Explorando a Integração de Ferramentas para a Engenharia de Software Contínua com o OSLC e Eclipse Lyo

Bruno Marcelo Soares Ferreira[1], **Fábio Basso**[1]

[1]Universidade Federal do Pampa (UNIPAMPA)

Abstract. *Software tools are used to support teams throughout the software development lifecycle. Yet, fully automated integrated environments are rarelly observed in the industry. In this sense, many approaches have been proposed for Continuous Software Engineering (CSE), in special those integrating tools with the Open Services for Lifecycle Collaboration (OSLC) standard. This paper reports on our experience through an exploratory study on a technology for modeling integration solutions, namelly Eclipse Lyo, and the respective generation of OSLC interfaces and adapters. As a result, we derived well defined steps in a metodology supporting the OSLC toolchain development process. The results are interesting and suggest that Eclipse Lyo provides some benefits to software engineers when configuring integration solutions in CSE contexts.*

Resumo. *Ferramentas de software são usadas para apoiar equipes em todo o ciclo de vida de desenvolvimento de software. No entanto, ambientes integrados totalmente automatizados são raramente observados na indústria. Nesse sentido, diversas abordagens têm sido propostas para a Continuous Software Engineering (CSE), em especial aquelas que integram ferramentas com o padrão Open Services for Lifecycle Collaboration (OSLC). Este artigo relata nossa experiência por meio de um estudo exploratório sobre uma tecnologia para modelagem de soluções de integração, denominada Eclipse Lyo, e a respectiva geração de interfaces e adaptadores OSLC. Como resultado, derivamos etapas bem definidas em uma metodologia de suporte ao processo de desenvolvimento da cadeia de ferramentas OSLC. Os resultados são interessantes e sugerem que o Eclipse Lyo oferece alguns benefícios aos engenheiros de software ao configurar soluções de integração em contextos CSE.*

1. Introdução

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 por meio da integração de ferramentas.

Nesse contexto, uma emergente especificação industrial caracterizada como arquitetura comum para ferramentas de Engenharia de Software foi proposta: *Open Services for Lifecycle Collaboration (OSLC*) [OSLC 2020]. O OSLC é um padrão aberto para interoperabilidade de ferramentas de software, o qual define um modelo de representação comum para os artefatos produzidos por meio dos domínios do projeto, bem como métodos que permitem que ferramentas compartilhem dados entre si.

Diversas abordagens têm sido propostas para a *Continuous Software Engineering (CSE)*, em especial aquelas que integram ferramentas com o padrão OSLC. 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 CSE.

Com base nisso, este trabalho apresenta uma pesquisa em andamento com caráter exploratório da ferramenta Eclipse Lyo [El-khoury 2016] com base em *Model Driven-Development (MDD)* para integração de ferramentas com o OSLC. A abordagem MDD permite a geração de código-fonte de soluções de integração, o que pode ajudar na assimilação da teoria da integração e prática.

O Eclipse Lyo é um projeto aberto que visa ajudar a comunidade interessada em integrações a adotar as especificações OSLC em suas ferramentas. O Lyo possibilita o desenvolvimento de novas soluções compatíveis com o OSLC por meio da abordagem de desenvolvimento dirigido por modelos, a qual permite que engenheiros possam trabalhar com um nível maior de abstração e produtividade por meio de *Domain-Specific Language (DSL)*.

Este trabalho é organizado como segue: A Seção 2 apresenta a metodologia e caracteriza o cenário do nosso estudo. A Seção 3 descreve o processo do trabalho executado em nosso estudo. Por fim, a Seção 4 discute os trabalhos relacionados sobre o uso Lyo na indústria e a Seção 5 conclui este trabalho.

2. Metodologia de Estudo

Esse relato de experiência considera apenas um tipo de estudo empírico: estudo de caso. Runeson *et al.* [Runeson and Höst 2008] afirmam que os estudos de caso abrangem estudos bem organizados na área até pequenos exemplos, mas o consenso é que de ocorram no mundo real e conduzidos com baixo controle em comparação com estudos de experimentos controlados [Wohlin et al. 2012].

