Investigando a Integração de Ferramentas com OSLC Através do Eclipse Lyo

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

[1]Universidade Federal do Pampa (UNIPAMPA), PPGES
Alegrete - RS

[2]Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ)
Ijuí - RS

brunoferreira.aluno@unipampa.edu.br

Abstract. *The software industry invests in modern tools throughout the software development lifecycle. However, there are challenges to achieving an integrated end-to-end environment, such as dealing with multiple tool configurations and establishing project data interoperation in real time. To mitigate these challenges, many approaches have been proposed for integrating tools. This paper presents an exploratory study on a technology for modeling integration solutions and the respective generation of integration configurations in Open Services for Lifecycle Collaboration (OSLC) standard. The results are interesting and suggest that Eclipse Lyo provides some benefits to software engineers when configuring integration solutions in Software Engineering application lifecycles.*

Resumo. *A indústria de software investe em ferramentas modernas ao longo de todo o ciclo de vida de desenvolvimento de software. No entanto, existem desafios para alcançar um ambiente integrado de ponta a ponta, como por exemplo lidar com múltiplas configurações de ferramentas e estabelecer o compartilhamento de dados do projeto em tempo real. Para mitigar esses desafios, muitas abordagens foram propostas para a integração de ferramentas. Este artigo apresenta um estudo exploratório sobre uma tecnologia para modelagem de soluções de integração e respectiva geração de configurações de integração no padrão Open Services for Lifecycle Collaboration (OSLC). Os resultados são interessantes e sugerem que Eclipse Lyo provê alguns benefícios aos engenheiros de software na configuração de soluções de integração em ciclos de vida de aplicações de Engenharia de Software.*

1. Introdução

Ferramentas de software são utilizadas para apoiar profissionais na execução de atividades ao longo do ciclo de vida do software. Essas ferramentas possuem diferentes funcionalidades, configurações, fabricantes e são desenvolvidas sem o suporte nativo para serem integradas com outras ferramentas. Além disso, manipulam artefatos (requisitos de software, modelos, código-fonte, casos de teste, etc.) produzidos em diferentes fases do desenvolvimento. Este cenário contribui para que ambientes integrados de ponta a ponta de forma totalmente automatizada sejam raros na indústria [Wicks and Dewar 2007].

Organizações de software obtêm novas ferramentas ou desenvolvem suas soluções de software ao longo dos anos, resultando em um ecossistema de software heterogêneo [Messerschmitt 2005]. Isso demanda que as organizações adotem abordagens para melhoria dos processos de software e suporte ferramental que permitam a automação do processo de desenvolvimento através da integração de ferramentas.

Entre as alternativas está o *Application Lifecycle Management* (ALM), que adota a ideia de integração ponta a ponta, evitando silos de informações devido a características como rastreabilidade, visibilidade e automação de processos [Schwaber et al. 2006]. Para reduzir os problemas relacionados à integração de artefatos de software ao longo de várias fases do ALM, padrões de representação para modelos, metamodelos e transformações têm sido propostos na literatura [Kleppe et al. 2003]. Além disso, para mitigar a complexidade desse ambientes, várias ferramentas que auxiliam nas tarefas de ES foram propostas [Ebert 2013]. Essas ferramentas ajudam os desenvolvedores a aproveitar experiências anteriores ao especificar um novo software, enquanto ferramentas baseadas em modelos ajudam a capturar, organizar e armazenar a maioria das informações trocadas como *assets* reutilizáveis.

Para ajudar na integração de ferramentas usadas na produção de software, muitas abordagens foram propostas interoperando dados entre serviços [Biehl et al. 2014]. As abordagens foco destes trabalhos caracterizam-se por minimizar os problemas relacionados ao processo de integração. Elas são adotadas nos contextos de Engenharia de Software (ES), tipicamente implementadas por meio de uma arquitetura comum para serviços [Zhang and Møller-Pedersen 2014], o que demanda um protocolo de comunicação comum entre diferentes ferramentas.

Nesse contexto, uma emergente especificação industrial caracterizada como arquitetura comum para ferramentas de ES foi proposta: *Open Services for Lifecycle Collaboration* (OSLC) [OSLC 2020]. O OSLC é um padrão aberto para interoperabilidade de ferramentas de software, que define um modelo de representação comum para os artefatos produzidos ao longo do ciclo de vida do software, bem como métodos que permitem ferramentas compartilhar dados entre si.

