UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

**Uma abordagem baseada em serviços para facilitar o reúso
de sistema de sistemas e seus constituintes**

Diego de Lima Nascimento

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências
de Computação e Matemática Computacional (PPG-CMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Diego de Lima Nascimento

Uma abordagem baseada em serviços para facilitar o reúso de sistema de sistemas e seus constituintes

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Rosana Teresinha Vaccare Braga

USP – São Carlos
Abril de 2018

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

N244a Nascimento, Diego de Lima
Uma abordagem baseada em serviços para facilitar
o reúso de sistema de sistemas e seus constituintes
/ Diego de Lima Nascimento; orientadora Rosana
Teresinha Vaccare Braga. -- São Carlos, 2018.
119 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2018.

1. Sistema de Sistemas. 2. Reúso. 3. Repositório.
4. Arquitetura Orientada a Serviços. 5.
Interoperabilidade. I. Teresinha Vaccare Braga,
Rosana, orient. II. Título.

Diego de Lima Nascimento

**A service-based approach to facilitate the reuse of system of
systems and its constituents**

Master dissertation submitted to the Institute of
Mathematics and Computer Sciences – ICMC-USP,
in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Profa. Dra. Rosana Teresinha Vaccare Braga

USP – São Carlos
April 2018

“Cada um de nós tem, em algum lugar em seu coração, o sonho de fazer um mundo vivo, um universo. Aqueles de nós que foram treinados como arquitetos têm esse desejo talvez no centro de nossas vidas: que um dia, em algum lugar, de alguma forma, construiremos um edifício maravilhoso, lindo, deslumbrante, um lugar onde as pessoas possam caminhar e sonhar durante séculos.”

(Christopher Alexander)

RESUMO

NASCIMENTO, D. L. **Uma abordagem baseada em serviços para facilitar o reúso de sistema de sistemas e seus constituintes**. 2018. 119 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Sistemas de software estão cada vez mais presentes na sociedade, bem como são componentes imprescindíveis no mundo contemporâneo, uma vez que facilitam e fomentam as praticidades que os cidadãos e desenvolvedores necessitam. Esses sistemas tornam-se muito mais eficientes quando conseguem interoperar com outros, de forma que troquem recursos e não necessitem implementar essas funcionalidades de maneira individual. Os *stakeholders* desses sistemas buscam cada vez mais métodos que facilitem essa comunicação, de um jeito prático e sem perder a independência gerencial de cada um. Desse modo, a classe emergente de Sistema de Sistemas (SoS) ganhou uma atenção considerável, por causa de suas características intrínsecas que promovem a interoperabilidade de sistemas distintos e heterogêneos. No entanto, essa comunicação não é trivial, em razão das diferentes interfaces de cada sistema, além da presença de vários *stakeholders* e a falta de documentação dos sistemas constituintes, dificultando assim, a busca e o uso de cada capacidade. Assim, o objetivo deste trabalho de mestrado é estabelecer uma abordagem baseada em serviços que visa auxiliar os engenheiros de SoS, principalmente na falta de ferramentas de suporte e no esforço para procurar, compreender e adaptar componentes reutilizáveis. A abordagem proposta possui elementos fundamentais que foram utilizados como base para o desenvolvimento de um repositório para engenheiros de SoS e proprietários de sistemas constituintes, de forma organizada e bem documentada, visto que pode ser usado tanto pela comunidade de pesquisa quanto por instituições privadas.

Palavras-chave: Sistema de Sistemas, Reúso, Repositório, Interoperabilidade, Arquitetura Orientada a Serviços.

ABSTRACT

NASCIMENTO, D. L. **A service-based approach to facilitate the reuse of system of systems and its constituents.** 2018. 119 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Software systems are increasingly present in society, and consist in indispensable components in the contemporary world, since they facilitate and foster the practices that citizens and developers need. These systems become much more efficient when they can interoperate with others, so that they exchange resources and do not need to implement these features individually. Stakeholders of these systems increasingly seek methods that facilitate such communication in a practical way and without losing the managerial independence of each one. Therefore, the emerging System Of System class (SoS) gained considerable attention, because of its intrinsic characteristics that promote the interoperability of distinct and heterogeneous systems. However, this communication is not trivial, due to the different interfaces of each system, besides the presence of several stakeholders and the lack of documentation of the constituent systems, making it difficult to search and to use each capacity. Thus, the aim of this master's project is to establish a service-based approach that aims to assist SoS engineers, especially in the lack of supporting tools, in the effort to search, understand and adapt reusable components. The proposed approach has fundamental elements that were used as the basis for the development of a repository for SoS engineers and owners of constituent systems, in an organized and well documented way, since it can be used by both the research community and private institutions.

Keywords: System Of Systems, Reuse, Repository, Interoperability, Service-Oriented Architecture.

LISTA DE FIGURAS

Figura 1 – Metodologia de Pesquisa	22
Figura 2 – Escala de características associadas à definição de SoS	26
Figura 3 – Tipologia de SoS	27
Figura 4 – Características de SoS e Sistemas	29
Figura 5 – SoS como um Time	30
Figura 6 – Visão geral da SOA	35
Figura 7 – Primeira geração de padrões de especificação	37
Figura 8 – Conteúdo da descrição do serviço	39
Figura 9 – Definição Geral de Ativos Reusáveis	40
Figura 10 – Relacionamento entre RAS e Profiles	40
Figura 11 – Seções da Core RAS	41
Figura 12 – Relacionamento das Seções do Core RAS	41
Figura 13 – Modelo da RAS	42
Figura 14 – Visão Geral da MBA4SoSI	43
Figura 15 – Uma abordagem orientada a Serviços para SoS	44
Figura 16 – Visão Geral do SoS-Reúso	46
Figura 17 – Visão Geral do Framework Arrowhead	46
Figura 18 – Visão Geral do Modelo Arquitetural da AIRES	47
Figura 19 – Abordagem Proposta	50
Figura 20 – Modelo Conceitual da abordagem proposta	53
Figura 21 – Casos de Uso da Abordagem	59
Figura 22 – Cenário Típico	60
Figura 23 – Nível de Conhecimento em SoS	61
Figura 24 – Respostas da Pergunta 1a	62
Figura 25 – Respostas da Pergunta 2a	63
Figura 26 – Respostas da Pergunta 3a	63
Figura 27 – Respostas da Pergunta 4a	64
Figura 28 – Respostas da Pergunta 5a	65
Figura 29 – Respostas da Pergunta 6a	66
Figura 30 – Respostas da Pergunta 7a	66
Figura 31 – Pacotes da Implementação do Repositório	71
Figura 32 – Visão Geral do Repositório	72
Figura 33 – Documentação Swagger do Repositório	73

Figura 34 – Exemplo da Documentação	74
Figura 35 – Fluxo de Solicitações	77
Figura 36 – Exemplo de Busca dos Serviços de um CS	77
Figura 37 – Exemplo de Consumo pelo Postman e cURL	79
Figura 38 – Aplicação Gráfica - Login e Cadastro	80
Figura 39 – Aplicação Gráfica - Menu lateral	81
Figura 40 – Aplicação Gráfica - Cadastro de CS	81
Figura 41 – Aplicação Gráfica - Meus Constituintes	82
Figura 42 – Aplicação Gráfica - Busca de Serviços	82
Figura 43 – Distribuição dos Estudos	101

LISTA DE TABELAS

Tabela 1 – Distinções entre Sistema e SoS	27
Tabela 2 – Classificação dos Domínios de Aplicação	31
Tabela 3 – Classificação dos Tipos de Estudos	32
Tabela 4 – Categorização dos Problemas	32
Tabela 5 – Glossário do Modelo Conceitual (em ordem alfabética)	54
Tabela 6 – Glossário dos Casos de Uso	57
Tabela 7 – Sugestões da Pergunta 1b	62
Tabela 8 – Considerações da Pergunta 3b	64
Tabela 9 – Considerações da Pergunta 4b	65
Tabela 10 – Considerações da Pergunta 7b	67
Tabela 11 – HTTPVerb e Códigos de Status do Repositório	72
Tabela 12 – Base de Dados Seleccionadas	99
Tabela 13 – Palavras-chaves e seus Sinónimos	99
Tabela 14 – Classificação dos Domínios de Aplicação (Com ID dos estudos)	100
Tabela 15 – Classificação dos Tipos de Estudos (Com ID dos estudos)	101
Tabela 16 – Estudos Aceitos.	101
Tabela 17 – Categorização dos Problemas (Completa)	102
Tabela 18 – Serviços do Módulo User Managmt	109
Tabela 19 – Serviços do Módulo CS Managmt	111
Tabela 20 – Serviços do Módulo Service Managmt	113
Tabela 21 – Serviços do Módulo SoS Managmt	117

SUMÁRIO

1	Introdução	17
1.1	Contextualização	17
1.2	Motivação e Problema	19
1.3	Objetivos	20
1.4	Metodologia	21
1.5	Organização	21
2	Fundamentação Teórica e Trabalhos Relacionados	23
2.1	Considerações Iniciais	23
2.2	Sistema de Sistemas	23
2.2.1	<i>Conceitos e Definições</i>	23
2.2.2	<i>Características e Classificações de Sistema de Sistemas</i>	25
2.2.3	<i>Sistemas Monolíticos vs Sistema de Sistemas</i>	27
2.2.4	<i>Exemplos de Sistema de Sistemas</i>	29
2.3	Problemas com o Desenvolvimento de SoS - Um Mapeamento Sistemático	30
2.3.1	<i>Planejamento</i>	31
2.3.2	<i>Resultados</i>	31
2.3.2.1	<i>Domínios e Tipos de Estudos</i>	31
2.3.2.2	<i>Conclusões das Questões de Pesquisa</i>	32
2.4	Arquitetura Orientada a Serviços	35
2.4.1	<i>Características da Arquitetura Orientada a Serviços</i>	35
2.4.2	<i>Web Services</i>	36
2.4.2.1	<i>Conceitos Básicos</i>	36
2.4.2.2	<i>Papéis do Web Service</i>	38
2.4.2.3	<i>Estrutura do Web Service</i>	38
2.5	Especificação de Ativos Reusáveis	39
2.6	Arquitetura Baseada em Modelos para Integração de Sistema de Sistemas	41
2.7	Trabalhos Relacionados	43
2.8	Considerações Finais	48
3	Uma Abordagem Orientada a Serviços para Facilitar o Reúso no Contexto de SoS	49
3.1	Considerações Iniciais	49

3.2	Abordagem Proposta	50
3.2.1	<i>Conceitos Iniciais</i>	50
3.2.2	<i>Visão Geral e Modelo Conceitual da Abordagem</i>	51
3.2.3	<i>Diretrizes de uso</i>	55
3.2.4	<i>Cenário de Uso</i>	58
3.3	Avaliação da Abordagem	60
3.3.1	<i>Questionário</i>	60
3.3.2	<i>Análise dos Resultados</i>	61
3.4	Considerações Finais	66
4	Detalhes da Implementação do Repositório da Abordagem	69
4.1	Considerações Iniciais	69
4.2	Detalhamento do Repositório	70
4.2.1	<i>Implementação</i>	70
4.2.2	<i>Documentação</i>	72
4.2.3	<i>Módulo User Managmt</i>	75
4.2.4	<i>Módulo CS, Services e SoS Managmt</i>	76
4.3	Utilização do Repositório	78
4.3.1	<i>APIs e Linha de Comandos</i>	78
4.3.2	<i>Camada de Interface Gráfica com o Usuário</i>	80
4.4	Considerações Finais	83
5	Considerações Finais e Trabalhos Futuros	85
5.1	Visão Geral da Pesquisa	85
5.2	Contribuições	86
5.3	Dificuldades e Limitações	86
5.4	Trabalhos Futuros	87
	Referências	89
	APÊNDICE A Mapeamento Sistemático	97
	A.1 Planejamento	97
	A.2 Condução	99
	A.2.1 <i>Sumarização dos Dados</i>	100
	A.3 Ameaças à Validade	102
	APÊNDICE B Questionário	105
	APÊNDICE C Serviços do Módulo User Managmt	109
	APÊNDICE D Serviços do Módulo CS Managmt	111
	APÊNDICE E Serviços do Módulo Service Managmt	113
	APÊNDICE F Serviços do Módulo SoS Managmt	117

INTRODUÇÃO

1.1 Contextualização

Os sistemas intensivos em software estão cada vez mais onipresentes e complexos, surgindo da cooperação de vários sistemas operacionalmente independentes, distribuídos geograficamente, e até mesmo desenvolvidos para diversas tecnologias e diferentes plataformas. Essa cooperação tem sido cada vez mais necessária para resolver missões maiores e mais complexas, que não poderiam ser resolvidas por sistemas que trabalhassem separadamente (NAKAGAWA *et al.*, 2015).

Frente a esse crescimento de tamanho e complexidade, os sistemas de software estão se movendo em direção a Sistema de Sistemas (SoS - do inglês *System of Systems*). Essa classe de sistemas surgiu com o propósito de resolver problemas de larga escala, sendo interoperabilidade de sistemas monolíticos uma de suas principais características. Além disso, pode existir um conjunto grande de *stakeholders* envolvidos, incluindo proprietários e operadores dos sistemas constituintes, resultando em uma necessidade de empregar métodos e ferramentas que deem suporte ao trabalho colaborativo, desde a obtenção de requisitos até os testes e manutenção (NIELSEN *et al.*, 2015). Vierhauser, Rabiser e Grünbacher (2016) também acrescenta que os SoS's (tipicamente) não são planejados e projetados, mas obtidos de maneira progressiva, muitas vezes como resultado de décadas de desenvolvimento por equipes independentes.

Ademais, as características intrínsecas de SoS podem ser correlacionadas com diferentes desafios e cenários de desenvolvimento. De acordo com Lane (2013), Friedman (2005), os SoS's estão presentes em: (i) casas: hoje em dia, casas são conectadas com dispositivos móveis e sistemas de segurança, bem como casas inteligentes que contém inúmeros sistemas integrados e personalizados; (ii) companhias: as áreas de negócios podem conter diversos sistemas específicos interconectados, como monitoramento, entrada-saída, gerenciamento de clientes, recursos humanos, pagamento, entre outros; (iii) emergência: sistemas de emergência são variados e podem

depende de diferentes recursos. Por exemplo, um incêndio pode exigir a colaboração de vários setores diferentes, como corpos de bombeiros, departamento de polícia e controle de tráfego; (iv) defesa: sistemas de defesa necessitam de comunicação para resolver missões complexas. Um cenário comum é o envolvimento de outras organizações, como exército, marinha e força aérea; e (v) exploração: sistemas de exploração espacial também envolvem vários setores e necessitam da comunicação de diferentes dispositivos (GROTZINGER; VASAVADA; RUSSELL, 2013).

Atrelado a esse crescimento e necessidade, encontram-se diversas técnicas que foram desenvolvidas para dar suporte ao reúso de software. No contexto de SoS, o reúso de software desempenha um papel muito importante, por causa da necessidade de alcançar um sinergismo entre os inúmeros constituintes que fazem parte dessa rede. Segundo Sommerville (2011), o reúso é uma resposta para a exigência de redução de custo e manutenção, entregas mais rápidas e com mais qualidade. Por esses motivos, o desenvolvimento baseado em reúso tem aumentado significativamente, sendo que atualmente podem ser encontrados variados tipos de reúso (com baixo custo), como código aberto e padrões que oferecem reúso geral a uma variedade de aplicações.

No âmbito de SoS, um dos padrões arquiteturais mais comuns é a arquitetura orientada a serviços (SOA), por causa de sua praticidade em interoperar os sistemas constituintes que compõem o SoS. Diversas pesquisas (MADNI; SIEVERS, 2014; VARGA *et al.*, 2017; LEWIS *et al.*, 2011; KEWLEY *et al.*, 2008; COLOMBO; BANGEMANN; KARNOUSKOS, 2013; HALEY; NUSEIBEH, 2008; KAZMAN *et al.*, 2013; GUTIERREZ-GARCIA; RAMOS-CORCHADO; KONING, 2009; KRUEGER *et al.*, 2006) usam o princípio e a praticidade dessa arquitetura para essa classe de sistemas.

A abordagem orientada a serviços tornou-se uma maneira prática de desenvolvimento de software para implementar sistemas distribuídos e sem acoplamento. Esse estilo de arquitetura oferece recursos como padronização, independência de plataforma, interfaces bem definidas e suporte a ferramentas que permitem a integração de sistemas legados (LEWIS *et al.*, 2011).

Segundo Kaur *et al.* (2013), SOA fornece uma plataforma excelente para desenvolvimento de sistemas por meio de um conjunto de serviços oferecidos, incluindo sistemas legados. Além disso, o estilo arquitetural também oferece a capacidade de estender o comportamento do sistema em uma abordagem para SoS. Em suma, é provável que um SoS bem-sucedido tenha baixo acoplamento, mas seja bem integrado, o que é uma característica de um sistema baseado em SOA.

De acordo com Lewis *et al.* (2011), em uma abordagem orientada a serviços para SoS destacam-se as seguintes características: (i) os serviços oferecem funcionalidade de negócios reutilizáveis por meio de interfaces padronizadas e bem definidas para promover a interoperabilidade; (ii) os consumidores de serviços exploram as funcionalidades dos serviços disponíveis; (iii) a interface e a implementação do serviço são claramente separadas para promover a independência da plataforma; (iv) uma infraestrutura SOA permite descoberta, composição e

invocação de serviços para promover o baixo acoplamento entre consumidores e serviços; e (v) os protocolos são predominantemente (mas não exclusivamente) trocas de documentos baseadas em mensagens.