O termo estudo de caso também é usado para descrever um estudo de campo e um estudo observacional. Portanto, diferentes taxonomias devem ser adotadas para a caracterização deste tipo de estudo, tais como: (i) Estudo exploratório, o qual busca descobrir o que está acontecendo, buscar novos *insights* e gerar ideias e hipóteses para novas pesquisas; (ii) Descritivo, retratando uma situação ou fenômeno; (iii) Explanatório, o qual busca uma explicação para uma situação ou problema, principalmente, mas não necessariamente, na forma de uma relação causal; (iv) e Melhoria, tentando otimizar um determinado aspecto do fenômeno estudado.

Nosso estudo foi construído com base em uma metodologia para estudos de caso do tipo Exploratório, executado do ponto de vista do pesquisador. No entanto, por motivos de espaço, não o relatamos como um estudo de caso, mas sim como um relato de experiência.

Nosso objetivo é analisar na prática uma parte de um cenário dessa organização, derivando um processo para desenvolvimento de integradores e identificando as dificuldades que uma equipe em cenário real enfrentará ao utilizar o aparato ferramental explorado. Trata-se de um estudo fundamental que relata a nossa experiência ao longo de um ano e

meio na investigação do tema, dedicado tanto para alinhamento conceitual, planejamento e execução de uma integração que faz uso de dados de um projeto de software real.

Portanto, é um relato de experiência que embasará os demais tipos de estudos de caso planejados para aplicação em ambiente real de desenvolvimento, incluindo estudos derivados como o Descritivo, o Explanatório e o de Melhoria. Ao melhor do nosso conhecimento, é o primeiro relato que busca caracterizar os elementos essenciais para a integração de ferramentas em vistas à uma futura implementação de engenharia de software contínua por uma fábrica de software governamental.

2.1. Caracterização do Cenário

As ferramentas que compõem a cadeia de ferramenta à serem exploradas neste estudo são mostradas na Tabela 1. Elas foram selecionadas com base no ecossistema de software da Diretoria de Tecnologia da Informação e Comunicação (DTIC) - UNIPAMPA. Esse setor de desenvolvimento trabalha com softwares livres e de código aberto e possui ferramentas que suportam atividades de desenvolvimento, gestão e comunicação. Dentre o aparato ferramental utilizado, este estudo teve por objetivo testar OSLC e Eclipse Lyo em um conjunto de três ferramentas que melhor representem um cenário de Engenharia de Software Contínua para a DTIC.

2.2. Foco de Contribuição

A implementação de um adaptador OSLC requer que muitas decisões sejam tomadas pelos desenvolvedores. Por exemplo, deve-se identificar quais os dados devem ser providos/consumidos, quais especificações de domínios utilizar, qual o fluxo da troca de mensagens, dentre outros [Leitner et al. 2016]. Nesse sentido, com o objetivo de explorar a ferramenta Eclipse Lyo, buscou-se desenvolver uma solução de integração OSLC com base na topologia ponto a ponto por meio de adaptadores OSLC. A topologia de integração ponto a ponto estabelece uma conexão direta entre duas ferramentas [Hohpe 2003]. Além disso, essa topologia é caracterizada por aplicações fortemente acopladas, portanto, quando uma ferramenta é substituída, é necessário refazer o processo de integração, resultando em significativo custo operacional.

ID	Ferramenta	Contexto	URL
T01	Redmine	Gerenciamento de Projeto	http://www.redmine.org
T02	Libreoffice /Excel	Gerenciamento de requisitos	http://pt-br.libreoffice.org
T03	Testlink	Gerenciamento de testes	http://testlink.org
T04	Mantis	Gerenciamento de testes	https://www.mantisbt.org
T05	GLPI	Gerenciamento de serviços de TI	https://glpi-project.org/pt-br
T06	Subversion	Controle de versões	https://subversion.apache.org
T07	Eclipse	Ambiente de desenvolvimento	https://www.eclipse.org

Tabela 1. Ferramentas levantadas para integração no contexto da Organização

A fim de explorar e ilustrar os benefícios da geração de códigos de adaptadores para ferramentas de diferentes domínios, foram selecionadas três ferramentas utilizadas pelos profissionais de desenvolvimento das sete identificadas inicialmente, sendo elas: Libreoffice/Excel, Redmine e Testlink.

