquisa bem definidas possibilitaram concluir resultados de acordo com o tópico de pesquisa. Além disso, uma preocupação constante do estudo foi garantir que o processo de seleção de trabalhos ocorresse conforme os critérios de inclusão e exclusão definidos. Para esse fim, foi cuidadosamente investido um grande esforço para filtrar os estudos primários, com o auxílio de ferramentas para organização e filtros dos trabalhos utilizados pelos pesquisadores.

Validade externa: Diz respeito sobre o quão generalizadas são as descobertas do estudo de mapeamento. Nesse sentido, a representatividade estatística pode ser uma

ameaça. Por considerar um número expressivo de artigos, essa ameaça é tratada pelo refinamento da *string* de busca por mais de um pesquisador, permitindo a inclusão de 59 estudos primários, os quais caracterizam um bom poder estatístico e a adaptação do texto para cada uma das bases selecionadas.

Validade de construção: O trabalho foi executado com base em um protocolo bem definido e amplamente utilizado na área de Engenharia de Software. Para garantir a extração dos dados, o esquema de classificação foi revisado e discutido por um segundo pesquisador. Além disso, o trabalho conta com critérios de qualidade para identificar os trabalhos mais relevantes para o tema de pesquisa. Entretanto, as categorias escolhidas para o esquema de classificação podem ser consideradas genéricas demais. Para mitigar esse viés, as categorias foram escolhidas a partir da base de conhecimento do RUP.

Validade da conclusão: Uma ameaça a conclusão pode ocorrer na fase de classificação, no qual pode ser influenciada pelo viés do *know-how* dos autores. Para superar esse viés, foi utilizado um formulário de extração de dados que contém todos os dados necessários para responder as questões de pesquisa.

5. Discussão

Na Engenharia de Software (ES), o principal desafio da integração de ferramentas é definir como as ferramentas com diferentes características podem trabalhar juntas. Para atingir um objetivo comum, as ferramentas podem precisar usar o mesmo tipo de dados, convenções de interface do usuário e funcionalidades [Thomas and Nejmeh 1992].

O desenvolvimento de software tem se caracterizado por desconexões entre atividades como planejamento, implementação de código-fonte e testes [Fitzgerald and Stol 2017]. Essas atividades são executadas com o suporte de ferramentas de software que possuem característica heterogêneas e esse é um dos principais motivos pelos quais os ambientes de desenvolvimento não são integrados.

Recentemente a prática de integração contínua surgiu com a proposta de integrar atividades de desenvolvimento e entrega de software. Entretanto, o desenvolvimento de software envolve atividades que vão além de codificação de software, mas também as atividades de governança e manutenção de software [Tüzün et al. 2019].

O padrão OSLC surge como alternativa para a integração de ferramentas e devido a sua característica de prover suporte a qualquer domínio do ciclo de vida do software é possível atingir o objetivo da Engenharia de Software Contínua de integrar todo o ambiente de desenvolvimento de forma automatizada. As vantagens do OSLC são obtidas a partir de princípios de dados ligados. Rastreabilidade, consistência e a troca de dados são alcançados por meio da especificação de domínios OSLC, e assim, a colaboração entre as diferentes equipes de desenvolvimento é melhorada.

Apesar do OSLC possuir mais de uma década de existência e ser utilizado por grandes empresas de software em contextos indústrias críticos de segurança, ainda há desafios para se popularizar em outros cenários de desenvolvimento de software devido ao conhecimento exigido das múltiplas configurações das diferentes ferramentas de uma cadeia. A difícil implementação do OSLC, seja pela falta de documentação ou pelo alto conhecimento técnico exigido ocasiona em um esforço de desenvolvimento maior do que aquele exigido por uma integração de serviço da web simples.

Estabelecer cadeias de ferramentas OSLC exige dos engenheiros de software o conhecimento de como criar interfaces de ferramentas para a troca de dados interoperada por meio de adaptadores. Essa tarefa não é fácil e requer etapas bem definidas que ajudam os engenheiros de software na automação do processo de desenvolvimento de software.