1.2 Motivação e Problema

A comunicação de sistemas é um desafio que está recebendo cada vez mais atenção da academia e indústria nos últimos anos. Métodos, arquiteturas e frameworks estão sendo analisados e remodelados para abordar questões de interoperabilidade entre sistemas heterogêneos (VARGA *et al.*, 2017). De fato, a maioria da população mundial vive em territórios urbanos (NATIONS, 2009), fazendo com que informações sejam recursos essenciais para a rotina da sociedade. Esses e outros fatores fazem com que setores governamentais e empresas privadas busquem e forneçam maneiras de melhorar a vida dos cidadãos e clientes. Desse modo, o interesse em criar sistemas complexos para fornecer recursos e uma melhor infraestrutura cresceu ainda mais, uma vez que sistemas verticais fazem parte da sociedade há muito tempo. Isso torna o compartilhamento de recursos e a interoperabilidade dos sistemas não só benéficos para alguns contextos, mas uma necessidade.

Na perspectiva de SoS, alcançar a interoperabilidade de constituintes que não são projetados para funcionar juntos ainda é complexa, exigindo uma carga de trabalho extremamente alta (MADNI; SIEVERS, 2014). Segundo Linthicum (2003), a simples comunicação (troca de dados) no nível de serviço e de informação é uma abordagem estratégica denominada integração de aplicações. Apesar de parecer simples, a integração de aplicações envolve uma combinação de problemas, uma vez que cada organização ou empresa tem seus próprios desafios de integração que devem ser considerados. Dessa forma, é quase impossível encontrar uma tecnologia que solucione todos os casos de integração de aplicações.

No contexto de SoS isso fica ainda mais complicado, por causa da dependência de diversos *stakeholders* que precisam se adaptar continuamente, principalmente em mudanças nos níveis de autonomia e nas missões do SoS. É importante enfatizar que isso resulta em dificuldades para os engenheiros responsáveis e, conseqüentemente, aumentam a possibilidade de erro, ressaltando que os seres humanos tendem a perder o desempenho em atividades paralelas, especialmente quando o contexto muda frequentemente (MADNI, 2010; MADNI, 2011).

Como já mencionado, a SOA está sendo utilizada como uma solução para a interoperabilidade desses constituintes, principalmente porque está havendo uma mudança da integração orientada à informação (comunicação por meio de APIs proprietárias projetadas com o objetivo de integrar os bancos de dados) para a orientada a serviços (compartilhamento de informação por meio de lógica de negócios e métodos através do uso de tecnologias como *Web Services*) (LINTHICUM, 2003). Além disso, SOA é uma das principais abordagens que apoiam o reúso de software.

As vantagens que o reúso de software proporciona para o desenvolvimento de sistemas são evidentes, ressaltando principalmente o tempo de mercado e diminuição no custo do projeto (SOMMERVILLE, 2011). Em contrapartida, existem alguns problemas comuns que podem acontecer em qualquer umas das unidades de software reusáveis (reúso de aplicações, reúso de componentes e reúso de objetos e funções). Mais especificamente, há dois problemas que fazem parte do contexto deste trabalho de mestrado, que são a falta de ferramentas de suporte e o esforço para encontrar, compreender e adaptar componentes reusáveis.

Diante disso, a motivação principal desta pesquisa partiu da conclusão de que a interoperabilidade de sistemas diferentes para formar um SoS não é trivial, devido aos vários fatores mencionados, além da falta de documentação, roteiros e controle/gerenciamento de *stakeholders*. De fato, um mapeamento sistemático (MS) (NASCIMENTO; BRAGA, 2018) realizado pelo autor e pela orientadora deste trabalho, procurou identificar os desafios/problemas que ocorrem durante o desenvolvimento de SoS. Por meio da sumarização dos resultados do MS, foi possível identificar nove categorias de problemas: integração, gerenciamento, documentação, evolução e manutenção, arquitetura e modelagem, requisitos e qualidade, teste, reconfiguração dinâmica e monitoramento, e comportamento. A maioria dos estudos apresentaram desafios/problemas de integração e gerenciamento, devido à dificuldade de interoperar sistemas existentes, à falta de engenheiros qualificados e ao controle de múltiplos *stakeholders*. Outro ponto interessante foi o número de projetos fechados que não estavam disponíveis para a comunidade de pesquisa. Essa falta de disponibilidade dificulta a entrada de novos engenheiros, bem como a evolução da área de SoS de forma geral.

Por fim, é importante enfatizar outra motivação que é contribuir com um projeto de pesquisa coordenado pela orientadora deste trabalho. O projeto, denominado MBA4SoSI (do inglês - *Model-Based Architecture for System-Of-Systems Integration*), visa prover uma solução arquitetural para a integração de sistemas em SoS, utilizando SOA como forma de comunicação entre os constituintes.

1.3 Objetivos

Este projeto de mestrado teve como objetivo principal estabelecer uma solução para os desafios de reúso de capacidades no contexto de SoS, visando facilitar a interoperabilidade entre os sistemas constituintes, por meio de uma abordagem orientada a serviços na qual um repositório é disponibilizado para engenheiros de SoS, proprietários de sistemas constituintes e desenvolvedores de software em geral. A ideia é fornecer meios para compartilhar uma documentação bem estruturada sobre SoS existentes, seus constituintes e serviços disponíveis, fomentando de forma geral o reúso.

A proposta é justificada por ser um passo importante rumo à solução para os desafios de interoperabilidade, falta de documentação e reúso de capacidades que podem estar presentes

nos serviços oferecidos por sistemas constituintes e no SoS. Entende-se que, se a documentação necessária sobre o SoS e seus constituintes for fornecida aos engenheiros de SoS, a interoperabilidade dos constituintes se tornará mais simples por haver melhor entendimento sobre as capacidades fornecidas por cada um deles. Além disso, a curva de experiência necessária para trabalhar com esses tipos de sistemas pode ser menor quando sua documentação e diretrizes de utilização estiverem presentes para novos engenheiros usarem. O conhecimento de especialistas poderá ser incorporado ao repositório, facilitando seu reúso por outros engenheiros de SoS.

A abordagem proposta engloba um modelo conceitual, o repositório, a definição dos atores envolvidos e seus papéis, bem como diretrizes para o uso do repositório, o qual foi disponibilizado em forma de código aberto para possíveis colaboradores. Foi implementada também uma aplicação para consumir os serviços de acesso ao repositório, visando auxiliar engenheiros que tenham interesse em consumir os serviços por meio de uma aplicação gráfica.

1.4 Metodologia

Para a condução deste projeto de mestrado, um MS foi realizado com intuito de verificar os desafios que pesquisadores enfrentam durante o desenvolvimento de SoS. Dessa forma, foram identificados os principais conceitos, aplicações e soluções utilizadas no contexto de SoS, além da sumarização dos resultados em categorias que apresentam uma visão geral sobre os desafios.

Com a familiaridade com o estado da arte e os desafios, observou-se a necessidade de propor uma solução para o problema de reúso no desenvolvimento de SoS, por meio de uma abordagem que facilita a documentação e reúso dos *assets* de SoS (serviços e constituintes). Depois de definir os objetivos, iniciou-se a construção de um modelo com base no escopo do trabalho, destacando a composição dos casos de uso e os atores que deveriam estar presentes no repositório. Paralelamente, foi realizada uma avaliação conceitual com especialistas (pesquisadores e desenvolvedores) de SoS sobre o modelo, atores e escopo da pesquisa. Assim, o refinamento foi feito de acordo a necessidade e as considerações dos sujeitos.

Posteriormente, foi iniciado o desenvolvimento do repositório baseado em serviços. Por fim, foi implementada uma aplicação que auxilia e facilita o uso do repositório. O principal objetivo dessa aplicação é fornecer uma interface gráfica e elucidar as diferentes formas que os engenheiros interessados podem consumir os serviços do repositório. A Figura 1 apresenta os passos realizados para a conclusão deste mestrado.

1.5 Organização

O restante da dissertação está organizado da seguinte forma: No Capítulo 2 apresenta-se a fundamentação teórica necessária sobre SoS e SOA, enfatizando as suas terminologias e os

Figura 1 – Metodologia de Pesquisa

Fonte: Elaborada pelo autor.

conceitos básicos, incluindo as diferentes definições, classificações e comparações com sistemas tradicionais. Além disso, também são apresentados o MS e os trabalhos relacionados.

No Capítulo 3 é apresentada a abordagem proposta. Nesse capítulo, são relatados quem são os atores e suas responsabilidades, bem como a validação conceitual por especialistas na área de SoS.

No Capítulo 4 são apresentados todos os passos e tecnologias utilizadas durante o desenvolvimento do repositório que dá suporte à abordagem proposta. Além disso, é disponibilizada e apresentada uma forma concreta de acesso ao repositório, por meio de uma aplicação gráfica desenvolvida com o propósito de facilitar o uso.

Por fim, no Capítulo 5 são apresentadas as considerações finais da pesquisa, assim como suas principais contribuições e trabalhos futuros.

FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

2.1 Considerações Iniciais

Uma parte fundamental desta pesquisa de mestrado é promover o reúso de software durante atividades de composição de diversos sistemas monolíticos com intuito de formar um SoS. Para facilitar o entendimento desta pesquisa, este capítulo visa apresentar os conceitos principais de SoS, bem como uma solução (SOA) utilizada na interoperabilidade dos sistemas que formam essa classe. Assim, na Seção 2.2 são apresentados os conceitos, definições, características, classificações e exemplos de SoS, além de uma breve comparação com sistemas monolíticos. Na Seção 2.3 é apresentada uma síntese dos resultados do mapeamento sistemático realizado. Na Seção 2.4 é definida a Arquitetura Orientada a Serviços, assim como suas características e formas de disponibilizar os serviços pela internet (*Web Services*). Na Seção 2.5 é apresentada a RAS (*Reusable Asset Specification* - Especificação de Ativos Reusáveis) que, por sua vez, foi usada como base para aperfeiçoar a busca e compartilhamento do repositório proposto. Na Seção 2.6 é detalhada uma solução arquitetural para integração de SoS com base em SOA, na qual esta proposta está contida. Na Seção 2.7 são apresentados alguns trabalhos que utilizam SOA para interoperar os sistemas de um SoS, bem como repositórios que utilizam principalmente a RAS como base. Por fim, na Seção 2.8 são apresentadas as considerações finais do capítulo.

2.2 Sistema de Sistemas

2.2.1 Conceitos e Definições

Atualmente, existe uma grande variedade de sistemas desenvolvidos por diferentes empresas e organizações governamentais, com o objetivo de facilitar e agilizar a vida dos usuários

no seu cotidiano. Cada sistema tem sua peculiaridade, todavia, todos possuem o propósito de prover um resultado satisfatório, isto é, cumprir o seu objetivo. Alguns desses sistemas estão dispersos geograficamente, entretanto, não é difícil perceber que grande parte têm um objetivo semelhante, logo suas funcionalidades podem se completar. Nesse contexto, apresenta-se uma classe de sistemas que contém ao menos uma ordem de complexidade maior que seus sistemas constituintes (CS – do inglês *Constituent System*), denominada Sistema de Sistemas (SAGE; CUPPAN, 2001).

O termo SoS está aparecendo cada vez mais na literatura, devido à busca constante por soluções na integração de sistemas. Simultaneamente, motivos como compartilhamento de capacidades, características organizacionais e operacionais de sistemas singulares, fomentam ainda mais esse crescimento. Em uma definição simples, o SoS pode ser visto como a integração de múltiplos sistemas com a finalidade de resolver problemas em larga escala (STUTZKE, 2005). Contudo, essa classe de sistemas é discutida há muito tempo e várias definições começaram a aparecer na literatura por causa disso. A seguir, são apresentadas as definições mais conhecidas:

- Um SoS é um conjunto de componentes integrados que individualmente podem ser considerados como sistemas, mas possuem duas propriedades adicionais: independência operacional e gerencial (MAIER, 1998);
- A engenharia de SoS (SoSE – do inglês *System Of Systems Engineering*) envolve a integração de sistemas em SoS, que por fim, contribuem para evolução da infraestrutura social (LUKASIK, 1998);
- Um SoS é um conjunto de diferentes sistemas tão conectados e/ou relacionados que produzem um sinergismo único que não pode ser alcançado pelos sistemas individuais (KRYGIEL, 1999);
- A integração de SoS é um método que visa o desenvolvimento, integração, interoperabilidade e otimização de sistemas, com o objetivo de melhorar o desempenho em futuros cenários em campos de batalha (PEI, 2000);
- Um SoS é formado quando há a presença da maioria das cinco características: independência operacional, independência gerencial, distribuição geográfica, comportamento emergente e desenvolvimento evolutivo (SAGE; CUPPAN, 2001);
- Um SoS pode ser definido vagamente como uma combinação de sistemas, a fim de satisfazer algum tipo de capacidade, com o fato adicional de que os sistemas integrados devem ter independência operacional e gerencial (AUTRAN *et al.*, 2008);
- Um SoS pode ser definido como qualquer sistema grande e complexo, de evolução dinâmica e distribuído fisicamente, composto de sistemas preexistentes, heterogêneos,

autônomos e independentemente governados, de forma que apresente quantidades significativas de comportamento e características emergentes inesperadas (FIRESMITH, 2010); e

- Por fim, a definição adotada por este trabalho. Um SoS é um conjunto ou arranjo de sistemas que resulta quando sistemas independentes e úteis são integrados em um sistema maior que possui uma capacidade específica (DOD, 2008).

2.2.2 Características e Classificações de Sistema de Sistemas

Mesmo com a capacidade de interoperar CS's distintos, o SoS permite que os CS's operem de maneira isolada e realizem suas funcionalidades normalmente. Dessa maneira, os engenheiros conseguem evoluir e gerenciar seus CS's comumente, isto é, é possível realizar funcionalidades dentro e fora do SoS. Maier (1998), Jamshidi (2008) retratam que um SoS apresenta cinco características principais:

- Independência operacional: um CS é parte do propósito do SoS, mas ainda assim, deve operar de forma independente com sua missão e/ou objetivo específico se o SoS for desfeito;
- Independência gerencial: apesar da colaboração com outros CS's do SoS, cada CS é gerenciado de forma independente;
- Distribuição geográfica: cada CS pode ser executado em diferentes lugares, desde que os elementos pertencentes ao SoS, principalmente, colaborem e alterem os recursos;
- Comportamento emergente: com a colaboração dos CS's envolvidos, um sinergismo é alcançado e uma nova capacidade única aparece. Dessa forma, esse comportamento não pode ser alcançado e/ou atribuído a um CS individualmente; e
- Desenvolvimento evolucionário: o SoS nunca está completo, isto é, o desenvolvimento evolucionário define que os objetivos e funcionalidades estão em constante mudança, sendo assim, podem ser adicionados, modificados ou removidos.

Outras maneiras de visualizar um SoS também são apresentadas na literatura. De acordo com Firesmith (2010), é possível separar as características que descrevem um SoS em nível de sistema e subsistema (Figura 2). Na figura, são apresentadas as características no nível de sistema, responsáveis por descrever um sistema como um todo, e de subsistema, cujo intuito é descrever um sistema em termos das características de seus subsistemas. Cada característica está associada a uma escala, enfatizando que qualquer sistema pode aparecer em diferentes pontos dela. Assim, a escala pode ser visualizada como medidores que medem o grau em que o sistema se relaciona com a característica associada.

Figura 2 – Escala de características associadas à definição de SoS

Fonte: Firesmith (2010).

Como já foi mencionado, existem diversas discussões sobre as definições de SoS. Por outro lado, em relação à classificação de SoS, eles são comumente distinguidos pela presença ou não de uma autoridade central, nível de controle gerencial e colaboração dos CS's. Os quatro tipos mais utilizados são explicados a seguir (MAIER, 1998; DAHMANN; BALDWIN, 2008; LANE, 2013):

- **Reconhecido:** o SoS tem objetivos reconhecidos, um gerente designado e recursos para o SoS. No entanto, os CS's mantêm sua independência, objetivos, financiamento e abordagens de desenvolvimento;
- **Direcionado:** um SoS integrado é desenvolvido e gerenciado para atingir objetivos específicos. Os CS's operam de forma independente, mas são controlados por uma autoridade central e sua evolução é controlada;
- **Colaborativo:** o SoS possui uma autoridade central, porém ela não tem poder coercivo para executar o sistema. Os CS's colaboram voluntariamente para cumprir sua missão. Dessa forma, não há equipe de engenharia de SoS para orientar ou gerenciar as atividades relacionadas ao SoS e seus CS's; e

- Virtual: o SoS não possui uma autoridade central. O SoS também não tem um propósito definido. Por fim, os CS's podem não ser reconhecidos.

[Madni e Sievers \(2014\)](#) também apresentam essa classificação por meio de uma tipologia com exemplos de cada tipo (Figura 3).

Figura 3 – Tipologia de SoS

- Virtual SoS
 - no central management authority
 - no centrally agreed-upon purpose
 - systems can enter/exit dynamically based on mission requirements
 - potential for emergence of large scale system behavior (which may be desirable)
 - relies on relatively hidden mechanisms for maintenance
 - examples: Stock market, national economies
- Collaborative SoS
 - component systems interact more or less voluntarily to fulfill agreed upon central purposes
 - e.g., internet is a collaborative SoS; the Internet Engineering Task Force defines but can't enforce standards
 - central players collectively decide how to provide/deny service (some means to enforce and maintain standards)
 - examples: World-wide web, multi-robot systems
- Acknowledged SoS
 - has recognized objectives, a designated manager, and resources; however,....
 - constituent systems retain independent ownership, objectives, funding, development and sustainment
 - changes in the system are based on collaboration between the SoS and the system
 - example: DoD's Ballistic Missile Defense System
- Directed SoS
 - integrated SoS is built and managed to fulfill specific purposes
 - centrally managed during long-term operation to fulfill purposes and any new ones set by system owners
 - component systems maintain ability to operate independently, but their normal operational mode is subordinated to the central managed purpose
 - example: JPL's Deep Space Network

Fonte: [Madni e Sievers \(2014\)](#).