Com base nisso, este trabalho, de caráter exploratório, possui o objetivo de identificar os aspectos envolvidos na modelagem de integração por meio da ferramenta Eclipse Lyo e da geração de código-fonte com base em artefatos em conformidade com o padrão OSLC. Durante a realização deste trabalho buscou-se analisar parte de um cenário de um setor de desenvolvimento de software de uma organização governamental. Este cenário composto por ferramentas de código-fonte aberto, foi parcialmente implementado, resultando em algumas contribuições como segue: (i) identificada a carência no estado da arte de um material que sintetiza ou organiza atividades para a geração de soluções de integração em OSLC, derivou-se dessa experiência um tutorial para desenvolvimento de integradores; e (ii) comprovação na prática de que o padrão OSLC e o Eclipse Lyo funcionam na integração dos artefatos do cenário explorado.

Este trabalho é organizado como segue: Na seção 2 aborda tecnologias investigadas durante a realização deste trabalho, como os conceitos do padrão OSLC e do Eclipse Lyo. A Seção 3 caracteriza o estudo conduzido no Eclipse Lyo e a Seção 4 as alternativas ao Eclpse Lyo para o desenvolvimento de adaptadores OSLC. Por fim, A Seção 5 discute

os resultados deste estudo.

2. Tecnologias Investigadas

Esta seção apresenta os conceitos das tecnologias investigadas durante a execução deste trabalho.

2.1. Open Services for Lifecycle Collaboration

Open Services for Lifecycle Collaboration (OSLC) é um padrão aberto para integração de ferramentas de software. O OSLC define uma forma de representação comum para artefatos, bem como métodos para o compartilhamento de dados em todos os domínios do projeto. As especificações OSLC consistem em regras para criar, atualizar, recuperar e ligar assets estruturados nos formatos RDF/XML e JSON. Existem também os domínios OSLC que estendem a especificação principal e definem como representar artefatos em domínios como gerenciamento de requisitos, gerenciamento de qualidade, gerenciamento de configuração e mudanças, etc.

O OSLC é baseado nos princípios dos dados ligados e segue as regras definidas por Tim Berners-Lee [Linked Data 2006] para ligar dados na web. Nesse sentido, os relacionamentos entre os artefatos em uma cadeia de ferramentas podem ser estabelecidos sem a duplicação de dados por meio de *links*. Esses *links* são mantidos em uma estrutura chamada tripla que consiste em dados de sujeito-predicado-objeto que descrevem o relacionamento entre artefatos, partes interessadas e atividades. 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 através do OSLC.

Em uma cadeia de ferramentas OSLC, uma aplicação pode ser classificada como Provedora ou Consumidora. Provedores destinam-se a armazenar e fornecer dados, permitindo aos consumidores acesso fácil 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 de suporte nativo é recomendada para desenvolvedores de ferramentas. As abordagens *plugin* e adaptador são semelhantes, entretanto a construção de *plugins* são recomendadas apenas nos casos em que há o conhecimento das tecnologias que envolvem a construção da ferramenta, como linguagem de programação e arquitetura. Em outros cenários, a abordagem recomendada é a construção de adaptadores OSLC. Nesse contexto, surge a necessidade de abordagens para se implementarem adaptadores OSLC, como por exemplo Desenvolvimento Dirigido por Modelos (MDD) [Fowler and Parsons 2011].

2.2. Eclipse Lyo

O Eclipse Lyo é um projeto aberto que visa ajudar a comunidade interessada em integrações a adotar as especificações OSLC em suas ferramentas [El-khoury 2016]. O Lyo possibilita o desenvolvimento de novas soluções compatíveis com o OSLC por meio da abordagem de MDD, que permite engenheiros a trabalhar com um nível maior de abstração através de Linguagens Específicas de Domínio (DSLs).