Uma alternativa para o OSLC se popularizar é o reúso dos adaptadores. Dessa forma, a pesquisa sobre o OSLC pode ser uma oportunidade para promover o reúso de software e assim poder ser utilizada a recomendação de adaptadores por meio do *design* oportunista. Mikkonen *et al.* [Mikkonen and Taivalsaari 2019] argumenta que o principal desafio no *design* oportunista é que ele não segue nenhuma sistemática. Dessa forma é importante mapear o contexto em que o adaptador pode ser aplicado bem como os domínios OSLC utilizados e elementos de entrada e saída das interfaces e assim disponibilizar essas informações para que outros desenvolvedores possam reutilizar o *asset*.

O OSLC é motivado por integrações ponto a ponto e, para isso, é necessário criar uma interface para troca de dados entre aplicações denominados adaptadores. Os adaptadores visam transformar os dados de um aplicação no formato de outra. A criação de tais adaptadores não é uma tarefa trivial, sendo onerosa para a organização. É necessário filtrar ferramentas, recursos (que serão trocados entre eles), atributos (que serão representados pelo OSLC) e até mesmo os relacionamentos vinculados em todo a cadeia.

Existem abordagens que geram código-fonte de adaptadores por meio da representação do modelo. Essas abordagens já foram validadas em contextos reais e apresentam pontos positivos como a diminuição de tempo para desenvolver as interfaces. O contexto em que foram validados relata que as abordagens MDE são convencionais em tais ambientes. O desenvolvimento de adaptadores OSLC com o Eclipse Lyo foi testado indústria como solução para geração de código-fonte de forma semiautomática, por meio da representação de metamodelos EMF [Biehl et al. 2012] [El-Khoury et al. 2016].

Outro exemplo é Biehl *et al.* [Biehl et al. 2013], onde é proposta a construção de adaptadores OSLC para integração de ferramentas web e desktop, utilizando a DSL *Tool Integration Language (TIL)*. Este trabalho é realizado pela substituição de ferramentas *desktop* por soluções web, de forma que essas cadeias de ferramentas permaneçam estáveis durante este período de transição.

Por fim, não há receita para desenvolver adaptadores OSLC. Um aspecto comum nos trabalhos relacionados é que os autores seguem estritamente as especificações OSLC, tentando mapear atributos de artefatos que são compartilhados com domínios OSLC, embora seja uma alternativa viável para a construção de adaptadores, seu uso na indústria indica que o conhecimento das vantagens do OSLC não são disseminadas nos mais diversos contextos de desenvolvimento de software, portanto, é necessário um especialista em conhecimentos de domínio. Além disso, as soluções propostas por todos os trabalhos que encontramos na literatura indicam que existe um interesse da indústria em melhorar a geração de códigos de adaptadores OSLC por meio da representação de modelos.

6. Considerações Finais

As especificações OSLC estão se tornando cada vez mais populares na indústria de software. Ao longo dos anos, desafios e práticas foram relatados na literatura para conectar artefatos entre domínios, aplicações e organizações. Nesse sentido, este trabalho empírico

ajuda a compreender as áreas potenciais de emprego do OSLC com base na análise de artigos selecionados. Por exemplo, nossos resultados mostram as motivações e contribuições do OSLC para apoiar a integração de ferramentas com base nas disciplinas da Engenharia de Software, contexto industrial, facetas da contribuição e tipo de avaliação de tais trabalhos.

Os trabalhos encontrados destacam a geração de código-fonte para conectar ferramentas. Os autores procuram maneiras de representar os artefatos bem como explorar ontologias e a extensão dos domínios OSLC. Além disso, é consenso da indústria integrar ferramentas por meio de adaptadores, visto que as ferramentas geralmente não possuem suporte nativo ao OSLC. Nesse contexto, as abordagens de Desenvolvimento Dirigido por Modelos (MDD) e MDE são de interesse da indústria para gerar essas interfaces e ao longo dos anos, vem se consolidando como a alternativa mais utilizada para o desenvolvimento de adaptadores OSLC devido ao custo benefício da abordagem. Apesar disso, a geração de código não é 100% automática e ainda não atingiu o nível de integração de todo o ambiente. Este cenário pode estar relacionado com o elevado desafio técnico necessário para integrar tais aplicações.