2.2.3 Sistemas Monolíticos vs Sistema de Sistemas

Um SoS geralmente é direcionado para problemas complexos de grande escala. Então, há algumas diferenças entre um sistema tradicional e um SoS, visto que algumas dessas características permitem a um SoS lidar com problemas com mais facilidade do que em um sistema tradicional ([XIAO et al., 2011](#)). Por exemplo, [Lowe e Chen \(2008\)](#) apresentam de maneira simples (Figura 4) como um SoS exibe algumas características operacionais de forma diferente dos sistemas monolíticos.

De fato, o SoS difere de outros sistemas tradicionais por causa da soma de suas partes, uma vez que os componentes atuam como sistemas autônomos, por terem os seus próprios *stakeholders* e apresentarem diversidades. De acordo com [Boardman e Sauser \(2006\)](#), a diferença entre um sistema e um SoS reside na composição. Nessa perspectiva, são apresentadas na Tabela 1 algumas distinções.

Tabela 1 – Distinções entre Sistema e SoS

Característica	SoS	Sistema
Autonomia (BOARDMAN; SAUSER, 2006)	Autonomia existe nos CS's com o propósito de alcançar o objetivo do SoS.	Autonomia é concebida para todo o sistema.

Pertencer (BOARDMAN; SAUSER, 2006)	CS's funcionam colaborativamente com o objetivo de satisfazer um propósito comum.	O "pertencer" dos elementos do sistema não é determinado por eles mesmos, mas por sua herança, função designada e o design específico do sistema.
Conectividade (BOARDMAN; SAUSER, 2006)	É possível alcançar o sinergismo pela rede altamente dinâmica e distribuída.	Design de alta qualidade com alta conectividade escondida nos elementos; e conectividade mínima entre subsistemas principais.
Complexidade (BOARDMAN; SAUSER, 2006)	As dinâmicas intra-sistema, inter-sistemas e as diversas conexões entre os CS's fazem o SoS extremamente complexo.	As complexidades vêm da dinâmica intra-sistema e do número limitado de combinações das partes do sistema.
Diversidade (BOARDMAN; SAUSER, 2006)	Os CS's são heterogêneos e auto-suficientes e, estão abertos para melhorias por evolução e adaptação.	A diversidade de elementos do sistema é envolvida por diferentes módulos funcionais com o objetivo de reduzir a complexidade para o próximo nível hierárquico.
Envolvimento Humano (LIU, 2011)	O envolvimento humano acontece durante a evolução e operação do SoS e é determinado pela dinâmica dos inter-sistemas.	O envolvimento humano é planejado e predefinido.
Interoperabilidade e Sinergismo (SECURITY, 2004)	Os CS's comportam-se como sistemas autônomos. Entretanto, é possível ter interoperabilidade entre esses sistemas, ajustando as estratégias e necessidades de acordo com o objetivo individual, assim como, o do SoS.	Os elementos e componentes do sistema têm funções predefinidas. Assim, as partes do sistema não podem empregar outros elementos e dados para fazer decisões independentes e mudar o sistema de acordo.
Alocação e utilização de recursos (LIU, 2011)	O CS deve ajustar sua projeção para o recurso conforme necessário. A utilização de recursos no SoS exerce várias mudanças e interoperabilidade complexas para alcançar os objetivos do SoS e do CS de forma econômica.	A alocação de recursos é baseada na função de cada sistema. É possível fazer um orçamento relativamente preciso para os recursos necessários para o design e a implantação de um sistema.
Estrutura ou Arquitetura (LIU, 2011)	O CS e as diversas conexões do SoS formam uma arquitetura de sistema dinâmico, que alteram-se de forma correspondente para alcançar o propósito do SoS (e seus CS's) de forma econômica.	As estruturas fundamentais do sistema são planejadas. Dessa forma, as mudanças que podem acontecer no comportamento são determinadas por esse planejamento.
Intervenção política (LIU, 2011)	As intervenções de políticas geralmente ocorrem em um ou vários CS's, não no nível do SoS. Por isso, afetam sensivelmente o curso de implementação das outras políticas em diferentes sistemas, de maneira que torne outras políticas inválidas.	A intervenção da política geralmente altera a estrutura do sistema e consequentemente o comportamento.
Incerteza (LIU, 2011)	As incertezas podem ocorrer de dois lados. Primeiro, a partir da dinâmica dos CS's individuais; e a partir da dinâmica do contexto e das numerosas interações por meio de diferentes conexões entre diferentes sistemas.	As incertezas podem variar de acordo com as decisões humanas, intervenções inesperadas e dinâmicas do sistema.
Emergência (BOARDMAN; SAUSER, 2006; LIU, 2011)	O SoS é uma arquitetura dinâmica que gera ambientes de gerenciamento de emergências, por meio do desenvolvimento de um evento emergente para prevenir, mitigar e eliminar comportamentos desfavoráveis.	O princípio do design de um sistema, é fornecer uma permissão para lidar com eventos previsíveis, considerando e testando todos os cenários.

Fonte: Adaptada de Liu (2011).

Por outro lado, Firesmith (2010) retrata que quase todo sistema é um SoS. Segundo a sua definição, *“um sistema é um grupo integrado e coeso de componentes interagindo ou interdependentes, cujo objetivo é colaborar com o intuito de fornecer um comportamento e característica que: (1) são necessárias para atender às necessidades e desejos válidos dos stakeholders; e*

Figura 4 – Características de SoS e Sistemas

<u>SoS</u>	<u>System</u>
<ul style="list-style-type: none"> ✓ Loosely coupled systems <ul style="list-style-type: none"> ⇒ Each contributes to SoS behavior ⇒ Each an operating system in its own right ⇒ SoS boundaries are defined, yet fuzzy ✓ Unpredictable, complex behavior <ul style="list-style-type: none"> ⇒ Spontaneous emergent behavior ⇒ Complex human (system) interactions ⇒ Opportunistic, unanticipated evolution 	<ul style="list-style-type: none"> ✓ Tightly coupled components <ul style="list-style-type: none"> ⇒ Each contributes to system behavior ⇒ None operate independently ⇒ System boundaries are concise & defined ✓ Well-understood behavior <ul style="list-style-type: none"> ⇒ Emergent behavior we understand ⇒ Human (operator) interfaces ⇒ System evolution expected and planned

Fonte: Lowe e Chen (2008).

(2) *não podem ser fornecidos separadamente pelos componentes individuais*” (FIRESMITH *et al.*, 2008). Considerado-se a segunda parte da definição, um sistema é mais do que a soma de suas partes, porque ele fornece comportamentos ou características emergentes benéficas que não estão presentes em seus componentes individuais de forma independente. Dessa forma, um carro na garagem é um sistema, mas todos os componentes dele jogados no chão não.

De qualquer maneira, é importante reconhecer em qual ambiente pretende-se desenvolver um sistema ou o SoS, pois é mais simples aplicar os princípios da engenharia de software no ambiente adequado (DOD, 2008).

2.2.4 Exemplos de Sistema de Sistemas

O SoS pode variar de acordo com as diferentes situações e contextos do ciclo de vida de uma organização. Nessa perspectiva, pode-se utilizar um time como exemplo (Figura 5), onde cada integrante é considerado um SoS, montados a partir dos vários componentes físicos separados, mas interligados e interrelacionados de maneira exclusivamente funcional como um todo, além de vários subsistemas precisarem de foco e atendimento crítico quando necessário. Ademais, quando uma pessoa faz parte de uma equipe, ela se torna um componente de um sistema maior. Dessa forma, se estender esse caso para uma equipe esportiva, consequentemente ela vai se tornar parte de uma liga que, por sua vez, passa a fazer parte de um mundial (CHOWN, 2017). Assim, uma equipe precisa trabalhar e cooperar para contribuir no avanço de forma geral.

De acordo com essa visão e com os conceitos já mencionados, nota-se que é possível utilizar e exemplificar o SoS em vários domínios. Segundo Maier (1998), são exemplos de SoS: “(i) um sistema integrado de defesa aérea é composto por uma rede geograficamente dispersa de elementos semiautônomos. Estes incluem radares de vigilância, sistemas de vigilância passiva, baterias de lançamento de mísseis, locais de controle, radares de vigilância, rastreamento aéreo, aviões de combate e artilharia antiaérea. Todas as unidades estão conectadas por uma rede de comunicação com comando e controle aplicado em centros locais, regionais e nacionais; (ii) a Internet é um exemplo de um SoS do tipo colaborativo. Seus elementos são redes e *sites* que operam de maneira colaborativa, oferecendo sempre novos serviços, mas sem precisar alterar os

Figura 5 – SoS como um Time

Fonte: [Chown \(2017\)](#).

protocolos subjacentes. Algumas dessas redes de componentes também podem ser compostas por outras subredes. Os *sites* da Internet trocam informações colaborativamente usando protocolos documentados. A adesão ao protocolo é amplamente voluntária, sem autoridade central com poder coercivo. O poder coercivo emerge por meio de acordos entre os principais *sites* para bloquear o tráfego daqueles que apresentaram um comportamento não desejado; e (iii) sistemas de transporte inteligentes cobrem uma ampla gama de potenciais aplicações de informação e tecnologia para redes rodoviárias e de transporte. Estes variam desde melhoria na comunicação entre veículos de serviço público até as rodovias automatizadas com carros robotizados.”.

2.3 Problemas com o Desenvolvimento de SoS - Um Mapeamento Sistemático

Diante da temática de SoS, que possui diversas definições e ainda pode ser considerada recente no âmbito da computação, principalmente na área engenharia de software, surge a motivação de identificar os principais problemas enfrentados durante o seu desenvolvimento, mais especificamente obter uma visão dos problemas e/ou desafios.

Uma forma de conseguir isto é realizando um MS, que é um método específico de revisão da literatura. Um MS permite identificar e quantificar estudos primários relevantes para responder questões de pesquisa em uma área específica de conhecimento ([PETERSEN et al., 2008](#)). Para estabelecer um processo de revisão da literatura sistemático e que possa ser reproduzido por outros interessados, foram seguidas as diretrizes estabelecidas por [Kitchenham e Charters \(2007\)](#).

Assim, durante este mestrado foi conduzido um MS para averiguar como os pesquisadores estão trabalhando com SoS, de tal forma que seja possível identificar os ambientes, ferramentas, soluções, possíveis melhorias para serem consideradas e/ou desafios em aberto. No decorrer desta Seção, são apresentados os tópicos principais deste MS, mais especificamente as questões de pesquisa (RQ - *Research Question*) e os resultados. No Apêndice A é feito o detalhamento completo do MS.

2.3.1 Planejamento

Esse MS foi conduzido com o objetivo de responder duas RQ direcionadas ao ambiente de desenvolvimento de SoS, cujo intuito é obter e categorizar os problemas que ocorrem no contexto de SoS. As RQ são:

1. **RQ1 - Quais são os desafios enfrentados durante o desenvolvimento de SoS?** -Apontar dificuldades durante o desenvolvimento de SoS, indiferentemente do domínio ou ferramenta utilizada;
2. **RQ2: Foi evidenciada alguma solução para os desafios encontrados?** -Verificar se foi apontada alguma solução/consideração para os problemas encontrados durante a análise dos estudos.

Com a condução do MS, 39 estudos foram selecionados para a fase de extração e síntese, visto que foram lidos completamente com o objetivo de extrair: (i) identificação dos diferentes domínios de uso de SoS; (ii) identificação dos tipos de estudos no contexto de SoS; (iii) identificação dos desafios/problemas que ocorrem no uso de SoS; e (iv) determinação se algum autor propõe ou não uma solução para os problemas descritos.

2.3.2 Resultados

2.3.2.1 Domínios e Tipos de Estudos

A maioria dos estudos (61.53%) não deixava claro qual era o domínio de aplicação do SoS, entretanto, era comum o uso de exemplos de sistemas grandes (sistemas complexos e/ou de grande escala). Isso pode ser observado na Tabela 2, em que nota-se a presença de 24 estudos sem domínio específico. O único domínio com um número razoável de estudos foi o de automação industrial.

Tabela 2 – Classificação dos Domínios de Aplicação

Quantidade	Domínio
24	Sem domínio específico
4	Automação industrial
2	Sistema de observação da terra
2	Departamento de defesa
2	Setor público
2	Marítimo
2	Fisicamente acoplado
1	Automotivo, Estrada de ferro e Aviônica

Fonte: Elaborada pelo autor.

Na Tabela 3 são listados os tipos de estudos encontrados e sua respectiva quantidade. O propósito principal dessa categorização foi visualizar o desenvolvimento da área e verificar se os problemas tinham alguma relação com os domínios e os tipos de estudo. Posto isto, o primeiro

tipo de estudo é relacionado a estudos secundários, mais especificamente uma revisão e um mapeamento sistemático. Ambos são do contexto de SoS, porém são direcionados a diferentes usos, no qual o primeiro retrata integração e o segundo o desenvolvimento de SoS com base em Internet das Coisas.

Dez estudos foram classificados como *frameworks*, os quais abordam processos, testes, técnicas, e/ou metodologias. Já os tipos de estudos classificados como arquitetura propõem padrões, definem conceitos ou abrangem algumas dificuldades relacionadas à linha de pesquisa de SoS.

Os estudos que discutem a área de SoS ou SoSE em si, explicando conceitos, definições ou abrangendo problemas comuns relacionados a tipos específicos ou não, foram denominados como abordagem. Pesquisas que discutem um domínio mais específico ou tem um escopo mais limitado do problema, de forma que caracterizem o ambiente de aplicação, foram classificadas como estudo de caso. Por fim, foi encontrada uma ontologia, que discute e define termos da integração de SoS (SoSI - do inglês *SoS Integration*).

Tabela 3 – Classificação dos Tipos de Estudos

Quantidade	Tipo
2	RS e MS
10	Framework
4	Arquitetura
15	Abordagem
7	Estudo de caso
1	Ontologia

Fonte: Elaborada pelo autor.

2.3.2.2 Conclusões das Questões de Pesquisa

-RQ1-O objetivo desta questão foi extrair os desafios/problemas encontrados no desenvolvimento de SoS. O resultado dessa questão foi alcançado depois de analisar os 39 estudos, cujo propósito era obter uma visão geral dos principais problemas em aberto. A seguir, são apresentadas as categorias de problemas encontrados, que é resumida na Tabela 4:

Tabela 4 – Categorização dos Problemas

Quantidade	Desafio
15	Integração
11	Gerenciamento
5	Documentação
10	Evolução e Manutenção
8	Arquitetura e Modelagem
5	Requisitos e Qualidade
2	Teste
3	Reconfiguração Dinâmica e Monitoramento
5	Comportamento

Fonte: Elaborada pelo autor.

- **Integração:** A integração de SoS não é uma tarefa trivial. De acordo com [Madni e Sievers \(2014\)](#), a integração deve ser tratada durante o desenvolvimento de SoS, de modo que esteja presente ao longo do ciclo de vida. Estudos nessa categoria retratam principalmente que a integração ainda é um desafio por causa da falta de conhecimento específico sobre cada CS; a variedade de implementações (os CS's não possuem padronização de *interfaces*); independência gerencial e operacional pode ser um problema quando a proteção da informação é necessária; dificuldade em coordenar o SoS ou o CS; a distribuição geográfica dos CS's pode causar problemas de comunicação e compartilhamento de informações; e dificuldade em interoperar sistemas heterogêneos de forma eficaz e transparente. A dificuldade na integração de sistemas não é um tópico novo, e é fácil notar a variedade de problemas que ocorrem, uma vez que esses problemas (padronização, interdependência, interoperabilidade e complexidade) também são encontrados em outras áreas de engenharia de software;
- **Gerenciamento:** As dificuldades de gerenciamento começam ao procurar engenheiros qualificados para trabalhar na área de SoS. Isso acontece devido à alta necessidade de conhecimento específico sobre sistemas em geral (ou sobre um sistema particular) e é agravada pela necessidade de controlar vários *stakeholders* que apresentam diferentes objetivos e problemas de comunicação. Além disso, vários riscos podem estar presentes e futuras incertezas do SoS devem ser gerenciadas com cuidado;
- **Documentação:** Um SoS necessita de comunicação e compartilhamento para evoluir adequadamente. A falta de documentação implica que não há diretrizes disponíveis para auxiliar os engenheiros durante o processo de desenvolvimento. Isso afeta o crescimento como um todo, causando problemas de dependência de engenheiros que conhecem o sistema e como fazê-los interoperar com novos sistemas. Além disso, ela pode afetar outras subáreas, uma vez que torna a comunicação mais difícil, por exemplo, ao contratar engenheiros para evoluir ou integrar novos sistemas;
- **Evolução e Manutenção:** O desenvolvimento evolutivo é uma das principais características do SoS, mas fazê-lo em tempo real é extremamente complexo. Portanto, a colaboração em um SoS pode ser comprometida devido à dificuldade em evoluir o sistema como um todo. De acordo com [Henshaw et al. \(2013\)](#), a evolução do SoS pode prejudicar o seu desempenho, uma vez que novos comportamentos emergentes prejudiciais podem surgir e, consequentemente, essa característica única do SoS pode tornar pouco confiável a generalização de experimentos. Nesse contexto, também há um estudo ([BATISTA, 2013](#)) que descreve que os componentes podem ser adicionados e removidos há qualquer momento, visto que a independência e evolução ao longo do tempo é uma característica predominante do SoS e, como tal, a interação e composição dos CS's do SoS são realizados com o propósito de cumprir uma missão compartilhada;

- **Arquitetura e Modelagem:** De acordo com [Kalawsky et al. \(2013\)](#), é muito difícil modelar e analisar um SoS por meios convencionais, especialmente como representar o SoS como um todo, porque os detalhes completos dos CS's podem não estar prontamente disponíveis. Portanto, nessa categoria, as discussões sobre arquitetura e modelagem foram classificadas no contexto de SoS, a partir da baixa mobilidade da comunidade na evolução das arquiteturas de referência existentes até a dificuldade de representar as características de um SoS e como representá-lo em um modelo único;
- **Requisitos e Qualidade:** De acordo com [Batista \(2013\)](#), a qualidade é uma questão chave para o sucesso do SoS, pois inconsistências e ambiguidades podem surgir quando atributos de qualidade são ignorados na arquitetura do SoS. Nessa categoria, foram incluídos estudos que discutem a importância da qualidade e a dificuldade de aplicar a engenharia de requisitos no contexto de SoS;
- **Teste:** Um SoS pode apresentar problemas em seu contexto atual. Além disso, os problemas podem ser encontrados nas *interfaces* dos CS's ou nos próprios CS's, então os desafios de teste são amplificados ([ALI; PETERSEN; MÄNTYLÄ, 2012](#)). Apenas dois estudos retrataram desafios de testes no contexto de SoS;
- **Reconfiguração Dinâmica e Monitoramento:** Três estudos abordaram problemas relacionados à reconfiguração do SoS em tempo de execução, especialmente no contexto de mudanças estruturais e na composição de um SoS; e
- **Comportamento:** Por fim, vários estudos discutiram problemas relacionados às interações, consequências e efeitos do comportamento emergente do SoS.