3. Metodologia para Desenvolvimento da Integração

Este tipo de estudo pode e deve ser replicado em ambientes de desenvolvimento reais. Para tal, é essencial a definição de uma metodologia que sintetiza toda a bagagem adquirida ao longo da execução da integração em vistas à implementação da Engenharia de Software Contínua. Por um motivo de fomentar a transferência de conhecimento tácito em explícito, derivou-se um conjunto de atividades à ser desempenhada pelos engenheiros de software.

Assim, estabeleceu-se um processo para o desenvolvimento de adaptadores OSLC no Eclipse Lyo, ilustrado na Figura 1. Para seu início, é necessário que os desenvolvedores conheçam o escopo da cadeia de ferramentas, definindo quais vão compor a solução de integração. Portanto, parte-se do ponto onde o conjunto de ferramentas está bem definido pela organização.

Com esse cenário definido, o próximo passo é estabelecer os relacionamentos entre as ferramentas. Nesse subprocesso/atividade, são definidos o fluxo da troca de dados ao longo da cadeia de ferramentas representadas pelas conexões entre cada uma das ferramentas.

A integração de ferramentas com OSLC é estabelecida por meio da topologia ponto a ponto. Dessa forma, cada uma das ferramentas possuem uma ou mais ligações com as demais. Essas ferramentas podem ser classificadas como provedoras ou consumidoras e podem pertencer a ambas categorias.

Relacionado a isso, é necessário identificar quais os artefatos serão compartilhados entre as ferramentas. Por exemplo, uma ferramenta de gerenciamento de requisitos pode consumir dados de uma ferramenta de testes, enquanto uma ferramenta de análise estática de código-fonte pode prover relatórios e consumir requisitos ao mesmo tempo. Assim, não é obrigatório que todos os artefatos de uma ferramenta sejam compartilhados via OSLC.

Identificando os artefatos que serão compartilhados, ainda resta selecionar quais os atributos serão representados e compartilhados via OSLC. Essa atividade geralmente é realizada com base na interface de usuário da ferramenta. Isso não significa que seja obrigatório que todos os atributos sejam representados em OSLC. Por isso, o filtro das propriedades essenciais é importante para o restante do processo.

Além dos relacionamentos em nível de ferramenta, é necessário estabelecer quais os artefatos devem ser relacionados através dos dados ligados. Este relacionamento de propriedades no formato RDF/XML em que o artefato é transformado para ser compartilhado.

O consórcio de empresas e acadêmicos que mantém o OSLC propuseram uma série de domínios OSLC. Esses domínios significam que na teoria, qualquer ferramenta que pertença a uma fase do ciclo de vida do software (*e.g.* requisitos, testes), mantém artefatos que podem ser representados por essas especificações. Essas informações não estão disponíveis no Eclipse Lyo, sendo necessário acessar as várias páginas webs correspondentes aos domínios. Com base nisso, para escolher os domínios OSLC devem estar de acordo com os domínios das ferramentas selecionadas.