O primeiro trabalho que encontramos foi publicado em 2010 e relata a pesquisa e o potencial para uma nova especificação para integração de ferramentas. Desde então, pesquisadores exploraram o OSLC e propuseram novas abordagens para melhorar a integração de ferramentas em ambientes ALM [Tüzün et al. 2019].

Os principais benefícios do OSLC estão relacionadas aos Dados Ligados e envolve não apenas adaptadores de ferramentas para integração ponto a ponto, mas também abordagens propostas para promover a substituição de ferramentas na cadeia de ferramentas, incluindo modificação de especificações de domínio OSLC e soluções para atividades automatizadas para integração de ferramentas. Todos esses elementos podem ser realizados por meio de abordagens como o MDE, que através de uma representação comum, pode fornecer todo o dispositivo que permite a automação das atividades e integração de todo o ambiente de desenvolvimento de software.

O OSLC é motivado por problemas de integração de contextos industriais de alta complexidade. Nesse sentido, as soluções de integração propostas pelos autores geralmente são projetadas para domínios específicos de aplicações de software de engenharia de sistemas. Por exemplo, a maioria dos estudos é dedicada a sistemas embarcados e a outros contextos de aplicações da indústria automobilística, naval, entre outros. Embora o OSLC proponha a integração de ponta a ponta de todo o ciclo do software, os estudos mostram que as cadeias de ferramentas existentes são, em sua maior amostra, adotadas nos estágios iniciais do projeto de software e também nas que envolvem atividades de verificação e validação.

Por fim, nós seguimos um protocolo replicável que pode ser implementado ao longo dos anos para identificar novas contribuições na área. Esse fator é especialmente importante para a comunidade acadêmica, pois fornece uma visão geral do estado da arte e da prática, sobre as áreas exploradas, as deficiências para adoção do OSLC e também as áreas inexploradas. Além disso, este trabalho pode servir para tomada de decisão no desenvolvimento de soluções de integração na indústria.

7. Agradecimentos

Este estudo foi parcialmente financiado através da Pró-Reitoria de Pesquisa (PROPESQ), com bolsa do programa AGP (Apoio a Grupos de Pesquisa), e pela FAPERGS, por meio do projeto ARD N. 19/2551-0001268-3.

Referências

Aichernig, B. K., Hörmaier, K., Lorber, F., Nickovic, D., Schlick, R., Simoneau, D., and Tiran, S. (2014). Integration of requirements engineering and test-case generation via oslc. In *2014 14th International Conference on Quality Software*, pages 117–126.

Alvarez-Rodríguez, J., Mendieta, R., Vara, J., Fraga, A., and Llorens, J. (2018). Enabling system artefact exchange and selection through a linked data layer. *Journal of Universal Computer Science*, 24(11):1536–1560. cited By 1.

Arnould, V. (2018). Using model-driven approach for engineering the system engineering system. In *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, pages 608–614.

Baumgart, A. and Ellen, C. (2014). A recipe for tool interoperability. In *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 300–308.

Biehl, M., El-Khoury, J., and Törngren, M. (2012). High-level specification and code generation for service-oriented tool adapters. In *2012 12th International Conference on Computational Science and Its Applications*, pages 35–42.

Biehl, M., Gu, W., and Loiret, F. (2012). Model-based service discovery and orchestration for oslc services in tool chains. In Brambilla, M., Tokuda, T., and Tolksdorf, R., editors, *Web Engineering*, pages 283–290, Berlin, Heidelberg. Springer Berlin Heidelberg.