Contudo, não é possível observar uma relação entre os desafios, tipos e domínio de aplicação dos estudos (Tabela 2 e Tabela 4), ou seja, não há evidências de que um domínio particular apresentaria desafios específicos. No entanto, é possível notar a falta de ferramentas para facilitar o desenvolvimento de SoS e a falta de experiência dos *stakeholders* na área. Isso pode ser (parcialmente) justificado pelo fato de que existe uma grande proporção de projetos fechados e não disponíveis, resultando em uma curva de aprendizado mais alta para engenheiros, além de levar a falta de padronização de arquiteturas e modelos.

-**RQ2**- Essa RQ teve como objetivo investigar os estudos em que os problemas/desafios já possuem uma solução proposta. Nesse sentido, esta questão foi focada em procurar considerações para os problemas apresentados no contexto de SoS. Para facilitar essa investigação, um dos critérios de leitura foi examinar os estudos em ordem decrescente.

De forma geral, essa RQ é muito complexa de ser analisada. Os pesquisadores geralmente apontam considerações ou maneiras de mitigar os problemas em sua área, no entanto, nos estudos sobre SoS, eles tendem a propor soluções específicas para os desafios encontrados durante o desenvolvimento de sua pesquisa ou projetos em andamento. Assim, foi decidido fazer uma

correspondência de estudos que apresentam considerações relevantes para as subáreas dos problemas descritos acima, o que pode ser encontrado no Apêndice A.

2.4 Arquitetura Orientada a Serviços

A Arquitetura Orientada a Serviços (SOA – do inglês *Service-Oriented Architecture*) apresenta um estilo arquitetural com várias vantagens e por isso vêm ganhando um enorme interesse na indústria e na academia. Segundo Erl (2008), SOA é um padrão de *design* de arquitetura distribuída, com *interfaces* de baixo acoplamento.

De maneira simplificada, SOA pode ser vista como uma aplicação da arquitetura cliente-servidor em um novo contexto (MA, 2005). De modo geral, em SOA, um consumidor de serviço (cliente) necessita de uma informação ou confirmação de alguma funcionalidade, logo esse cliente envia uma requisição para um provedor de serviço (servidor) e recebe uma mensagem com a resposta correspondente. Uma visão geral da SOA é apresentada na Figura 6.

Figura 6 – Visão geral da SOA

Fonte: Adaptada de Blomstedt *et al.* (2014).

De acordo com Erl (2005), “serviços são módulos independentes, bem definidos e autocontidos que fornecem funcionalidades de negócios; e são independentes do estado ou contexto de outros serviços, ocultando detalhes de implementação e expondo apenas informações relativas às suas interfaces”. Por isso, pode ser interpretado como uma função específica que permite a solicitação de requisições e que recebe respostas, ressaltando que essas trocas são feitas por uma *interface* predefinida e padronizada. Além disso, são apresentados de duas formas: provedor, expõe sua *interface*, publicando sua descrição para ser posteriormente invocado; e consumidor, pode utilizar as funcionalidades de um provedor de acordo com seu propósito específico (PAPAZOGLU; HEUVEL, 2007).

2.4.1 Características da Arquitetura Orientada a Serviços

SOA oferece várias vantagens durante o processo de desenvolvimento, em que destacam-se (NEWCOMER; LOMOW, 2005; SOUZA, 2006):

- Baixo acoplamento: o baixo acoplamento é o princípio fundamental de SOA. Existem várias formas de baixo acoplamento em SOA e, isso resulta em uma grande vantagem na flexibilidade e agilidade no desenvolvimento;
- Interoperabilidade: serviços podem ser implementados com o uso de tecnologias novas e padronizadas, de forma que seja possível atingir um maior número de plataformas;
- Composição: serviços podem ser combinados com o propósito de fornecer novos serviços com diferentes funcionalidades;
- Reusabilidade: o nível de produtividade aumenta com o uso da arquitetura. Os módulos são independentes e não padronizados para alguma linguagem de programação, resultando assim, em serviços flexíveis e disponíveis para serem reutilizados;
- Alta granularidade: o encapsulamento de funcionalidades no nível de serviço, permite um alto grau de granularidade nos componentes básicos da arquitetura; e
- Ubiquidade: os serviços devem estar disponíveis e acessíveis em qualquer lugar e em qualquer momento.

A abordagem de serviços transcende as soluções de tecnologia e automação. É uma teoria estabelecida e genérica que pode ser usada para tratar uma variedade de problemas ([ERL, 2005](#)).

2.4.2 Web Services

2.4.2.1 Conceitos Básicos

A tecnologia de *Web Services* teve início após a aceitação da submissão do *Simple Object Access Protocol* (SOAP), pelo *World Wide Web Consortium* (W3C) no ano 2000. Esse formato de mensagem baseado em XML, estabeleceu a transmissão entre aplicações e/ou serviços via protocolo de comunicação HTTP, onde foi uma alternativa atrativa para os protocolos proprietários, sendo que seu grande diferencial é sua característica neutra para fornecedores. No ano seguinte, surgiu outra especificação, *Web Services Description Language* (WSDL), também implementada em XML, mas com o intuito de fornecer uma linguagem para descrever a interface dos *Web Services*. Posteriormente, foi complementada pela *Universal Description, Discovery, and Integration* (UDDI), provendo assim, um mecanismo padrão para descrição de descoberta dinâmica de serviços. Posto isso, existem cinco padrões fundamentais para *Web Service* ([JOSUTTIS, 2007](#)):

1. eXtensible Markup Language (XML); e
2. Hypertext Transfer Protocol (HTTP).

Os outros três são específicos para *Web Service* e são os primeiros padrões especificados para eles (Figura 7):

3. Simple Object Access Protocol (SOAP);
4. Universal Description, Discovery and Integration (UDDI); e
5. Web Services Description Language (WSDL).

No contexto geral da computação, os *Web Services* podem ser considerados uma tecnologia relativamente nova, mas que recebeu ampla aceitação como uma importante implementação de SOA. Isso ocorre porque os *Web Services* oferecem uma abordagem de computação distribuída para integrar aplicativos heterogêneos pela Internet. As especificações de *Web Services* são completamente independentes de linguagem de programação, sistema operacional e do hardware para promover o baixo acoplamento entre o consumidor e o provedor de serviços (ENDREI *et al.*, 2004). De acordo com Austin *et al.* (2004), a definição formal é: “Um *Web Service* é um aplicativo de software identificado por um Identificador Uniforme de Recurso (URI - do inglês *Uniform Resource Identifier*), cujas interfaces e ligações são capazes de serem definidas, descritas e descobertas como artefatos XML. Um *Web Service* apoia interações diretas com outros agentes de software, usando mensagens baseadas em XML trocadas por meio de protocolos baseados na Internet”.

Figura 7 – Primeira geração de padrões de especificação

Fonte: Erl (2004).

2.4.2.2 Papéis do Web Service

Os serviços podem assumir diferentes papéis em diferentes cenários de interação. Isso é definido pelo contexto em que o serviço é visualizado, assim como o estado da tarefa que está em execução. Por isso, o mesmo *Web Service* pode mudar de papel ou ser designado simultaneamente para múltiplos papéis. A seguir, apresenta-se detalhadamente cada papel de um *Web Service* (KUMAR, 2004; ERL, 2005):

- *Service Provider*: neste papel, o *Web Service* expõe sua *interface* pública e pode ser requisitado por solicitantes de serviço. O provedor de serviço disponibiliza essa *interface* publicando uma descrição do serviço. Nesse caso, o provedor de serviços pode ser comparado com um servidor em um modelo cliente-servidor padrão. Esse serviço pode informar o resultado de um serviço solicitado, ou pode agir como um solicitante de serviço também;
- *Service Requestor*: este serviço tem a função de remetente de uma mensagem *Web Service* ou de um programa que solicita um *Web Service* específico. Nesse caso, o solicitante de serviço pode ser comparado com um cliente em um modelo cliente-servidor padrão. Ademais, ele também pode responder a sua solicitação inicial realizada para outro *Web Service*, sendo assim, ele pode agir como um provedor de serviços também;
- *Intermediary*: neste papel, o *Web Service* recebe a mensagem do solicitante de serviços e, posteriormente encaminha para o provedor de serviços. Nessa interação é que ocorre o papel do intermediário, que pode ser um provedor (recebendo a mensagem) e um solicitante (passando a mensagem);
- *Initial Sender*: este serviço também pode ser classificado como solicitante, porém esse papel auxilia a identificar o primeiro *Web Service* que solicitou o serviço; e
- *Ultimate Receiver*: este serviço tem um papel semelhante ao anterior, entretanto, esse provedor é o último *Web Service* a receber a mensagem.

2.4.2.3 Estrutura do Web Service

Um *Web Service* é descrito por meio de uma coleção de documentos de definição. A Figura 8 esclarece a relação desses documentos.

Na figura, o *Abstract* é a descrição da *interface* de um *Web Service*, independente dos detalhes de implementação. Essa definição de *interface* abstrata é composta principalmente das construções da mensagem e da *interface* dentro de uma especificação WSDL. A especificação concreta descreve a localização e informação da implementação sobre um *Web Service*, por meio de um documento WSDL em que se encontram os elementos de conexão (*binding*), serviço (*service*) e porta (*endpoint* ou *port*). Os documentos WSDL geralmente contém a definição do serviço (*service definition*), que inclui a *interface* (*abstract*) e a definição da implementação

Figura 8 – Conteúdo da descrição do serviço

Fonte: Erl (2004).

(*concrete*). De maneira geral, uma descrição de serviço (*service description*) consiste em um único documento WSDL, que fornece uma definição do serviço. No entanto, também pode incluir vários documentos de definição adicionais, que auxiliam a fornecer informações complementares (ERL, 2004).

2.5 Especificação de Ativos Reusáveis

Recentemente, muitas técnicas foram desenvolvidas para dar suporte à reutilização de software. Essas técnicas exploram o fato de que sistemas em um mesmo domínio de aplicação são semelhantes e têm potencial para reutilização. De fato, essa reutilização é possível em diferentes níveis, desde funções simples até aplicativos completos; e esses padrões para componentes reutilizáveis facilitam de forma geral o reúso (SOMMERVILLE, 2011).

Um padrão bem conhecido e utilizado na literatura é a RAS (*Reusable Asset Specification*), proposta pelo *Object Management Group* (OMG, 2005). Na perspectiva do OMG, os ativos reusáveis são utilizados como uma forma de solução para um determinado contexto. Na Figura 9, observa-se que o ativo tem um ponto de variabilidade relacionado ao contexto, com regras específicas que devem descrever o seu uso. Por isso, o ativo fornece uma solução abstrata do problema e os artefatos são quaisquer tipos de arquivo (documentos de requisito, modelos, código fonte, casos de teste, entre outros) compartilhados com o objetivo de reutilização.

A RAS é descrita em duas categorias principais: (i) *Core RAS*, que representa os elementos fundamentais da especificação dos ativos; e (ii) *Profiles*, que descreve as extensões dos elementos fundamentais. Na Figura 10 é apresentado o relacionamento entre as categorias. O

Figura 9 – Definição Geral de Ativos Reusáveis

Fonte: Adaptada de [OMG \(2005\)](#).

Default Profile é uma realização da *Core RAS*; e *Default Component Profile* e *Default Web Service Profile* são derivações do perfil padrão, que estendem suas semânticas básicas.

Figura 10 – Relacionamento entre RAS e Profiles

Fonte: Adaptada de [OMG \(2005\)](#).

No âmbito de descrições, a *Core RAS* define elementos e quatro seções principais para o ativo, que são: Classificação, Solução, Uso e Ativos Relacionados (Figura 11). Na Figura 12, é ilustrado como são os relacionamentos de domínio do *Core RAS*.

Por fim, na Figura 13, é apresentado o modelo com as classes principais utilizadas na especificação dos ativos reusáveis. É importante ressaltar que o padrão RAS foi utilizado como uma inspiração para este projeto. Para isso, algumas classes e relacionamentos foram utilizadas e adaptadas para o modelo da abordagem, a fim de fomentar o reúso e busca dos elementos de um SoS. A necessidade de adaptar a RAS é decorrente das especificidades inerentes ao reúso em SoS, conforme explicado posteriormente no Capítulo 3.

Figura 11 – Seções da Core RAS

Fonte: Adaptada de [OMG \(2005\)](#).

Figura 12 – Relacionamento das Seções do Core RAS

Fonte: Adaptada de [OMG \(2005\)](#).

2.6 Arquitetura Baseada em Modelos para Integração de Sistema de Sistemas

Conforme mencionado na Seção 1.3, este projeto de mestrado visa contribuir com um projeto de pesquisa coordenado pela mesma orientadora, que propõe uma arquitetura denominada MBA4SoSI (do inglês - *Model-Based Architecture for System of Systems Integration*), a qual visa facilitar a integração de sistemas rumo a um SoS. Essa arquitetura utiliza SOA para favorecer a interoperabilidade dos CS's e desenvolvimento baseado em modelos (MDD – do inglês *Model-Driven Development*) para facilitar a união de diferentes plataformas de software e futura evolução do SoS e seus CS's.

O principal objetivo da MBA4SoSI é averiguar a união dos conceitos de SoS, SOA e MDD para desenvolver uma arquitetura integrada de processos e ferramentas que facilitem a integração de sistemas existentes, em direção a um SoS. Como resultados esperados desse projeto, estão: a MBA4SoSI, a instanciação da MBA4SoSI para integração de ambientes de

Figura 13 – Modelo da RAS

Fonte: Adaptada de [OMG \(2005\)](#).

reúso de software e de aplicações da Internet das Coisas, um conjunto de diretrizes genéricas para integração de sistemas em SoS e ferramental de apoio à MBA4SoSI ([BRAGA, 2016](#)).

Alguns trabalhos foram realizados para justificar a necessidade desse *framework* de integração. [Vargas, Gottardi e Braga \(2016\)](#) averiguaram, por meio de uma revisão sistemática (RS), abordagens, técnicas e ferramentas que podem auxiliar na integração de sistemas rumo a um SoS. Um resultado importante extraído da RS foi que grande parte dos estudos (54.16%) utilizam SOA como uma solução arquitetural no contexto de SoS. Os autores também investigaram por meio de um mapeamento sistemático (MS), como abordagens, técnicas e ferramentas têm sido integradas em ambientes de reúso de software. Nesse MS, foi evidenciado que grande parte dos estudos encontram-se em estado inicial em direção à integração de ambientes de reúso, e consequentemente, não há trabalhos que utilizam SoS como uma possível solução.

Na Figura 14, é exposto um esboço da MBA4SoSI, com seus módulos específicos e

necessários para obter sucesso nessa solução arquitetural. O presente trabalho de mestrado representou uma contribuição para os módulos delimitados pela linha em vermelho, que referem-se ao controle de serviços, CS e SoS, além de um módulo não representado explicitamente para controle de usuários.

Figura 14 – Visão Geral da MBA4SoSI

Fonte: Adaptada de [Braga \(2016\)](#).

Por fim, é importante enfatizar que o projeto MBA4SoSI conta com outros alunos de mestrado e doutorado que atuarão em outras partes da arquitetura. Em particular, o módulo *SoS Managmt* tem um propósito muito maior do que esta pesquisa de mestrado e, por isso, esse módulo foi implementado de maneira simplificada de acordo com o propósito e o escopo do mestrado, e será evoluído em outros trabalhos.

2.7 Trabalhos Relacionados

Nesta seção são discutidos alguns trabalhos que utilizam SOA como solução para interoperabilidade de CS de um SoS, assim como repositórios e abordagens que utilizam a RAS. O intuito é revisar o estado da arte e relacionar o trabalho desenvolvido com pesquisas que de alguma forma apresentam aspectos importantes para o projeto aqui apresentado.