O projeto consiste nos seguintes componentes: OSLC4J SDK, que é um kit de desenvolvimento java para interfaces de ferramentas provedoras e consumidoras OSLC e o Lyo Designer, que é um gerador de código-fonte baseado em modelos, permitindo a representação gráfica dos modelos de integrações. Entretanto, o código-fonte gerado possui apenas um conjunto de métodos que permitem a comunicação entre as ferramentas e para acessar os dados armazenados internamente é necessário implementá-los manualmente.

Existem três perspectivas para a representação de modelos no Lyo Designer: Perspectiva Especificação de Domínios, Perspectiva Cadeia de Ferramentas e Perspectiva Interface do Adaptador. Cada perspectiva permite representar um tipo de diagrama, como visto a seguir:

1. **Perspectiva Especificação de Domínios -** Trata sobre a representação dos domínios OSLC. É possível adicionar múltiplos domínios. Cada domínio é composto por *Resources* (artefatos) e *Resources Properties* (valores permitidos, cardinalidade e opcionalidade).
2. **Perspectiva Cadeia de Ferramentas -** Refere-se a estrutura da cadeia de ferramentas e artefatos que serão compartilhados entre as aplicações, definindo quais as ferramentas que serão consumidoras e provedoras.
3. **Perspectiva Interface do Adaptador -** Representa a estrutura dos adaptadores de cada ferramenta representada na perspectiva Cadeia de Ferramentas. Essa estrutura segue a especificação principal do OSLC, sendo a base para a geração de códigos.

Após realizar as três perspectivas de modelagem, as abordagens baseadas em MDD permitem gerar código para adaptadores OSLC, promovendo assim a integração automatizada de um modelo independente para uma representação específica da plataforma.

3. Caracterização do Estudo Exploratório

Esta seção detalha a implementação de adaptadores OSLC por meio da geração de código-fonte baseada em modelos no Eclipse Lyo por meio de um estudo exploratório. Este tipo de estudo não possui uma análise conduzida conforme os protocolos de estudo de caso de outras naturezas como: descritivo, explanatório, e melhoria. Estudos exploratórios possuem a característica de relatar uma experiência, sendo o primeiro tipo de estudo conduzido para caracterizar a aplicabilidade de determinada solução [Runeson and Höst 2008]. A Figura 1 mostra a cadeia de ferramentas do nosso estudo exploratório, composta por três ferramentas de domínios diferentes, sendo elas Excel para o gerenciamento de requisitos, Redmine para o gerenciamento de projetos e Testlink para gerenciamento de testes.

A ferramenta Excel mantém requisitos de software e deseja-se ligar através das propriedades dos dados ligados do OSLC às tarefas gerenciadas pelo Redmine e aos casos de testes do Testlink. Com base nisso, o adaptador OSLC do Excel possui duas interfaces consumidoras para tarefas e casos de testes respectivamente e uma interface que provê requisitos de software. Além disso, as ferramentas Redmine e Testlink possuem interfaces provedoras para o compartilhamento de dados.

Embora nosso cenário tenha definido a estrutura da cadeia de ferramentas do nosso estudo, o primeiro passo para gerar os códigos-fonte dos adaptadores é a representação