Cada um dos domínios possuem uma série de *resources* que são representações

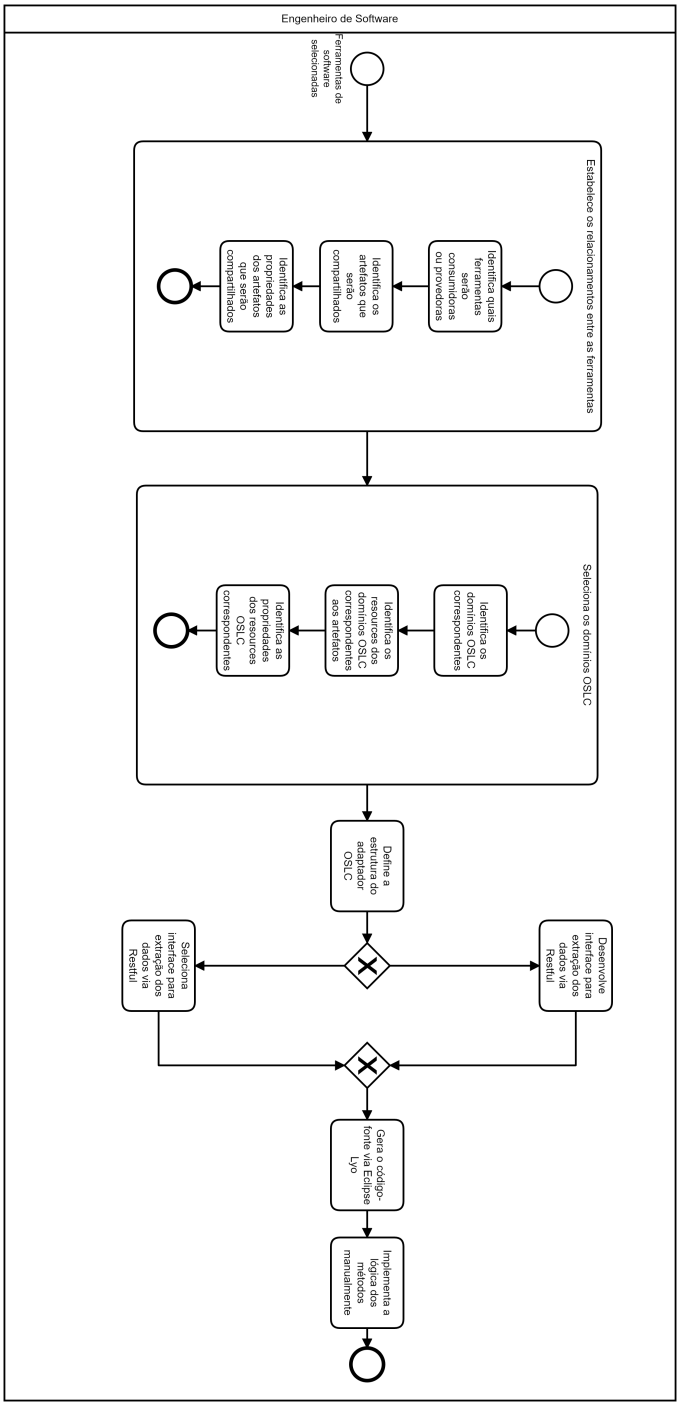

Figura 1. Processo para o desenvolvimento de elementos para a integração de ferramentas em OSLC e Eclipse Lyo

dos artefatos. Por exemplo, o domínio de Gerenciamento de Testes possui os *resources* de Caso de Testes, Plano de Testes, etc. Além disso, esses *resources* possuem propriedades/atributos correspondentes aos artefatos.

Além dos domínios principais, existem os domínios auxiliares que servem para representar outros tipos de dados. Por exemplo, os domínios FOAF e dcterms são padrões para representar dados pessoais na web. Por exemplo, para representar o criador de um

plano de testes, o domínio de gerenciamento de qualidade não possui um *resources* pessoa para tal, sendo necessário referenciar esses domínios auxiliares.

Outra atividade essencial é definir a estrutura dos adaptadores OSLC. Para isso, é necessário seguir a especificação principal (Core) do OSLC. Nessa especificação, todos os artefatos são compartilhados via serviços. Para isso, cada um deles possui um *Uniform Resource Identifier (URI)*. Com isso, por meio de operações Restful é possível acessá-los e manipulá-los através de funções como *Get* e *Post*.