[Lewis et al. \(2011\)](#) apresentam uma maneira para facilitar a interoperabilidade entre os CS's de um SoS, considerando a interação de diferentes *stakeholders* em uma abordagem SOA. Para os autores, SoS é uma classe que apresenta características importantes e que pode ser útil na integração de sistemas monolíticos. Os principais benefícios da SOA são utilizados com o

objetivo de propor uma abordagem para a SoSE, na qual as etapas e os atores fundamentais são descritos para facilitar o entendimento e uso de engenheiros/integradores.

Na abordagem (Figura 15), existem quatro atores principais: (i) *SoS end user*, que representa o solicitante de novas capacidades do SoS; (ii) *SoS developer/integrator*, que representa o time responsável pelo desenvolvimento e/ou integração das novas capacidades que o *SoS end user* solicitou; (iii) *System developer*, que representa o time de desenvolvimento do CS; e (iv) *System end user*, que representa os usuários finais do CS. Vale ressaltar que na abordagem proposta neste trabalho de mestrado são adotados (e adaptados) esses mesmos quatro atores. O ciclo proposto por [Lewis et al. \(2011\)](#) é detalhado da seguinte forma:

Figura 15 – Uma abordagem orientada a Serviços para SoS

Fonte: [Lewis et al. \(2011\)](#).

1. O *SoS end user* tem um novo requisito para uma nova capacidade;
2. O *SoS developer/integrator* analisa os requisitos da nova capacidade;
3. O *SoS developer/integrator* busca no repositório de capacidades uma correspondência;
4. O *SoS developer/integrator* analisa o retorno do repositório;
5. As capacidades podem ou não existir no repositório. Caso não exista, pode ser solicitado uma alternativa ou um novo desenvolvimento;

6. Se a compatibilidade não for exata, o *SoS developer/integrator* pode adaptar (por código) uma capacidade já existente;
7. O *SoS developer/integrator* integra a nova capacidade no SoS;
8. O *SoS developer/integrator* testa e valida a nova capacidade no SoS;
9. O *SoS developer/integrator* implanta a nova capacidade;
10. A nova capacidade é entregue ao *SoS end user*;
11. O *SoS developer/integrator* busca alternativas (repete a busca com critérios diferentes ou apresenta ao *SoS end user* outras capacidades;
12. Inicia o passo 2 novamente.

Se a capacidade não existe no repositório, mas o *SoS developer/integrator* pode esperar o desenvolvimento:

13. O *SoS developer/integrator* envia os requisitos para o *system developer*;
14. O *system developer* desenvolve a nova capacidade;
15. O *system developer* pode decidir disponibilizar a nova capacidade para o *System end user*;
16. O *system developer* padroniza e publica a nova capacidade no repositório. É importante ressaltar que o *SoS developer/integrator* pode solicitar o teste antes de publicar;
17. A capacidade fica disponível para o *SoS developer/integrator* depois que ela é publicada. Inicia o passo 3 novamente;

O tempo necessário para realizar essa iteração depende de quanto tempo o *SoS end user* pode esperar no desenvolvimento da capacidade, ou o quanto ele está disposto a aceitar desajustes em capacidades semelhantes.

Como parte do projeto da MBA4SoSI, [Vargas \(2017\)](#) propôs o SoS-Reúso, um SoS do tipo direcionado que possui como CS's instâncias de um ambiente de reúso de software. Considera-se que cada CS é um sistema independente operacionalmente e gerencialmente, servindo como repositório para ativos reusáveis em determinado domínio e de determinado tipo. Diante disso, essa abordagem tem o intuito de facilitar a implementação de comportamentos emergentes relacionados ao reúso de software, por meio da integração de ambientes já existentes. Na Figura 16 é apresentada a visão geral da abordagem. Ademais, o SoS-Reúso também utiliza a RAS como inspiração e, por isso, faz uso de classes e atributos específicos do padrão de reúso de ativos.

Figura 16 – Visão Geral do SoS-Reúso

Fonte: Vargas (2017).

Varga *et al.* (2017) apresentam um dos principais projetos que permitem a automação colaborativa em dispositivos em rede. De acordo com os pesquisadores, o *framework* Arrowhead compreende um conjunto de diretrizes para o desenvolvimento de dispositivos, que leva em consideração os requisitos dos sistemas baseados em SOA. Nesse sentido, o *framework* fornece um conjunto de regras para a configuração da arquitetura dos diferentes elementos definidos na plataforma, de modo que auxilie e assegure a interoperabilidade entre diferentes sistemas e/ou dispositivos. Na Figura 17 é apresentado uma visão geral desse *framework*.

Figura 17 – Visão Geral do Framework Arrowhead

Fonte: Varga *et al.* (2017).

Em termos de uso, o *Arrowhead* apresenta três serviços básicos essenciais para fornecer uma arquitetura SOA funcional: (i) descoberta: serviço responsável por registrar e descobrir todos os serviços que estão registrados e disponibilizados no *framework*; (ii) autorização: serviço responsável pelo controle de acesso dos serviços disponíveis aos usuários do *framework*; e (iii) orquestração: serviço responsável pelo controle de como os sistemas são desenvolvidos e como devem ser conectados.

Em outro projeto relacionado ao reúso de software, apresenta-se a AIRES (*ArchItecture for REuse using Services*), que é um modelo arquitetural que abrange os principais elementos necessários para suportar a reutilização de software em grande escala. Esse modelo (Figura 18) visa tornar a reutilização intrínseca ao ciclo de vida do processo de desenvolvimento, fornecendo mecanismos para facilitar uma variedade de processos e representações de artefatos. A AIRES também é baseado em SOA e RAS, permitindo consumir os serviços fornecidos para gerenciar os ativos, que podem ser armazenados na nuvem (BRAGA *et al.*, 2016).

Figura 18 – Visão Geral do Modelo Arquitetural da AIRES

Fonte: Braga *et al.* (2016).

Do ponto de vista de repositórios, existem algumas pesquisas que desenvolveram repositórios com base na RAS. Por exemplo, LAVOI (MOURA, 2013) e OpeCom (REN; YAN; ZHANG, 2009) fizeram adaptações no perfil RAS para vários tipos de ativos, de modo que facilite a classificação e o uso dos ativos reusáveis. No entanto, não foi encontrado nenhum trabalho que use a SOA para desenvolver um repositório no contexto de SoS.

Abdalla (2017) apresenta vários conceitos do campo de pesquisa de SoS em seu trabalho

de mestrado, visando principalmente estabelecer um consenso sobre os termos, de forma que facilite a comunicação e o compartilhamento das características de um SoS. Nesse trabalho, é estabelecida uma ontologia que descreve termos e conceitos relacionados ao campo de SoS. Para isso, um vocabulário comum foi definido, de modo que uma compreensão comum da informação possa ser compartilhada entre profissionais e pesquisadores, trazendo a reutilização do conhecimento e tornando explícitas as definições. A correlação dessa pesquisa com esta proposta de mestrado é que a ontologia pode servir como um guia para engenheiros que desejam desenvolver SoS, por isso, algumas partes dessa ontologia foram estudadas e adotadas neste trabalho de mestrado. Os detalhes da adaptação são descritos no Capítulo 3.

Por fim, os trabalhos de [Lewis *et al.* \(2011\)](#), [Vargas \(2017\)](#) foram importantes para a condução desta pesquisa, em razão da exemplificação dos princípios de SOA para o contexto de SoS. Além disso, os atores utilizados no repositório proposto foram modelados e adaptados com base nesses trabalhos.

2.8 Considerações Finais

Neste capítulo foram apresentados os principais conceitos necessários para embasar esta proposta de mestrado. Os conceitos de SoS são fundamentais para a compreensão e foco deste trabalho, uma vez que a abordagem proposta foi desenvolvida com o objetivo de auxiliar engenheiros e pesquisadores que trabalham com essa classe de sistema. Ademais, também foi resumido um MS realizado com o objetivo de obter uma visão dos desafios que os pesquisadores enfrentam, além de auxiliar a compreensão e familiaridade do pesquisador com a área. SOA é empregada por diversos pesquisadores com uma solução de interoperabilidade de CS's que desejam se unir para formar um SoS. Por isso, é importante apresentar e caracterizar esse estilo arquitetural.

Na perspectiva de reúso, foi apresentado um padrão adotado (RAS) que serviu como inspiração para aperfeiçoar a busca e reúso dos elementos de um SoS. O projeto MBA4SoSI também foi apresentado, visto que este projeto de mestrado é parte dele. Por fim, foram apresentados os trabalhos que de alguma forma apresentam fatores interessantes que podem ser correlacionados a esta pesquisa. Entretanto, é importante mencionar que não foram encontradas pesquisas com interesse em criar um repositório para SoS, de forma que incentive o reúso e documentação dos CS's.

UMA ABORDAGEM ORIENTADA A SERVIÇOS PARA FACILITAR O REÚSO NO CONTEXTO DE SOS

3.1 Considerações Iniciais

Neste capítulo, é apresentada a abordagem baseada em serviços proposta nesta dissertação, que visa facilitar o reúso no contexto de SoS. Essa abordagem foi desenvolvida com base nos fundamentos estabelecidos na literatura e em outros trabalhos de pesquisa já validados, utilizando definições, entidades e atores importantes para uma proposta com essa finalidade, cujo principal foco é nos CS's e seus serviços. Portanto, no decorrer deste capítulo, serão respondidas (implicitamente) às seguintes perguntas:

- O que é a abordagem?
- Quem são os atores do repositório?
- O que são os *assets* do repositório e por que a RAS foi utilizada?
- Qual é a relação entre os atores?
- Como os *assets* são disponibilizados?

Assim, na Seção 3.2 são apresentados os conceitos iniciais e uma visão geral da abordagem, com seu modelo conceitual, repositório de *assets*, principais casos de uso e as referências utilizadas no desenvolvimento. Com o intuito de esclarecer o que é o repositório e o seu funcionamento, foram utilizadas personas para apresentar os atores, além de um cenário típico de uso. Em seguida, na Seção 3.3, é apresentada uma validação da abordagem por meio de um *survey*. Essa validação apresentou visões importantes sobre a abordagem e o escopo do trabalho, por

isso, as considerações dos sujeitos foram analisadas e utilizadas para o refinamento da proposta. As sugestões de mudança que não foram acatadas são explicadas pelo proponente em tabelas com suas respectivas respostas.

3.2 Abordagem Proposta

3.2.1 Conceitos Iniciais

A abordagem aqui proposta visa facilitar o reúso no contexto de SoS. Neste trabalho, entende-se por abordagem um conjunto com os seguintes elementos: (i) modelo conceitual; (ii) repositório de *assets*; (iii) definição dos atores envolvidos e seus papéis; e (iv) diretrizes para o uso do repositório. Uma visão da abordagem é apresentada na Figura 19.

Figura 19 – Abordagem Proposta

Fonte: Elaborada pelo autor.

Os elementos da abordagem foram definidos e desenvolvidos para auxiliar os engenheiros de software que, para que os SoS's possam alcançar um comportamento emergente, precisam de recursos que não têm em seus sistemas, mas que existem em outros sistemas independentes e, muitas vezes, no mesmo domínio. De forma geral, a abordagem destina-se a fomentar o compartilhamento e organização das capacidades no contexto de SoS, além de propor diretrizes e documentação para o uso. Conforme descrito na Seção 1.2, isso faz parte de um projeto maior de interesse do grupo de pesquisa da qual este trabalho de mestrado faz parte, mais especificamente, o projeto MBA4SoSI, descrito na Seção 2.6.

O repositório proposto é direcionado ao contexto de SoS, por isso, as características intrínsecas e termos inerentes a SoS foram considerados para promover o reuso nesse domínio específico. Assim, a criação da abordagem foi feita tendo em mente principalmente, mas não exclusivamente, engenheiros de SoS que precisam desenvolver com rapidez um novo comportamento emergente. Para isso, eles gostariam de beneficiar-se do reuso de ativos disponibilizados por potenciais sistemas constituintes acessíveis por meio do repositório.

Entretanto, deve-se notar que os atores nem sempre estarão interessados em desenvolver um SoS. Nesse sentido, é possível que alguns usuários tenham apenas o interesse de procurar serviços para suas aplicações *standlone*, assim como compartilhar seus próprios serviços, de maneira semelhante (ou comparando) a um sistema orientado a serviços e não um SoS que é maior que a soma de suas partes. Isso faz com que a abordagem sirva também para reuso de software em geral, e não somente no contexto de SoS, para o qual ela foi especialmente concebida.

Por fim, é importante ressaltar que a abordagem é orientada a serviços, ou seja, o acesso ao repositório é feito exclusivamente por invocação de serviços pelas aplicações cliente que o desejarem. Isso facilita seu uso por quaisquer aplicações que se interessarem em promover o reuso de SoS, CS's e respectivos serviços. Mais detalhes sobre como isso pode ser feito são fornecidos no Capítulo 4.

3.2.2 Visão Geral e Modelo Conceitual da Abordagem

O passo inicial realizado na condução deste trabalho de mestrado foi procurar a melhor maneira de compartilhar os recursos de um sistema monolítico que seja candidato potencial a colaborar em um SoS. Neste trabalho, um sistema monolítico é descrito como CS, pois esta proposta é direcionada a um ambiente em mudança, no qual existem engenheiros/desenvolvedores que buscam serviços (também descritos como capacidades) para seus sistemas *standlone*.

Nesse ambiente, vários *stakeholders* procuram interoperar vários CS's com o objetivo de atingir um sinergismo maior. Um exemplo simples que ilustra como um SoS é uma forma de solução para esse ambiente de mudança, é o de cidades inteligentes (CAVALCANTE *et al.*, 2016), em que se faz necessária a comunicação entre sistemas verticais existentes, para facilitar a vida dos cidadãos e agilizar o controle/funcionamento da cidade. Nesse cenário, podem surgir comportamentos emergentes para tratar situações específicas, e a implementação desses comportamentos requer a interoperabilidade entre os sistemas verticais, por exemplo, o tratamento de situações caóticas diante de eventos como enchentes ou epidemias.

Conforme relatado nos capítulos anteriores, a revisão bibliográfica indicou que a interoperabilidade entre diferentes CS's sofre várias barreiras. Assim, na abordagem proposta decidiu-se criar um repositório de ativos reusáveis e direcioná-lo para a classe de SoS, de forma que pudesse auxiliar na solução dos problemas encontrados.

Dessa forma, foi necessário fazer uma revisão da literatura para descobrir a melhor maneira de atacar esse problema, mais especificamente descobrir como e quais elementos devem ser compartilhados, de maneira que facilite o reúso. Vale a pena ressaltar que o MS foi um passo importante nessa condução, porque foi possível entender como os pesquisadores estão trabalhando e quais problemas enfrentam, como por exemplo falta de documentação e ferramentas de suporte, e dificuldade de encontrar capacidades.

Uma outra atividade importante nessa fase inicial foi obter um completo entendimento sobre os elementos de um SoS, suas várias categorias e classificações, para identificar quais características seriam essenciais em um repositório voltado ao contexto de SoS. Isso levou a um primeiro esboço do modelo conceitual da abordagem.

Ademais, diversos trabalhos também foram estudados e contribuiriam para a composição do repositório, a saber: (i) elementos de uma ontologia de SoS para o desenvolvimento do modelo ([ABDALLA, 2017](#)); (ii) uma adaptação de atores que encontram-se na literatura para o repositório ([LEWIS *et al.*, 2011](#); [VARGAS, 2017](#)); e (iii) o padrão de ativos reusáveis (RAS) ([OMG, 2005](#)). No caso da RAS, é importante enfatizar que ela foi adaptada e não utilizada em sua totalidade, isto é, foram utilizados atributos e elementos com o objetivo de fomentar o reúso e aprimorar a busca no contexto de SoS. Nesse sentido, este trabalho adotou da RAS os conceitos de *assets*, sendo que são considerados *assets* tanto CS's quanto serviços e SoS's. Com isso, a granularidade do reúso pode ser desde um simples serviço, até um CS com seus serviços, ou um SoS em si.

Na Figura 20, apresenta-se o modelo conceitual que compreende todos os conceitos envolvidos na abordagem proposta, bem como o relacionamento entre eles. Na Tabela 5, apresentam-se: o glossário com a descrição e a referência usada na criação de cada elemento do modelo conceitual. As possíveis referências podem ser, em especial, advindas dos: trabalhos resultantes do MS, a ontologia de SoS ([ABDALLA, 2017](#)), a RAS ([OMG, 2005](#)), ou sugestões do próprio autor desta dissertação (proponente).