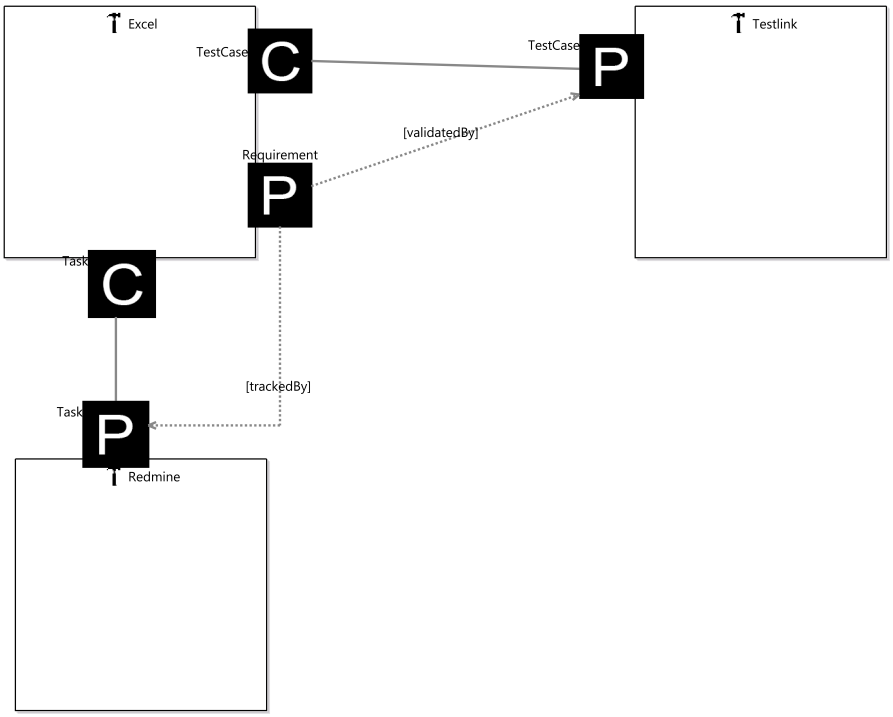

Figura 1. Estrutura da Cadeia de Ferramentas

gráfica dos domínios OSLC. Como mostra a Figura 2, exploramos três domínios OSLC: *Requirements Management (RM)* [1], *Change Management (CM)* [2] e *Quality Management (QM)* [3]. Cada um desses domínios possuem propriedades e *resources* para a representação dos artefatos mantidos pelas ferramentas e seus relacionamentos. Por exemplo, o domínio Requirement Management possui o *resource* Requirement com as propriedades que o descrevem como identificador, data de criação e modificação. Além disso, nessa representação são estabelecidos os relacionamentos entre os artefatos a partir de propriedades como a *validateBy* em que é possível ligar um caso de teste que valida determinado requisito. Além dos domínios OSLC, também exploramos outros domínios auxiliares, que tem objetivo de padronizar informações da web, como o FOAF [4] e Dublin Core [5].

A partir de serviços OSLC, é possível recuperar, editar, atualizar e excluir artefatos de software gerados pela cadeia de ferramentas. Essas funcionalidades são modeladas na perspectiva do adaptador. A Figura 3 mostra a interface do adaptador da ferramenta Redmine. A estrutura desse adaptador segue as especificações da Especificação principal do OSLC. Em adaptadores OSLC, todos os serviços são acessados através de *Uniformd Resource Identifier* (URIs). Ao acessar o adaptador é disponibilzado um catálogo de serviços (*Service Provider Catalog*) que mantém URIs para todos os provedores de serviços (*Service Providers*), que representam os projetos cadastrados no Redmine. Cada provedor de

[1]http://docs.oasis-open.org/oslc-domains/oslc-rm/v2.1/oslc-rm-v2.1-part2-requirements-management-vocab.html

[2]http://docs.oasis-open.org/oslc-domains/cm/v3.0/cs02/part2-change-mgt-vocab/cm-v3.0-cs02-part2-change-mgt-vocab.html

[3]https://docs.oasis-open-projects.org/oslc-op/qm/v2.1/psd02/quality-management-vocab.html