Biehl, M., Sosa, J. D., Törngren, M., and Díaz, O. (2013). Efficient construction of presentation integration for web-based and desktop development tools. In *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*, pages 697–702.

Buffoni, L., Pop, A., and Mengist, A. (2017). Traceability and impact analysis in requirement verification. In *Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, EOOLT '17, pages 95–98, New York, NY, USA. ACM.

d. Martino, B., Esposito, A., Nacchia, S., and Maisto, S. A. (2016). Towards a uniform semantic representation of business processes, uml artefacts and software assets. In *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 543–548.

El-khoury, J. (2016). Lyo code generator: A model-based code generator for the development of oslc-compliant tool interfaces. *SoftwareX*, 5:190 – 194.

El-Khoury, J., Ekelin, C., and Ekholm, C. (2016). Supporting the linked data approach to maintain coherence across rich emf models. In Wasowski, A. and Lönn, H., editors, *Modelling Foundations and Applications*, pages 36–47, Cham. Springer International Publishing.

Fitzgerald, B. and Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176–189.

Gallina, B., Padira, K., and Nyberg, M. (2016). Towards an iso 26262-compliant oslc-based tool chain enabling continuous self-assessment. In *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 199–204.

Gotel, O. and Mäder, P. (2012). *Acquiring Tool Support for Traceability*, pages 43–68. Springer London, London.

Gürdür, D., Feljan], A. V., El-khoury, J., Mohalik], S. K., Badrinath, R., Mujumdar], A. P., and Fersman, E. (2018). Knowledge representation of cyber-physical systems for monitoring purpose. *Procedia CIRP*, 72:468 – 473. 51st CIRP Conference on Manufacturing Systems.

Lu, J., W ang, J., Chen, D., Wang, J., and TöRngren, M. (2018). A service-oriented tool-chain for model-based systems engineering of aero-engines. *IEEE Access*, 6:50443–50458.

Mikkonen, T. and Taivalsaari, A. (2019). Software reuse in the era of opportunistic design. *IEEE Software*, 36(3):105–111.

Mustafa, N. and Labiche, Y. (2017). Employing linked data in building a trace links taxonomy. pages 186–198. cited By 2.

OSLC (2020). Open services for lifecycle collaboration primer web page. Accessed at February 2020.

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1 – 18.

Regan, G., Biro, M., Flood, D., and McCaffery, F. (2015). Assessing traceability - practical experiences and lessons learned. *Journal of Software: Evolution and Process*, 27(8):591–601. cited By 6.

Regan, G., Biro, M., Mc Caffery, F., Mc Daid, K., and Flood, D. (2014). A traceability process assessment model for the medical device domain. In Barafort, B., O'Connor, R. V., Poth, A., and Messnarz, R., editors, *Systems, Software and Services Process Improvement*, pages 206–216, Berlin, Heidelberg. Springer Berlin Heidelberg.

Thomas, I. and Nejmeh, B. A. (1992). Definitions of tool integration for environments. *IEEE Software*, 9(2):29–35.

Tüzün, E., Tekinerdogan, B., Macit, Y., and İnce, K. (2019). Adopting integrated application lifecycle management within a large-scale software company: An action research approach. *Journal of Systems and Software*, 149:63 – 82.

VanZandt, L. (2015). Enabling rational decision making with provenance-annotated oslc relationships. In *2015 IEEE International Symposium on Systems Engineering (ISSE)*, pages 346–352.

Wicks, M. N. and Dewar, R. G. (2007). Controversy corner: A new research agenda for tool integration. *J. Syst. Softw.*, 80(9):1569–1585.

Wieringa, R., Maiden, N., Mead, N., and Rolland, C. (2005). Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir. Eng.*, 11(1):102–107.

Zhang, W. and Møller-Pedersen, B. (2013). Establishing tool chains above the service cloud with integration models. In *2013 IEEE 20th International Conference on Web Services*, pages 372–379.

Zhang, W. and Møller-Pedersen, B. (2014). Modeling of tool integration resources with oslc support. In *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 99–110.