Tabela 5 – Glossário do Modelo Conceitual (em ordem alfabética)

Entidade	Descrição	Referência
Acknowledged	Este tipo de SoS tem objetivos reconhecidos, um gerente e recursos no nível SoS. Além disso, O gerente não possui autoridade completa sobre os CS's.	(LANE, 2013)
Asset	Entidade que representa um asset reusável que fornece uma solução para um determinado problema dentro de um contexto.	(OMG, 2005)
CatalogOfTrustableSystems	Entidade que representa um Catálogo de Sistemas Confiáveis, que referem-se a CS's que podem ser confiáveis pelo SoS e serão incluídos na composição.	Proponente, adaptado de (VARGAS, 2017; BRAGA; VARGAS; GOT-TARDI, 2016)
Centralized	Tipo de SoS gerenciado por uma autoridade central.	(ABDALLA, 2017)
Classification	Entidade que contém descritores que classificam as principais características de um asset.	(OMG, 2005)
Collaborative	Nesse tipo de SoS, as equipes trabalham juntas para cumprir os propósitos comumente acordados. Não existe uma equipe SoSE para orientar ou gerenciar as atividades relacionadas aos SoS's dos CS's.	(LANE, 2013)
ConstituentSystem	Um sistema monolítico que compõe um SoS.	(DOD, 2008)
Context	Entidade que ajuda a definir o domínio de um asset e classificá-lo de acordo com um determinado contexto.	(OMG, 2005)
CSOwner	Proprietário ou desenvolvedor do constituinte. Esta é a pessoa a contatar no caso de qualquer necessidade em relação ao CS.	(LEWIS <i>et al.</i> , 2011; VARGAS, 2017)
Decentralized	Tipo de SoS não gerenciado por uma autoridade central.	(ABDALLA, 2017)
Description	Uma entidade simples que contém uma descrição completa do asset.	(OMG, 2005)
Directed	Essa classe de SoS é gerenciada e construída de forma centralizada para atender a objetivos específicos. Os CS's mantêm sua capacidade de operar de forma independente, mas a evolução é controlada pelo gerenciador do SoS.	(LANE, 2013)
Goal	Um objetivo específico atribuído a um sistema.	Proponente
Model	Entidade que representa modelos usados para melhor especificar o asset. Pode haver vários modelos, como modelo conceitual e caso de uso.	(OMG, 2005)
Service	Serviços que oferecem funcionalidade de negócios reutilizáveis por meio de interfaces padronizadas e bem definidas para promover a interoperabilidade.	(LEWIS <i>et al.</i> , 2011)
ServiceType	Entidade que categoriza o tipo de serviço. Dessa forma, os serviços com o mesmo propósito são agrupados.	Proponente
SoSDeveloper	Engenheiro, integrador ou arquiteto com interesse na gestão/desenvolvimento de SoS.	(LEWIS <i>et al.</i> , 2011; VARGAS, 2017)
Stakeholder	Pessoa física ou organização com direito, participação, reivindicação ou interesse em um sistema ou em sua posse de características que atendam às suas necessidades e expectativas.	ISO/IEC/IEEE 24765
SystemOfSystems	Um conjunto ou arranjo de sistemas resultante da integração de CS's independentes e úteis em um sistema maior que oferece recursos únicos.	(DOD, 2008)
SystemOfSystemsType	Categorização do SoS de acordo com a extensão da independência gerencial apresentada pelos seus CS's.	(ABDALLA, 2017)
User	O usuário representa uma pessoa externa para a abordagem. Essa pessoa pode ser o proprietário de um CS (CSOwner), um desenvolvedor de SoS (SoSDeveloper) ou alguém que tenha interesse em encontrar um serviço, CS ou SoS (um usuário simples - User).	(LEWIS <i>et al.</i> , 2011; VARGAS, 2017)
Virtual	Essa classe de SoS não possui uma autoridade de gerenciamento central e um propósito claro. É ad hoc e os CS's podem não ser reconhecidos.	(LANE, 2013)

Fonte: Elaborada pelo autor.

3.2.3 Diretrizes de uso

Para esclarecer o funcionamento e o significado dos principais elementos e/ou atributos da abordagem, são fornecidas nesta seção diretrizes de uso, em que apresenta-se cada ator individualmente por meio de personas. A primeira persona é representada por Suelen (*CSOwner*), uma desenvolvedora e proprietária de um CS que provê serviços de geolocalização para encontrar CEP de ruas, localizar um dispositivo, buscar/listar latitude e longitude, procurar cidades e estados, entre outros.

Para entrar no repositório, Suelen precisa fornecer suas informações na entidade *User*, preenchendo alguns atributos, como seu nome, ocupação, *e-mail*, *login*, senha e se deseja receber solicitações de outros usuários para implementar serviços relacionados a seu CS (atributo *requestFunctionality*), mas que ainda não têm no repositório. Após isso, ela pode cadastrar seu CS informando o endereço de acesso (*csHost*) e a porta, assim como alguns atributos necessários e obrigatórios que auxiliam a classificar e identificá-lo, ou seja, os elementos da RAS: *Asset*, *Description*, *Classification* e *Context*. Posteriormente, pode ser cadastrado qualquer tipo de modelo (por exemplo, textual ou da UML - *Unified Modeling Language* (RUMBAUGH; JACOBSON; BOOCH, 2004)) que auxilie outros usuários a entender seu CS, por meio da entidade *Model*, ou seja, que constitua a documentação do serviço adicionado.

Cada serviço do CS da Suelen deve ser cadastrado com base na entidade *Service*, informando o nome do serviço, URI de acesso, a interface do serviço (*reference*), versão atual, *status* e o direito de acesso (público ou privado). No repositório, os serviços que compartilham o mesmo propósito são agrupados e oferecidos de forma categorizada e organizada por meio da entidade *ServiceType*. Assim, os serviços de CS's diferentes que possuem o mesmo objetivo são encontrados na mesma categoria, mesmo que eles tenham interfaces diferentes. Diante disso, Suelen precisa encontrar uma categoria para seu serviço ou cadastrar uma nova. O *ServiceType* também usa (herança) elementos da RAS que auxiliam e enriquecem a busca de *assets*, por isso, os atributos necessários devem ser preenchidos para realizar o cadastro. No caso dela, uma categoria relacionada à geolocalização seria ideal para cadastrar seus serviços.

Suelen pode adicionar mais CS's e serviços a qualquer momento, além de poder editar as informações ou removê-los. Outro ponto importante é o atributo *currentStatus*, que pode conter os valores 0-Desativado, 1-Ativado, 2-Revisão e 3-NãoPublicado. Esse atributo é essencial, porque ela possui pouco tempo na área de geolocalização e tem um servidor pequeno que recebe várias requisições no fim de ano. Dessa forma, ela pode gerenciar seus *assets* mudando apenas o *status*, sem precisar removê-los, em épocas em que ocorrem muitas solicitações.

Um outro ator presente no repositório, é o *SoSDeveloper* representado por Manuel, um engenheiro que trabalha na prefeitura e gerência um SoS de causas emergenciais. O SoS dele possui alguns comportamentos emergentes, em especial um denominado EB1, que calcula rotas e manda mensagens para determinados departamentos em causas emergenciais. Por exemplo,

quando ocorre um incêndio, esse comportamento usa serviços de vários CS's para localizar o endereço, enviar mensagens, avisar o corpo de bombeiros mais próximo, calcular a rota mais curta e se necessário, avisar o departamento de polícia.

Recentemente, esse comportamento emergente foi comprometido em razão da falta de serviços de geolocalização, já que um CS de seu SoS cessou de prover o serviço. Manuel soube do repositório e decidiu criar uma conta para utilizá-lo, com o objetivo de encontrar um CS que forneça serviços de geolocalização para conseguir implementar esse comportamento novamente.

Após uma busca minuciosa, ele encontrou o CS de Suelen, que oferece exatamente os serviços necessários para seu SoS. A interoperabilidade dos CS's foi facilitada para ele, em razão dos modelos que esclarecem o funcionamento dos *assets* disponibilizados por ela.

Outras agências também usam os serviços dos CS's e comportamentos emergentes do SoS do Manuel, porém, a organização e o acesso são sempre um problema, por exemplo, quando novos engenheiros eram contratados e precisavam entender para depois acessar. Um exemplo disso é a empresa de vigilância do João, que utiliza o comportamento emergente EB1 para fazer chamados em casos de emergência, visto que a sua empresa é contratada para proteger vários condomínios na cidade. Além disso, João possui várias filiais espalhadas pelo Brasil e é comum ter mais de um ponto de gerenciamento em cada cidade que, por sua vez, são responsáveis por toda comunicação entre os engenheiros, clientes e os vigias. Nesses pontos podem ocorrer problemas, pois sempre contratam novos engenheiros para administrar a segurança do local, em que uma das principais atividades é interoperar o sistema do ponto com o comportamento emergente EB1, fornecido pelo SoS de Manuel. Nesse caso, um engenheiro mais experiente sempre fica com a atividade de ensinar como funciona o sistema do ponto com o comportamento emergente EB1, sem mencionar que Manuel também é consultado com frequência, não só por essa empresa, mas também por outras que costumam enfrentar o mesmo problema.

Com esse problema em mente, Manuel percebeu como o repositório auxiliou na sua busca e resolveu disponibilizar o seu SoS para ser reusado também. Ele cadastrou todos os seus CS's e serviços do mesmo jeito que Suelen e, posteriormente, cadastrou seu SoS informando seu respectivo tipo e modelos que explicam o seu propósito e o funcionamento. Após isso, ele adicionou todos os seus CS's no catálogo de sistemas confiáveis (*CatalogOfTrustableSystems*), e enviou um convite para o CS de Suelen, pois ele também é um integrante. Como uma das características dos integrantes do SoS é a independência gerencial, o CS de Suelen só pode ser “ativado” no catálogo com a permissão dela, em outras palavras, só ela pode mudar o *status*. Manuel também possui as características gerenciais no repositório, isto é, ele pode mudar o *status* de seus *assets*, remover, adicionar *stakeholders* no SoS (Por exemplo a Suelen), entre outros.

Por fim, apresenta-se o *User* representado por Jorge, um usuário que apenas está interessado em procurar serviços, CS ou SoS. Dado isso, Jorge possui uma aplicação e precisa implementar um serviço de mensagem urgentemente para completar seu projeto. Portanto, ele

cadastro no repositório e encontrou um serviço de um dos CS's de Manuel, que corresponde a seu requisito e decidiu utilizá-lo. Observe que a finalidade do *User* é bem simples e também foi abordada pelo Manuel, um vez que ele apresentou características de um *SoSDeveloper*, *CSOwner* e de um usuário comum.

Um ponto importante a mencionar é que a abordagem não mantém ou contém serviços, CS's e SoS's fisicamente no repositório, ou seja, os recursos são recuperados por meio das informações fornecidas pelo *CSOwner* e *SoSDeveloper*. Assim, os *assets* são registrados e uma referência é fornecida a localização física de cada um. Portanto, o repositório visa auxiliar na sua organização e recuperação. Em relação à documentação (modelos que ajudam a entender a finalidade dos *assets*), os usuários também devem indicar uma referência que aponte onde os modelos estão, por exemplo, repositórios como Github ¹ e GoogleDrive ².

Em seguida, apresentam-se os principais casos de uso da abordagem (Figura 21) e o glossário com suas respectivas descrições (Tabela 6).

Tabela 6 – Glossário dos Casos de Uso

Ator	Caso de Uso	Descrição
CSOwner	Registrar CS	Esse caso de uso é responsável pelo cadastro de novos CS's no repositório. O CSOwner pode adicionar um ou mais CS e disponibilizá-los no repositório.
CSOwner, SoSDeveloper	Adicionar Modelos	Cada serviço, CS e SoS pode ser detalhado por qualquer tipo de modelo (Caso de Uso, Conceitual, Classes, entre outros). A documentação tem o propósito de auxiliar quem está buscando o asset, de forma que ajude a compreender o funcionamento e sanar possíveis dúvidas referente ao asset.
CSOwner	Gerenciar Serviços	O CSOwner pode gerenciar seus serviços a qualquer momento. Ele pode trocar o status (ativado, desativado, revisão ou não publicado), mudar a versão (evolução), remover ou adicionar modelos, entre outros.
CSOwner	Adicionar Serviços	Cada CS pode fornecer um ou mais serviços para o repositório.
CSOwner	Gerenciar CS	O CSOwner pode gerenciar seus CS's a qualquer momento. Ele pode trocar o status (ativado, desativado, revisão ou não publicado), mudar a versão (evolução), remover ou adicionar modelos, entre outros.
SoSDeveloper	Adicionar SoS	O SoSDeveloper adiciona um novo SoS. O SoSDeveloper pode adicionar um ou mais SoS e disponibilizá-los no repositório.
SoSDeveloper, CSOwner	Adicionar CS Confiável	O SoSDeveloper convida CS (que confia e tem capacidades de seu interesse) para formar um SoS. Consequentemente, o CSOwner decide se confia no SoSDeveloper e se tem interesse em aceitar o convite. É importante ressaltar que o CSOwner tem o poder de retirar seu CS a qualquer momento.
SoSDeveloper, CSOwner	Ver Catálogo de Confiáveis	O SoSDeveloper e CSOwner podem visualizar a qualquer momento um catálogo de CS's confiáveis do SoS. Esse catálogo também pode auxiliar na decisão do CSOwner, isto é, se aceita ou não o convite.

¹ <https://github.com>.

² <https://www.google.com/drive>.

User	Procurar Asset	O modelo do repositório foi inspirado na RAS. Dessa maneira, os serviços, CS e SoS apresentam atributos específicos (herança) com o objetivo de fomentar o reúso.
User	Procurar CS	O User pode fazer uma busca específica por CS.
User	Procurar Serviços	O User pode fazer uma busca específica por serviços.
User	Procurar SoS	O User pode fazer uma busca específica por SoS.
Fora do Escopo		
SoSDeveloper	Implementar Comportamento Emergente	O repositório não aborda a parte do comportamento emergente. Porém, ele pode ser implementado e oferecido fora do escopo dessa abordagem.
User	Fazer Solicitação de Implementação	O User pode fazer uma solicitação de uma nova capacidade que não encontrou em um determinado CS/SoS. A solicitação deve ser enviada diretamente ao responsável.

Fonte: Elaborada pelo autor.

3.2.4 Cenário de Uso

Para esclarecer o uso da abordagem proposta, é apresentado a seguir um exemplo com base em um cenário em que uma pesquisa é feita no repositório. Nesse exemplo, considera-se que o SoS e vários CS's foram adicionados ao repositório, bem como seus serviços fornecidos.

O padrão de apresentação segue o mesmo da RAS. Dessa forma, o nome e a descrição breve devem ser os primeiros elementos a serem visualizados na busca de um serviço, CS ou SoS. A pesquisa pode ser realizada usando os elementos adicionais da adaptação do padrão RAS, como classificação, contexto, palavra-chave, entre outros.

Na Figura 22, apresenta-se um cenário típico de uso. Nesse cenário, um usuário busca por um *asset* de acordo com seus requisitos. Os passos são:

1. O usuário precisa de um comportamento que não tem na sua aplicação padrão. O objetivo é encontrar um *asset* no repositório que corresponde aos requisitos desse comportamento. Após a busca, o usuário recebe o retorno do repositório;
2. O usuário analisa o *asset* e verifica se corresponde aos requisitos;
3. Com base na análise, três decisões podem ser feitas;
4. Se o *asset* corresponde aos requisitos, ele retorna para o usuário e uma análise mais profunda pode ser feita. Por exemplo: (i) se o *asset* é um CS, ele pode verificar quem está disponibilizando e quais serviços esse CS fornece, assim como os modelos (caso de uso, diagramas, código fonte, entre outros) cadastrados; (ii) se o *asset* é um serviço, ele pode verificar que CS está disponibilizando e que interfaces são oferecidas, assim como os modelos que auxiliam a entender o uso do serviço; e (iii) se o *asset* é um SoS, ele pode verificar qual é sua missão, quem é o *SoSDeveloper* e quem são os CS's envolvidos, assim como os modelos que facilitam o entendimento do propósito do SoS;

Figura 21 – Casos de Uso da Abordagem

Fonte: Elaborada pelo autor.

5. Se o *asset* não corresponde aos requisitos, ele pode fazer outra busca (mudando os critérios de busca, visto que o repositório utiliza e fornece vários meios para enriquecer e facilitar a busca);
6. Se o *asset* não corresponde aos requisitos ou não está disponível, ele pode entrar em contato com o responsável diretamente. Essa funcionalidade é opcional, porque o *CSOwner* ou *SoSDeveloper* podem não estar interessados em desenvolver capacidades além do seu domínio, então eles podem omitir o e-mail para contato no cadastro. Em caso de contato

Figura 22 – Cenário Típico

Fonte: Elaborada pelo autor.

positivo, o usuário faz a solicitação de desenvolvimento ou de permissão de uso;

7. *CSOwner* ou *SoSDeveloper* podem atender o pedido do usuário e disponibilizar as capacidades; e
8. O gerenciamento e operação dos CS's estão fora do escopo do SoS. Por outro lado, o *CSOwner* pode gerenciar os serviços a serem fornecidos aos SoS, adicionando-os ao repositório, enquanto o *SoSDeveloper* é responsável pela realização de novos SoS. Nesse caso, ele pode fornecer novos recursos que exigem serviços novos. Por fim, as informações sobre os CS's e os SoS's estarão disponíveis no repositório, bem como os serviços fornecidos pelos CS's.

3.3 Avaliação da Abordagem

3.3.1 Questionário

Após definir o modelo conceitual, glossário e atores, foi realizada uma enquete com especialistas da área SoS com o objetivo de avaliar se os conceitos estavam coerentes e corretos do ponto de vista de quem trabalha com SoS. A enquete foi disponibilizada por meio do Google-Forms. As perguntas respondidas pelos especialistas podem ser visualizadas no Apêndice B.

O formulário apresentava a introdução da proposta e os seguintes documentos: detalhes da MBA4SoSI, modelo conceitual (Figura 20), glossário do modelo conceitual (Tabela 5), casos de uso (Figura 21), glossário dos casos de uso (Tabela 6) e um cenário típico de uso (Figura 22).

Esse questionário foi conduzido paralelamente ao desenvolvimento da abordagem, por isso, as considerações foram analisadas e alguns refinamentos foram feitos.

3.3.2 Análise dos Resultados

Após o envio do formulário, oito sujeitos participaram e responderam às perguntas com suas respectivas considerações, levando em média o tempo de uma hora.

O conhecimento no domínio de SoS foi considerado essencial para cobrir o escopo do trabalho, por isso foram convidadas a responder a enquete apenas pessoas que já se sabia serem da área, seja como desenvolvedor, pesquisador ou ambos. Na Figura 23 são apresentadas as experiências e o nível de conhecimento em SoS de cada participante.