Com base nesse cenário, a fim de organizar o projeto e permitir o acesso a esses serviços, deve-se definir a estrutura desses adaptadores através de catálogos. Esses catálogos disponibilizam URIs para os provedores de serviços e seus serviços. Os provedores de serviços permitem organizar os serviços de forma que seja atrativo aos interesses do projeto. Por exemplo, é possível filtrar os defeitos de uma ferramenta *bugtracker* através dos projetos. Ou ainda, filtrar as tarefas em uma ferramenta de gestão de projetos pelos times.

Com essas atividades planejadas e efetuadas, é possível gerar o código dos adaptadores no Lyo. Entretanto, para que seja possível manipular os dados, primeiro deve-se buscar e estudar uma API RESTful para cada uma das ferramentas envolvidas. Caso não seja possível localizar, deve ser necessário criar sua própria API para a extração dos dados.

Ao final, após validar o projeto e gerar os códigos, percebe-se que apenas os esqueletos dos código foram gerados, sendo necessário implementar a lógica de cada um deles. Para isso, são implementadas as funções REST com base na API selecionada para a ferramenta.

4. Trabalhos Relacionados

A principal abordagem para o desenvolvimento de adaptadores OSLC encontrada na literatura é a geração de códigos-fonte com base em modelos. Esses trabalhos reportam o uso do Eclipse Lyo na indústria como alternativa para desenvolver interfaces OSLC em soluções de integração. Nesse sentido, Marko *et al.* [Marko et al. 2015] reporta o desenvolvimento de um adaptador OSLC para o módulo de gerenciamento de requisitos em uma ferramenta ALM. Nardone *et al.* [Nardone et al. 2020] descreve o desenvolvimento de um ambiente para testes funcionais em nível de sistema para controladores ferroviários integrado com adaptadores OSLC.

Existem trabalhos que buscam integrar ambientes com base no contexto de sistemas críticos de segurança [Biró et al. 2017] [El-khoury et al. 2019] [Gürdür et al. 2018] [Zhang and Møller-Pedersen 2014]. Esses ambientes são conhecidos por possuírem atividades de verificação e validação, além de simulações e do grande volume de dados.

Outros trabalhos focam na avaliação do Eclipse Lyo como solução para a geração semi-automática de código de interfaces OSLC através da representação de metamodelos EMF [Biehl et al. 2012] [El-Khoury et al. 2016]. Esses trabalhos discutem o desenvolvimento de adaptadores para ferramentas de análise e projeto, motivados pela heterogeneidade dos modelos gerados por essas ferramentas.

A Tabela 2 demonstra que todos os trabalhos que vem explorando o uso de OSLC na integração de aplicações para a Engenharia de Software Contínua são limi-

Tabela 2. Propriedades de Integração dos Trabalhos Relacionados

Estudo	Ano	Ferramentas Integradas	Fases de Desenvolvimento Exploradas	Artefatos Compartilhados
Zhang:2014:s6	2014	Ferramenta para modelos Simulink, UML e especificação IEC 61131 e	Análise e Projeto	Modelos
[Biehl et al. 2012]	2014	Matlab Simulink	Análise e Projeto	Modelos
[Marko et al. 2015]	2015	HP Quality Center, ReFine e VeVat	Requisitos e Testes	Requisitos escritos em linguagem natural e semiformal
[Biró et al. 2017]	2017	RTCT (Requiremets Traceability and Consistency cheking Tool)	Requisitos	Requisitos
[Gürdür et al. 2018]	2018	Sistemas ciberfísicos	Análise e Projeto	Modelos
[El-khoury et al. 2019]	2019	Sesammtool, D2RQ	Requisitos,	Análise & Projeto Requisitos e Modelos
[Nardone et al. 2020]	2020	RailModel GUI, IBM Rational Doors, IBM Rational Quality Manager, IOP Test Writer	Requisitos, Testes	Requisitos, Casos de Testes, Planos de Testes, Scripts de Teste, Registros de Execução de Testes, Resultados de Testes
Nosso estudo	2020	Libreoffice/Excel, Redmine e Testlink	Processo do Projeto, Requisitos e Testes	Requisitos Funcionais, Requisitos não Funcionais, Casos de Uso, Tarefas e cronograma, Planos de Teste e Casos de Teste