[4]http://xmlns.com/foaf/spec/

[5]https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

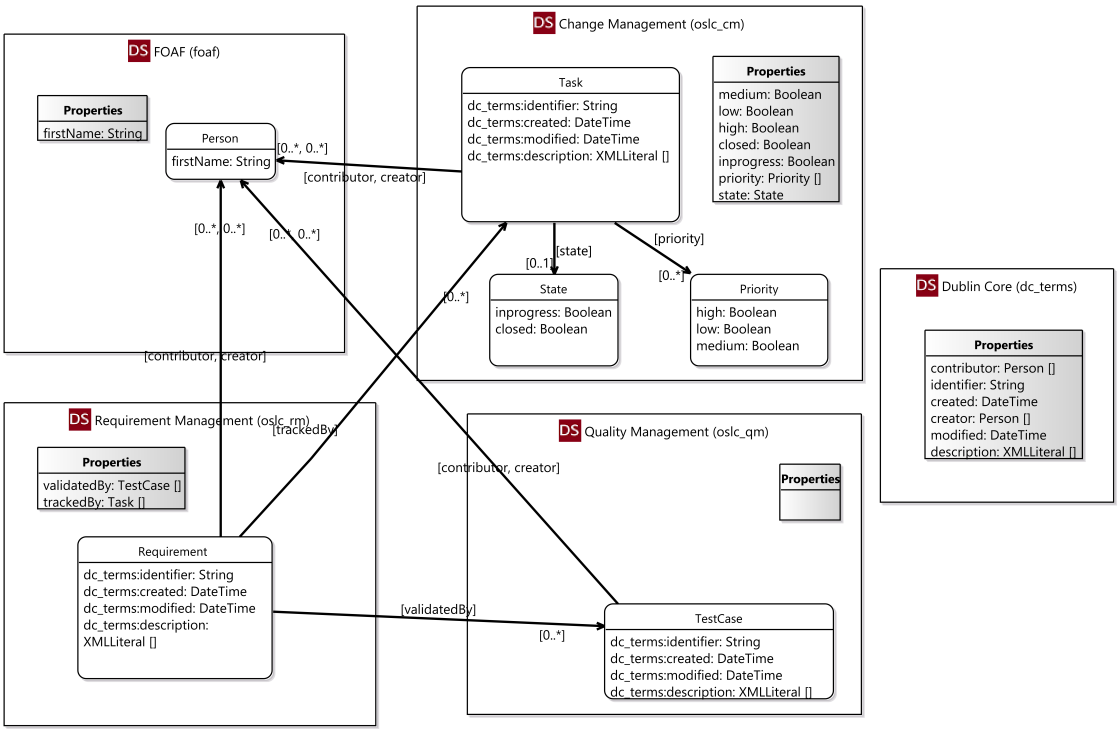

Figura 2. Representação Gráfica dos Domínios OSLC no Eclipse Lyo

serviço disponibiliza URIs para acessar o serviços (*Services*), que nesse cenário são as tarefas de cada um dos projetos. Ao final do fluxo, cada é possível acessar cada tarefa e manipulá-la através das funções Restful que foram modeladas na perspectiva da interface do adaptador.

O Lyo gera apenas os esqueletos de classes java, sendo necessário implementar manualmente a lógica dos adaptadores. Para que isso seja possível, as ferramentas devem possuir alguma interface que possibilite manipular seus dados, como uma API RESTful. Em nosso estudo, usamos três APIs RESTful, uma para cada ferramenta envolvida no processo. Graças ao componente OSLC4J do Lyo, todos os objetos tornam-se possível de serem representados nos formatos RDF/XML e JSON, possibilitando a troca de dados entre as ferramentas após a transformação dos seus dados em objetos java para o padrão OSLC.

4. Trabalhos Relacionados

Integração de ferramentas em ambientes de ES trata sobre como ferramentas com características heterogêneas podem ser compatíveis em diferentes aspectos. Esses aspectos incluem os formatos dos dados, convenções de interface do usuário, funcionalidades da aplicação e outros aspectos da construção da ferramenta [Thomas and Nejmeh 1992].

Ainda que o maior desafio seja prover um ambiente integrado de forma automatizada que compreenda todo o ciclo de vida do software [Brown and Penedo 1992], existem diversas abordagens para integrar ferramentas de software. Cada uma dessas abordagens envolve uma série de características que devem ser levadas em consideração durante sua escolha [Hohpe 2003].