Figura 23 – Nível de Conhecimento em SoS

Fonte: Dados da pesquisa.

Com a primeira pergunta, objetivou-se verificar se o modelo apresentado estava de acordo com o propósito da abordagem. **1a) - O modelo conceitual está de acordo com o objetivo de criar um repositório para disponibilizar e buscar ativos (Serviços, Constituintes e SoS)?**. Nota-se (Figura 24) que as respostas foram positivas, sendo que um sujeito concordou fortemente, quatro concordaram e três responderam que não tinham certeza. Opcionalmente, os sujeitos poderiam apresentar suas considerações/sugestões na pergunta **1b**. As sugestões apontadas e que deram ou não origem a mudanças e refinamentos na abordagem proposta são apresentadas na Tabela 7. Nos casos em que o autor não aceitou a sugestão do participante, foi apresentada na tabela uma breve justificativa para tal. Deve-se também notar que os modelos e tabelas apresentados anteriormente já se referem à versão refinada da abordagem, após terem sido incorporadas as sugestões decorrentes da enquete.

Figura 24 – Respostas da Pergunta 1a

Fonte: Dados da pesquisa.

Tabela 7 – Sugestões da Pergunta 1b

Consideração do Participante	Sugestão Aceita?	Nível SoS
P2 - Stakeholder pode ter mais de um objetivo.	Sim.	Doutorando (Alto).
P2 - Os SoS's podem ter mais de um tipo de classificação.	Não. Os tipos apresentados no modelo possuem forte embasamento teórico e foram modelados conforme as referências utilizadas no glossário.	Doutorando (Alto).
P2 - Faltou os conceitos de interfaces e protocolos dos serviços.	Não. As interfaces são definidas no atributo <i>reference</i> da entidade <i>Service</i> . Já os protocolos, podem informados por meio de modelos (entidade <i>Model</i>).	Doutorando (Alto).
P8 - O <i>User</i> pode assumir perfis diferentes. Nesse caso, o modelo deveria apresentar isso.	Sim.	Doutorando (Regular)
P8 - Colocar a denotação para as entidades da RAS.	Não. A RAS (classes e atributos) foi utilizada como inspiração para a construção do modelo e não seguem exatamente as definições/relacionamentos do padrão, até porque não se aplica ao propósito do repositório. Então, foi decidido não colocar um pacote referente a RAS, pois não é exatamente o metamodelo do padrão.	Doutorando (Regular)
P8 - Entidade com as informações do desenvolvedor do CS.	Não. As informações básicas necessárias estão presentes.	Doutorando (Regular)
P8 - Remover a entidade <i>SystemOfSystemType</i> porque é redundante.	Não. Essa entidade foi utilizada com base em outro trabalho validado e segue os padrões da literatura.	Doutorando (Regular)

Fonte: Elaborada pelo autor.

A segunda pergunta teve por objetivo analisar o uso da RAS na abordagem. **2a) - A RAS (Reusable Asset Specification) é utilizada com o objetivo de fomentar o reúso de ativos (assets), por isso, alguns elementos foram adaptados e utilizados no modelo desta proposta. Na sua opinião, essa adaptação/inspiração na RAS auxilia a promover o reúso de recursos na abordagem?** Três sujeitos concordaram fortemente, dois concordaram, dois responderam que não tinham certeza e um não tinha conhecimento suficiente sobre a RAS para responder (Figura 25). Na pergunta **2b**, poderiam ser apresentadas sugestões, porém as respostas foram direcionadas a dúvidas sobre a RAS e quais foram os elementos adaptados, além de elogios, por isso não são aqui apresentadas.

A terceira pergunta teve por objetivo verificar o entendimento dos casos de uso. **3a) - Os**

Figura 25 – Respostas da Pergunta 2a

Fonte: Dados da pesquisa.

casos de uso estão de acordo com o objetivo (criar um repositório) da abordagem?. Nessa pergunta, dois candidatos concordaram fortemente, três concordaram, dois não tinham certeza e um discordou (Figura 26). Opcionalmente, os sujeitos poderiam apresentar suas considerações/-sugestões na pergunta **3b**. As considerações/sugestões são apresentadas na Tabela 8.

Figura 26 – Respostas da Pergunta 3a

Fonte: Dados da pesquisa.

A quarta pergunta objetivou verificar os atores da proposta. **4a) - De acordo com os casos de uso, os papéis de cada ator (User, CSOwner, SoSDeveloper) estão bem definidos?**. Um sujeito concordou fortemente, três responderam que concordam, três não tinham certeza e um discordou (Figura 27). Opcionalmente, os sujeitos poderiam apresentar suas considerações/-sugestões na pergunta **4b**. As considerações/sugestões são apresentadas na Tabela 9.

A quinta pergunta também objetivou verificar os atores. **5a) - Na sua opinião, existe algum ator que não foi apresentado, mas deveria estar presente na abordagem?**. Cinco

Tabela 8 – Considerações da Pergunta 3b

Consideração do Participante	Explicação do Proponente	Nível SoS
P2 - O caso de uso “implementar comportamento emergente” vai contra o conceito de emergência, que significa um comportamento inesperado do SoS. Assim, como é pretendido implementar um comportamento que não se tem conhecimento da sua existência?	O proponente concorda que o comportamento emergente pode surgir por meio do sinergismo dos CS's e muitas vezes é inesperado. Porém, isso não é regra. A definição utilizada neste trabalho é de que um comportamento emergente é único e não é alcançado por nenhum CS individualmente. Porém, de alguma forma esse comportamento pode envolver a implementação de algum código, seja por meio de um sistema que exerce o papel de coordenador do SoS, no caso de SoS do tipo direcionado ou reconhecido, ou por meio de código inserido nos CS's quando se trata de um SoS do tipo colaborativo. Entende-se também que é possível ocorrer comportamentos inesperados, tanto nos SoS direcionados, reconhecidos e colaborativos, mas principalmente nos SoS virtuais. Ademais, como já foi mencionado, o foco principal deste trabalho não é no desenvolvimento dos comportamentos emergentes do SoS, mas em facilitar o reúso durante o desenvolvimento de novos SoS.	Doutorando (Alto)
P1 - Não está claro o objetivo do caso de uso “Fazer solicitação de implementação”, já que um SoS não é implementado mas ele é o resultado da integração e interoperabilidade de CS.	“Fazer solicitação de implementação” significa fornecer uma nova capacidade, isto é, um novo comportamento emergente ou um novo serviço dos CS's.	Doutorando (Muito Alto)
P8 - Não está claro quais casos de uso ficam fora do escopo do repositório.	Consideração aceita. Casos de uso foram coloridos e possuem comentários.	Doutorando (Regular)

Fonte: Elaborada pelo autor.

Figura 27 – Respostas da Pergunta 4a

Fonte: Dados da pesquisa.

sujeitos responderam que não e três responderam que sim (Figura 28). Em relação a pergunta opcional **5b**, as respostas pediam a presença de um arquiteto, porém como explicado na tabela anterior, foi resolvido não aceitar a sugestão.

A sexta pergunta teve por objetivo verificar o glossário do modelo conceitual. **6a) - Cada elemento está bem definido e detalhado no glossário do modelo conceitual?** Um sujeito concordou fortemente, cinco concordaram e dois discordaram (Figura 29). A pergunta **6b** apresentou apenas um questionamento: “Qual a diferença das entidades *ServiceType* e *Context*”. Para sanar essa dúvida, é preciso ter conhecimento sobre a RAS, pois o *Context* é uma entidade que apresenta o domínio do ativo e as categorias de contexto, já o *ServiceType* representa o

Tabela 9 – Considerações da Pergunta 4b

Consideração do Participante	Explicação do Proponente	Nível SoS
P2 - Considerando que um SoS é raramente implementado, mas integrado, o papel do SoSDeveloper fica fora do contexto de SoS. Faz mais sentido usar papéis de SoSArchitect, SoSManager.	O SoSDeveloper é representado por uma pessoa experiente no contexto de SoS, ou seja, um engenheiro, integrador ou arquiteto. Além disso, esse ator foi baseado em outros trabalhos válidos, portando o nome do ator permanece.	Doutorando (Alto)
P2/P4 - Falta o CSArchitect.	O CSArchitect aborda aspectos de desenvolvimento do SoS e, ele até pode existir fora do contexto da abordagem. Porém, para o repositório, um desenvolvedor ou proprietário (CSOwner) é suficiente.	Doutorando (Alto)/Mestrando (Alto)

Fonte: Elaborada pelo autor.

Figura 28 – Respostas da Pergunta 5a

Fonte: Dados da pesquisa.

agrupamento de serviços semelhantes.

A sétima e última pergunta teve por objetivo apresentar o cenário típico de uso do repositório ilustrado na Figura 22. **7a) - O caso de uso (Encontrar asset) auxiliou a compreender a abordagem?** Um sujeito concordou fortemente, três concordaram, dois não tinham certeza e dois discordaram (Figura 30). É importante ressaltar que um sujeito discordou porque não encontrou o cenário, porém o link para acesso estava na introdução. O restante das considerações/sugestões da pergunta **7b** são apresentadas na Tabela 10.

A avaliação obtida por meio dessa enquête foi essencial para realizar a implementação do repositório e para auxiliar a definir o escopo do trabalho proposto. Alguns questionamentos foram importantes para os elementos de SoS, visto que foi abordado de maneira simplificada neste projeto, mas que serão válidas em considerações para os trabalhos futuros da MBA4SoSI. Por fim, é importante enfatizar que todas as mudanças aceitas foram realizadas após o término da enquête e a Seção 3.2 já apresenta todas as alterações realizadas.

Figura 29 – Respostas da Pergunta 6a

Fonte: Dados da pesquisa.

Figura 30 – Respostas da Pergunta 7a

Fonte: Dados da pesquisa.

3.4 Considerações Finais

Neste capítulo, foi apresentada a abordagem de mestrado, enfatizando o referencial teórico que foi utilizado como base, o público-alvo e a ideia de funcionamento. Todos esses artefatos foram enviados e avaliados por especialistas da área de SoS, cujo principal objetivo era concretizar a abordagem e o escopo de trabalho, de acordo com as respostas e o *feedback* dos participantes. Vale a pena ressaltar que essa abordagem foi redigida em um artigo científico (NASCIMENTO; FELONI; BRAGA, 2017) e submetida a um evento internacional.

No próximo capítulo, é apresentada a implementação do repositório, seguindo as diretrizes aqui apresentadas. O resultado dessa implementação são módulos que contêm serviços *web services* para cada funcionalidade necessária para seu uso. No decorrer do capítulo, também são apresentados os meios de acesso, a documentação e uma aplicação de exemplo.

Tabela 10 – Considerações da Pergunta 7b

Consideração do Participante	Explicação do Proponente	Nível SoS
P2 - Como o arquiteto e o usuário do framework vão saber quais ativos usar para um determinado objetivo do SoS?	A RAS foi usada como inspiração exatamente para facilitar o reúso e busca dos elementos do repositório. As adaptações realizadas visam auxiliar a busca de SoS existentes, pois todos os elementos (serviços, CS e SoS) são conectados. No padrão RAS isso não estaria exatamente relacionado, porque a RAS não foi feita para a semântica de SoS, mas sim para o contexto de reúso de forma geral, portanto o máximo que o padrão auxiliaria seria como forma de ativos relacionados. Nesse caso, o SoSDeveloper pode usar os atributos/elementos da RAS como critérios de busca. Alguns exemplos de busca possíveis são: (i) O usuário pode buscar qualquer ativo por meio da classificação, ou seja, palavra-chave, contexto, domínio, entre outros; (ii) O usuário pode buscar os serviços de um CS que é do mesmo domínio de seu SoS; e (iii) O usuário pode buscar diretamente as categorias (<i>ServiceType</i>), com intuito de encontrar os serviços e CS's que são correlacionados com o objetivo do SoS.	Doutorando (Alto)
P2 - Como um SoS que já foi integrado pode ser evoluído quando novos objetivos são incluídos?	As características intrínsecas do SoS e CS continuam. O SoS-Developer pode gerenciar seu SoS normalmente, assim como o CSOwner seus CS's e serviços. O repositório ajuda a organizar e disponibilizar os elementos, dessa forma o SoS pode evoluir e, posteriormente os objetivos e informações podem ser alterados no repositório.	Doutorando (Alto)
P8 - Melhorar os fluxos de decisão e saídas com a mesma condição.	Sugestões aceitas. Na versão aqui apresentada os fluxos foram melhorados.	Doutorando (Regular)

Fonte: Elaborada pelo autor.

DETALHES DA IMPLEMENTAÇÃO DO REPOSITÓRIO DA ABORDAGEM

4.1 Considerações Iniciais

Neste capítulo, são apresentados os detalhes da implementação do repositório proposto como parte da abordagem, bem como os meios de acesso a ele e uma aplicação gráfica que facilita a visualização e consumo dos serviços. Portanto, no decorrer deste capítulo, serão respondidas (implicitamente) as seguintes perguntas:

- Como os serviços foram separados?
- Como o repositório foi desenvolvido e é disponibilizado para possíveis contribuidores?
- Qual é a documentação fornecida para os usuários?
- Como os serviços do repositório devem ser consumidos?

O restante do capítulo está organizado da seguinte forma: na Seção 4.2 são apresentados todos os detalhes do repositório, como sua implementação, documentação e descrição dos módulos. Na Seção 4.3 são apresentados os meios de acesso ao repositório, assim como uma aplicação gráfica que foi desenvolvida para elucidar como o repositório pode ser expandido e consumido de várias maneiras. Finalmente, na Seção 4.4 são apresentadas as considerações finais deste capítulo.

4.2 Detalhamento do Repositório

4.2.1 Implementação

O repositório proposto tem quatro elementos fundamentais que gerenciam o acesso e controle dos usuários, nos quais três deles podem ser visualizados como *assets* e artefatos neste trabalho. Por isso, foi decidido separar esses elementos fundamentais em módulos responsáveis pelo funcionamento do repositório.

Os quatro módulos são: (i) *User Managmt*, módulo responsável pelo controle e acesso dos usuários; (ii) *CS Managmt*, módulo responsável pelo gerenciamento dos CS's disponibilizados pelo *CSOwner*; (iii) *Services Managmt*, módulo responsável pelo gerenciamento dos serviços de cada CS cadastrado no repositório; e (iv) *SoS Managmt*, módulo responsável pelo gerenciamento dos SoS's disponibilizados pelo *SoSDeveloper*. Dessa forma, o modelo apresentado no Capítulo 3 foi separado nesses quatro módulos, nos quais os três últimos foram inspirados na RAS, e usam entidades e atributos para melhor classificar e encontrar os ativos no repositório.

Cada módulo contém uma coleção de serviços REST(FIELDING, 2000) que controlam/gerenciam as funcionalidades de seu escopo, apesar de compartilharem o mesmo banco de dados para realizarem o armazenamento e recuperação das informações, que no caso dessa implementação, utiliza-se um banco de dados relacional. Quando um serviço de algum módulo necessita de outro que não é de seu escopo para ser completado, o módulo consome o respectivo serviço do outro módulo e posteriormente retorna a resposta para o usuário. Além disso, todas as solicitações foram definidas de acordo com o padrão *stateless* do estilo arquitetural REST, em que cada pedido do cliente (usuário que deseja consumir algum serviço) para o servidor (módulo) deve conter todas as informações necessárias para entender o pedido e não pode tirar proveito de qualquer contexto armazenado no servidor.

O repositório foi implementado na plataforma Java EE7, com a API JAX-RS para desenvolvimento dos *web services* e o *framework* hibernate para o mapeamento dos atributos da base de dados. É importante ressaltar que todo mapeamento das entidades foi realizada diretamente pelas anotações do hibernate, e não foi utilizado nenhum outro programa para gerenciar ou criar o banco de dados.

Para auxiliar a organização das classes, foram utilizados nove pacotes (Figura 31) no desenvolvimento do repositório. Eles são explicados a seguir:

- `com.serviceorientedsos.controllers`: Esse pacote possui os controladores das classes *Asset*, *Constituent*, *Model*, *Service*, *SoS* e *User*. Esses controladores possuem os métodos necessários para persistência em banco de dados das principais entidades utilizadas no repositório;
- `com.serviceorientedsos.dao`: Esse pacote possui a interface com os métodos que cada

controlador deve implementar;

- `com.serviceorientedsos.entities`: Esse pacote tem as entidades definidas no modelo mapeadas para o banco de dados. Todas as notações e relacionamentos são de acordo com o *framework* hibernate e, as *queries* para persistir no banco de dados são descritas em forma de *NamedQuery*;
- `com.serviceorientedsos.filter`: Esse pacote possui a classe responsável pelo filtro de cada requisição feita ao repositório;
- `com.serviceorientedsos.services`: Esse pacote possui quatro classes referentes aos serviços REST utilizados, ou seja, os quatro módulos. Essas classes definem os serviços e o *path* de acesso de cada um. Além disso, cada método possui anotações da documentação do swagger que são explicadas na Subseção 4.2.2;
- `com.serviceorientedsos.services.exceptions`: Esse pacote possui uma classe que mapeia todos os erros ocorridos durante cada solicitação dos serviços. O intuito dessa classe é auxiliar o desenvolvedor com os erros que podem ocorrer durante o uso dos módulos;
- `com.serviceorientedsos.services.util`: Esse pacote possui classes com métodos que acessam alguns serviços de cada módulo. Como explicado acima, alguns módulos necessitam de outros serviços (de outros módulos) para serem completados, por isso, aqui implementa-se o acesso desses respectivos serviços;
- `com.serviceorientedsos.util`: Esse pacote possui classes referentes a configuração do ambiente do repositório. Dessa maneira, a classe de controle do *poll* do banco de dados, configuração do hibernate, configuração do swagger e a configuração do e-mail são disponibilizadas aqui; e
- `com.serviceorientedsos.validators`: Esse pacote possui as classes que validam as solicitações feitas pelo usuário. Por exemplo, quando o *CSOwner* faz a solicitação de cadastro de um novo constituinte, o módulo *CS Managmt* passa essa solicitação para sua classe de validação, na qual os dados são verificados e só depois são persistidos no banco de dados.