tados, incluindo o nosso. Os trabalhos existentes conseguem explorar uma parte muito pequena da integração quando se considera todo o ecossistema de software utilizado nas organizações desenvolvedoras de software. Nossa contribuição também tem um limite quando considera-se o restante das ferramentas utilizadas pela DTIC e, como constatou-se, boa parte delas não possuem integração nativa especificada em OSLC. Portanto, é importante salientar que o estudo exploratório atende apenas uma parte do necessário pela DTIC para uma automação completa do processo de desenvolvimento desta organização.

Nosso estudo também é limitado para apenas uma DSL que permite a representação de elementos de integração. Além do Eclipse Lyo, existem outras alternativas para implementar soluções de integração de ferramentas baseadas em OSLC como a DSL TIL [Biehl 2011] e o Guaraná [Frantz et al. 2016] que devem ser consideradas para futuros estudos exploratórios, talvez até mesmo o desenvolvimento de uma DSL própria.

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 com base em modelos. Exemplos práticos e detalhes podem ser encontrados no trabalho [Bruno Marcelo Soares Ferreira 2020] 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 o desenvolvimento de adaptadores OSLC com base em DSLs na literatura.

Os domínios OSLC são disponibilizados em repositórios seguindo a documentação para servir como referência, entretanto são escassos os exemplos e são dispersos em fóruns da comunidade e em alguns repositórios. Essa falta de informação dificulta o trabalho principalmente de pesquisadores iniciantes sobre a integração de ferramentas com OSLC.

Outro ponto importante é de que para modelar diagramas no Eclipse Lyo é necessário o conhecimento da especificação OSLC. Ainda, para tomar decisões sobre a estrutura dos serviços web, é necessário saber o significado de alguns termos apresentados pelo OSLC, como por exemplo provedor e consumidor para modelar a interface de cada adaptador e também sobre as descrições dos domínios OSLC para decidir quais serão utilizados no projeto. Além disso, existem algumas lacunas de informações que surgem na construção de novas soluções OSLC. Algumas vezes a especificação é um pouco vaga ou está distribuída em vários documentos.

O OSLC propõe a representação global de artefatos mantidos por quaisquer ferramentas de software ao longo dos domínios do projeto. Entretanto, nem todos os atributos são cobertos pelos domínios OSLC. Por exemplo é possível manter o relacionamento entre um requisito e um caso de teste através da propriedade `validateBy` no domínio de Gerenciamento de Requisitos mas não é possível acessar quais tarefas estão relacionadas a um determinado caso de teste, pois não existe uma propriedade dedicada a este relacionamento no domínio de Gerenciamento de Qualidade ou Gerenciamento de Configuração e Mudanças.

Dessa forma, o desenvolvimento de adaptadores OSLC não através do Lyo não foi totalmente explorado na área de Engenharia de Software. 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.

Por fim, 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 com base na DSL investigada. Apesar disso, a partir desse 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 Engenharia de Software.

Acknowledgments

This study was partially funded by Pró-Reitoria de Pesquisa (PROPESQ) and AGP, and by FAPERGS, through the ARD project N. 19 / 2551-0001268-3.

Referências

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

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.

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.

Bruno Marcelo Soares Ferreira, F. B. (2020). Investigando a Integração de Ferramentas com OSLC. Monografia (Bacharel em Engenharia de Software), UNIPAMPA (Universidade Federal do Pampa), Alegrete, Brasil.

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*.

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.

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.

Leitner, A., Herbst, B., and Mathijssen, R. (2016). Lessons learned from tool integration with oslc. In Dregvaite, G. and Damasevicius, R., editors, *Information and Software Technologies*, pages 242–254, Cham. Springer International Publishing.

Marko, N., Leitner, A., Herbst, B., and Wallner, A. (2015). Combining xtext and oslc for integrated model-based requirements engineering. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pages 143–150.

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.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media, Berlin, Heidelberg, Dordrecht & New York City.

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.