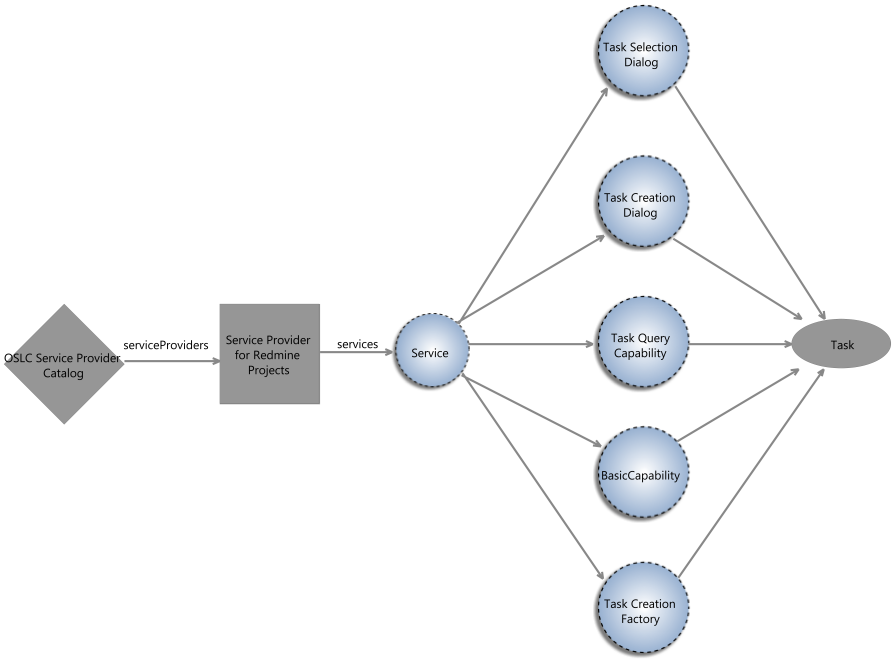

Figura 3. Interface do Adaptador OSLC para a ferramenta Redmine

Nosso estudo foca na análise de apenas uma abordagem. Além do Eclipse Lyo, existem outras alternativas para implementar soluções de integração de ferramentas baseadas em OSLC:

1. **Guaraná -** O Guaraná é uma abordagem para o suporte de implementações de soluções Enterprise Application Integration [Frantz et al. 2016]. Além disso, Guaraná possui uma linguagem gráfica denominada Guaraná DSL, que permite engenheiros de software a projetar soluções de integração independente de plataformas de integração.
2. **Tool Integration Language (TIL) -** É uma linguagem de domínio específico para cadeia de ferramentas [Biehl 2011]. TIL serve para diversos propósitos, entre eles a geração de códigos de adaptadores a partir da representação de modelos independente da tecnologia, inclusive OSLC.

5. Considerações Finais

Esse trabalho apresentou um estudo exploratório sobre a ferramenta Eclipse Lyo com o objetivo de implementar adaptadores OSLC. Para isso, foram utilizadas abordagens para geração de código-fonte baseada em modelos. Essas abordagens propõem mitigar alguns desafios que envolvem o processo de integração de ferramentas pois permitem desenvolvedores a trabalhar com um nível maior de abstração. Apesar disso, foram encontradas poucas alternativas para a construção de adaptadores OSLC baseadas em DSLs na literatura.

O Eclipse Lyo surge como objeto de estudo para se atingir ambientes integrados de ponta a ponta por meio do OSLC. Nosso trabalho possibilitou identificar a estrutura que um projeto OSLC segue, bem como as tecnologias envolvidas no processo do desenvolvimento dos adaptadores.

Não existe uma receita para desenvolver adaptadores OSLC, o que dificultou bastante o aprendizado sobre a DSL Lyo e o respectivo suporte ferramental que suporta a execução das rotinas de integração. Apesar do Eclipse Lyo ser utilizado para desenvolver adaptadores OSLC na indústria, como visto nos trabalhos [Biró et al. 2017], [El-khoury et al. 2019], [Nardone et al. 2020], [Gürdür et al. 2018] e [Zhang and Møller-Pedersen 2014], esses estudos propõem integrações em contextos de sistemas críticos de segurança. Dessa forma, o desenvolvimento de adaptadores OSLC através do Lyo não foi totalmente explorado na área de ES. Além disso, cada cenário de integração em um ciclo de vida de aplicação de uma empresa possui particularidades que devem ser levadas em consideração na hora de tomar as decisões do projeto. Por mais que o estudo tenha considerado representações de ferramentas bastante utilizadas no desenvolvimento de software, o cenário representado jamais pode refletir um ambiente real de uma empresa.

Observou-se que o aparato ferramental de suporte propicia a geração automática de código. Entretanto, apenas os esqueletos dos códigos são gerados dessa forma. É necessário implementar manualmente o conteúdo dos métodos para que seja estabelecida a comunicação entre as ferramentas provedoras e consumidoras, possibilitando a troca dos artefatos mantidos por elas. Isso caracteriza uma limitação do aparato ferramental investigado, uma vez que a geração de 100% do código poderia ser obtida para OSLC baseado na DSL investigada. Os adaptadores desenvolvidos ainda precisam ser customizados e implementar algumas interfaces gráficas para busca e acesso aos artefatos. Apesar disso, com base nesse estudo foi possível identificar os componentes mínimos necessários para implementar um adaptador de ferramentas, e portanto viável para a execução de um estudo exploratório em domínios de ES.

As integrações com OSLC limitam-se a apenas algumas fases do desenvolvimento. Um dos motivos disso é a complexidade para desenvolver adaptadores para essas soluções de integrações. O Estudo do Eclipse Lyo possui entre os objetivos identificar as atividades necessárias para o desenvolvimento de adaptadores e possíveis vantagens e desvantagens que motivem a evolução da ferramenta por ser um projeto de código-fonte aberto ou a proposta de novas alternativas para o desenvolvimento de adaptadores OSLC com o intuito de fomentar a transferência de conhecimento tácito em explícito do OSLC para engenheiros de software.

Os próximos passos da pesquisa incluem finalizar o desenvolvimento dos adaptadores, realizar uma avaliação empírica da viabilidade do uso do Eclipse Lyo em ambientes organizacionais com a execução de um estudo de caso, aprofundar os conceitos de representações comuns de ativos de software para diferentes áreas do conhecimento e explorar outras DSLs identificadas na literatura para a representação de elementos de integração de ferramentas de ES por meio de OSLC.

Por fim, o OSLC possui um grande potencial como tecnologia para automatizar todo o ambiente de desenvolvimento devido a suas características como flexibilidade e reuso. Além disso, há o interesse da indústria no desenvolvimento de processos, metodologias e ferramentas motivadas principalmente para reduzir o custo operacional dessas integrações, inclusive por meio do Eclipse Lyo.

6. 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.

Biehl, M. (2011). Tool integration language (til). Technical Report 2011:14, KTH, Mechatronics. QC 20111130.

Biehl, M., El-Khoury, J., Loiret, F., and Törngren, M. (2014). On the modeling and generation of service-oriented tool chains. *Software & Systems Modeling*, 13(2):461–480.

Biró, M., Kossak, F., Klespitz, J., and Kovács, L. (2017). Graceful integration of process capability improvement, formal modeling and web technology for traceability. In Stolfa, J., Stolfa, S., O'Connor, R. V., and Messnarz, R., editors, *Systems, Software and Services Process Improvement*, pages 381–398, Cham. Springer International Publishing.

Brown, A. W. and Penedo, M. H. (1992). An annotated bibliography on integration in software engineering environments. *SIGSOFT Softw. Eng. Notes*, 17(3):47–55.

Ebert, C. (2013). Improving engineering efficiency with plm/alm. *Software & Systems Modeling*, 12(3):443–449.

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., Berezovskyi, A., and Nyberg, M. (2019). An industrial evaluation of data access techniques for the interoperability of engineering software tools. *Journal of Industrial Information Integration*.

Fowler, M. and Parsons, R. (2011). *Domain-specific languages*. Addison-Wesley.

Frantz, R. Z., Corchuelo, R., and Roos-Frantz, F. (2016). On the design of a maintainable software development kit to implement integration solutions. *Journal of Systems and Software*, 111(1):89–104.

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.

Hohpe, G. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional.

Kleppe, A., Warmer, J., and Bast, W. (2003). Mda explained: The model driven architecture: Practice and promise.

Linked Data (2006). Linked data web page. Accessed at April 2020.

Messerschmitt, D. G. (2005). *Software Ecosystem: Understanding an Indispensable Technology and Industry (The MIT Press)*. The MIT Press.

Nardone, R., Marrone, S., Gentile, U., Amato, A., Barberio, G., Benerecetti, M., Guglielmo], R. D., Martino], B. D., Mazzocca, N., Peron, A., Pisani, G., Velardi, L., and Vittorini, V. (2020). An oslc-based environment for system-level functional testing of ertms/etcs controllers. *Journal of Systems and Software*, 161:110478.

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

Runeson, P. and Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131.

Schwaber, C. et al. (2006). The changing face of application life-cycle management. *Forrester Research*, 18.

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

Wicks, M. and Dewar, R. (2007). A new research agenda for tool integration. *Journal of Systems and Software*, 80(9):1569 – 1585. Evaluation and Assessment in Software Engineering.

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.