Figura 31 – Pacotes da Implementação do Repositório

Fonte: Elaborada pelo autor.

No caso dos serviços REST, todas as requisições são realizadas por meio dos HTTPVerb. Já o retorno da solicitação pode ser visualizado por meio dos códigos de *status* padronizados no protocolo HTTP. Na Tabela 11 são apresentados os verbos e códigos utilizados no repositório.

Tabela 11 – HTTPVerb e Códigos de Status do Repositório

HTTPVerb	
POST	Criação dos elementos.
GET	Leitura/Retorno dos elementos.
PUT	Update dos atributos de um elemento.
DELETE	Remover do repositório.
Códigos de Status	
2xx	Requisição realizada com sucesso.
4xx	Erro na requisição. (ClientSide)
5xx	Erro na requisição. (ServerSide)

Fonte: Elaborada pelo autor.

A respeito da distribuição do repositório, é de grande interesse deixá-lo como uma solução em código aberto, de forma que incentive pesquisadores e engenheiros de software a auxiliar na evolução da abordagem. Portanto, todo o código implementado pode ser visualizado no github¹ do proponente. Por fim, na Figura 32 é apresentada uma visão geral do repositório.

Figura 32 – Visão Geral do Repositório

Fonte: Elaborada pelo autor.

4.2.2 Documentação

Os engenheiros interessados em utilizar o repositório devem saber primeiramente como os serviços funcionam e como devem consumi-los. Para isso, o repositório foi desenvolvido

¹ <<https://github.com/diegolnasc/ServiceOrientedSoS>>

visando fornecer dois meios de documentação para os usuários, a descrição de serviços HTTP (WADL) e a *OpenAPI Specification*, por meio do *framework* Swagger².

O Swagger provê uma API que é bastante utilizada para a descrição de serviços REST, apresentando uma grande facilidade de documentar os serviços diretamente pelo código ou por meio de seu editor *online*. Além disso, a documentação final é totalmente *human-readable*, diferentemente da descrição WADL.

Na Figura 33 é ilustrada uma amostra da documentação final do repositório proporcionado pelo *framework* Swagger. Outro ponto importante é que a API apresenta exemplos de como consumir os serviços e tem um mecanismo para geração de cliente/servidor em mais de quarenta linguagens. Por isso, recomenda-se fortemente o uso dessa documentação para o consumo dos serviços.

Figura 33 – Documentação Swagger do Repositório

Fonte: Elaborada pelo autor.

Ambas as documentações são encontradas no github do proponente, além de poderem ser acessadas *online* por meio do *link* do WADL³ ou Swagger⁴. Por fim, um exemplo das documentações é apresentado na Figura 34.

² <<https://swagger.io/>>

³ <<http://sosrepository.me/rest/application.wadl>>

⁴ <<http://sosrepository.me/swagger/index.html>>

Figura 34 – Exemplo da Documentação

```

<?xml version="1.0"?>
<resource path="/constituent">
  <method id="createConstituent" name="POST">
    <request>
      <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="Authorization" style="header" type="xs:string"/>
      <representation mediaType="application/xml"/>
      <representation mediaType="application/json"/>
    </request>
    <response>
      <representation mediaType="text/plain"/>
      <representation mediaType="application/xml"/>
    </response>
  </method>
</resource>
<resource path="/edit">...</resource>
<resource path="/delete">...</resource>
<resource path="/changeState">...</resource>
<resource path="/search">...</resource>
<resource path="/searchAll">...</resource>
<resource path="/searchAllByStatus">...</resource>
<resource path="/searchMyConstituents">...</resource>
<resource path="/searchMyConstituentsByStatus">...</resource>
<resource path="/addModel">...</resource>
<resource path="/editModel">...</resource>
<resource path="/deleteModel">...</resource>
<resource path="/searchModels">...</resource>
<resource path="/searchAllMyModels">...</resource>
<resource path="/searchByKeyword">...</resource>
<resource path="/searchByReuseScope">...</resource>
<resource path="/searchByContext">...</resource>
<resource path="/searchByDomain">...</resource>
</resource>

```

Módulo: CS Managmt

Serviço: Registrar CS

Restante dos Serviços

(a) Exemplo de um Serviço pelo WADL

POST /constituent/create Create a new constituent system

Insert the constituent object and the required fields

Parameters

Try it out

Name	Description
constituent required	ConstituentSystem object that will be added.
(body)	<div>Example Value</div> <div>Model</div> <div>ConstituentSystem { ... }</div>

Responses

Response content type: application/xml

Code	Description
201	CSManaget-> New constituent created
400	CSManaget-> error: Details

(b) Exemplo de um Serviço Swagger

4.2.3 Módulo *User Managmt*

O módulo *User Managmt* é responsável principalmente pelo gerenciamento (*Create, Read, Update, Delete – CRUD*) dos usuários. Nesse sentido, o acesso ao repositório só pode ser realizado por usuários registrados, ou seja, qualquer solicitação feita, independentemente do módulo, só pode ser completada por um usuário cadastrado. Caso contrário, o módulo *User Managmt* vai filtrar a solicitação e enviar o código 401 (Não autorizado) para o solicitante. Criar conta, buscar conta, recuperar a senha ou ver a documentação dos módulos do repositório, são os únicos quatro serviços para os quais o módulo permite acesso para pessoas não cadastradas. Os serviços fornecidos pelo módulo *User Managmt* são apresentados no Apêndice C. No Código-fonte 1, é apresentado um exemplo de um dos serviços desse módulo.

Código-fonte 1 – Serviço para Cadastrar um Usuário

```
1: @ApiOperation(  
2:     value = "Create a new user",  
3:     notes = "Insert the user object and the required fields",  
4:     httpMethod = "POST",  
5:     consumes = "application/xml, application/json",  
6:     produces = "application/xml, text/plain"  
7: )  
8: @ApiResponse(value = {  
9:     @ApiResponse(  
10:         code = 201,  
11:         message = "UserManagmt-> New user created"  
12:     ),  
13:     @ApiResponse(  
14:         code = 400,  
15:         message = "UserManagmt-> error: Details"  
16:     )}  
17: )  
18: @POST  
19: @Consumes({MediaType.APPLICATION_XML, MediaType.  
    APPLICATION_JSON})  
20: @Produces({MediaType.TEXT_PLAIN, MediaType.APPLICATION_XML})  
21: @Path("/create")  
22: public Response createUser(@ApiParam(name = "user", value = "  
    User object that will be added.", required = true) User user  
    ) {  
23:     try {  
24:         user = userChecker.isUserValidToCreate(user);  
25:         userController.save(user);
```

```
26:         return Response.status(Response.Status.CREATED).entity(
            "UserManagmt-> New user created: " + user.getLogin()).build
            ();
27:     } catch (Exception e) {
28:         return Response.status(Response.Status.BAD_REQUEST).
            entity("UserManagmt-> Error: " + e.getMessage()).build();
29:     }
30: }
```

Esse código-fonte apresenta a estrutura do serviço de cadastro de usuários no repositório. As primeiras linhas (1-17) denotam todas as informações desse serviço, visto que essa especificação é referente à documentação utilizada. Já as linhas 22-30 são responsáveis pela lógica realizada para o cadastro do usuário, na qual a linha 24 faz a validação das informações (entidade *user*) enviadas, a linha 25 realiza a persistência da entidade no banco de dados e a linha 26 retorna (XML ou TEXT_PLAIN) o código 201, que significa que o cadastro foi efetuado com sucesso. Caso a entidade *user* não passe na validação, a linha 26 retorna o código 401 e o respectivo erro para o usuário. Vale a pena ressaltar que todos os serviços possuem uma estrutura semelhante a essa.

Na perspectiva dos usuários cadastrados, todas as solicitações devem ser enviadas com as credenciais de acesso no cabeçalho da solicitação do XML ou JSON, em forma da autenticação básica do HTTP (encode base64). O *User Managmt* vai autenticar essas credenciais por meio do serviço “*search*” e, posteriormente, encaminhará a solicitação para o devido serviço. Dessa forma, qualquer solicitação para consumo de algum serviço do repositório, passará primeiramente pelo serviço de autenticação do módulo *User Managmt*. Como forma de segurança e proteção das contas, o repositório faz uso do protocolo de segurança TLS/SSL, em que é solicitada a autenticação do certificado do cliente também, mais conhecido como *Two-way SSL*. Por fim, o fluxo de uma solicitação é demonstrado na Figura 35.

4.2.4 Módulo CS, Services e SoS Managmt

Todos os módulos têm funcionalidades e responsabilidades específicas de seu escopo, entretanto, em algumas solicitações é necessário a comunicação entre eles. Um exemplo disso é apresentado na Figura 36, na qual um usuário busca os serviços de um determinado CS. Nesse exemplo, os passos executados são: (i) o usuário consome o serviço do *Services Managmt* informando suas credenciais e o nome do CS, por meio de um arquivo XML ou JSON; (ii) o módulo *Services Managmt* envia as credenciais de quem fez a solicitação para o *User Managmt*, com o propósito de verificar se o usuário é cadastrado no repositório; (iii) após verificadas as credenciais do usuário, o *Services Managmt* faz uma solicitação para o *CS Managmt* com o intuito de verificar se o CS existe; (iv) após validado o CS, o *Service Managmt* busca os seus serviços; e (v) por fim, o *Services Managmt* retorna os serviços para o usuário.

Figura 35 – Fluxo de Solicitações

Fonte: Elaborada pelo autor.

Figura 36 – Exemplo de Busca dos Serviços de um CS

Fonte: Elaborada pelo autor.

Esses e demais detalhes podem ser consultados na documentação e durante o uso do repositório. Uma breve explicação de cada módulo é apresentada a seguir:

- *CS Managmt*: Esse módulo é responsável pelo gerenciamento (CRUD) e demais funções de busca dos CS's no repositório. Todos os serviços desse módulo são apresentados no Apêndice D;
- *Services Managmt*: Esse módulo é responsável pelo gerenciamento (CRUD) e demais funções de busca dos serviços no repositório. Todos os serviços desse módulo são apresentados no Apêndice E; e

- *SoS Managmt*: Esse módulo é responsável pelo gerenciamento (CRUD) e demais funções de busca dos SoS's no repositório. Todos os serviços desse módulo são apresentados no Apêndice F.

Nota-se que é possível utilizar os serviços observando apenas o código de *status* do retorno, porém, para facilitar ainda mais o uso, cada módulo foi desenvolvido com o intuito de auxiliar o pesquisador/engenheiro. Assim, todas solicitações também retornam uma mensagem informando se o pedido foi concluído ou se ocorreu algum erro, bem como avisa se os atributos necessários para concluir foram enviados. Por fim, é importante enfatizar que todos os serviços dos módulos foram testados diretamente por invocação e/ou por ferramentas de linha de comando, como as mencionadas na Seção 4.3.

4.3 Utilização do Repositório

Conforme já descrito anteriormente, o repositório foi desenvolvido com base no paradigma de orientação a serviços. Isso foi planejado, desde o início, para facilitar seu uso, de forma que os usuários que desejam participar decidam qual a melhor maneira de utilizá-lo. Nas subseções seguintes são descritas as principais formas de utilização do repositório.

4.3.1 APIs e Linha de Comandos

Uma das maneiras mais simples de utilização do repositório é por meio de API's de desenvolvimento para cliente REST, como o Postman (Figura 37a). Com essa API, é possível consumir os serviços REST facilmente, pois ela fornece várias opções para o uso, ressaltando principalmente as de autenticação, informações de cabeçalhos e certificados.

Figura 37 – Exemplo de Consumo pelo Postman e cURL

(a) Exemplo de consumo pelo Postman

(b) Exemplo de consumo pelo cURL

Outro método simples de utilização dos serviços é por meio de ferramentas de linha de comandos como cURL (Figura 37b). Essa ferramenta envia e recebe documentos e arquivos para um determinado servidor, utilizando vários protocolos como HTTP, HTTPS, SMTP, SMTPS, FTP, FTPS, entre outros, além de dar suporte para certificados e meios de autenticação.

Do mesmo modo, é possível implementar os serviços diretamente nas aplicações *standalone* dos usuários. Assim, todo o consumo pode ser realizado diretamente pelo desenvolvedor na sua própria aplicação.

4.3.2 Camada de Interface Gráfica com o Usuário

Outra forma de utilização do repositório é por meio de uma aplicação gráfica com o usuário. Para deixar claro como isso pode ser feito, foi desenvolvida uma aplicação gráfica na plataforma Java EE7, por meio do *framework* JSF 2.2. A finalidade dessa aplicação é apresentar como os serviços do repositório podem ser facilmente invocados e disponibilizados para diferentes tipos de usuários, quando uma aplicação customizada é desenvolvida. Deve-se notar que a aplicação não contempla o uso de todos os serviços implementados, mas de uma parte deles (45,71% dos serviços). O restante dos serviços foram implementados e testados por invocação em linha de comando, conforme descrito na Subseção 4.2.4, e sua incorporação à aplicação gráfica pode ser feita em trabalhos futuros.

Na Figura 38 é apresentado o acesso inicial ao repositório, por meio dos serviços do *User Managmt*. Logo após o acesso, algumas funcionalidades podem ser visualizadas no menu lateral (Figura 39), uma vez que elas permitem os acessos aos quatro módulos do repositório.

Figura 38 – Aplicação Gráfica - Login e Cadastro

The image displays two side-by-side screenshots of the SoSRepository web application interface. Both pages have a light blue header with the 'SoSRepository' logo. The left screenshot is the login page, titled 'Faça login para iniciar sua sessão'. It features input fields for 'Insira seu login' and 'Insira sua senha', a 'Lembrar senha' checkbox, a green 'Entrar' button, and links for social login (Facebook, Google+) and password recovery. The right screenshot is the registration page, titled 'Registre uma nova conta'. It includes input fields for 'Nome', 'Email', 'Ocupação', 'login', 'Digite sua senha', and 'Confirme sua Senha', along with a checkbox for 'Permitir solicitação de funcionalidade'. A green 'Registrar' button is present, along with social login links and a link for existing users.

Fonte: Elaborada pelo autor.

Figura 39 – Aplicação Gráfica - Menu lateral

Fonte: Elaborada pelo autor.

Para o módulo *CS Managmt*, é apresentado o cadastro de novos CS's na Figura 40; e demais funcionalidades na guia *Meus Constituintes* (Figura 41). Já os módulos *Services Managmt* e *SoS Managmt* possuem as mesmas funcionalidades com telas semelhantes.

Figura 40 – Aplicação Gráfica - Cadastro de CS

Constituinte <small>Cadastrar ou editar um constituinte</small>			
Novo Constituinte			
UUID:* <input type="text"/>	Host:* <input type="text"/>	<input type="text"/>	
Porta: <input type="text"/>	Nome:* <input type="text"/>	<input type="text"/>	
Descrição curta:* <input type="text"/>	Descrição: <input type="text"/>	<input type="text"/>	
Palavras-chaves:* <input type="text"/>	Escopo do Reúso: <input type="text"/>	<input type="text"/>	
Contexto:* <input type="text"/>	Domínio:* <input type="text"/>	<input type="text"/>	
Status: <input type="text"/>	Versão: <input type="text"/>	<input type="text"/>	
<input type="button" value="Salvar"/>			

Fonte: Elaborada pelo autor.

Na guia *Buscar Assets*, são apresentadas algumas formas de busca dos CS's, serviços e SoS's. Na Figura 42, é apresentada a funcionalidade de busca por serviços.

Figura 41 – Aplicação Gráfica - Meus Constituintes

Lista de Constituintes Encontre seus constituintes e gerencie na plataforma Home

[+ Adicionar novo](#)
[✎ Editar selecionado](#)
[🗑 Deletar selecionado](#)
[📄 Adicionar serviço](#)
[📄 Adicionar modelo](#)
[🔄 Ver detalhes](#)

Procure por palavra-chave:

Selecione	Name ↕	CSHost ↕	Descrição Curta ↕
Você não possui constituintes cadastrados!			

[0 to 0 of 0] ⏪ ⏩ ⏴ ⏵ 15 ⏴ ⏵

Fonte: Elaborada pelo autor.

Figura 42 – Aplicação Gráfica - Busca de Serviços

Lista de Serviços Encontre serviços no repositório Home

[🔍 Buscar Meus Serviços](#)
[🔍 Buscar Todos](#)
[🔍 Buscar Por Palavra-chave](#)
[🔍 Buscar Por Domínio](#)
[🔍 Buscar Por Constituinte](#)

Digite um critério de busca:

Procure por palavra-chave:

Selecione	Name ↕	Tipo do Serviço ↕	Direito de Acesso ↕	Constituinte ↕	CSOwner ↕
Nenhum serviço foi encontrado!					

[0 to 0 of 0] ⏪ ⏩ ⏴ ⏵ 15 ⏴ ⏵

Fonte: Elaborada pelo autor.

Essa aplicação também é disponibilizada no github⁵ do proponente. Contudo, ela não contém todos os serviços que o repositório oferece, mas já contém algumas funcionalidades que permitem o uso e demonstra uma das inúmeras formas que o repositório pode ser utilizado.

4.4 Considerações Finais

Neste capítulo foram apresentadas as informações referentes à implementação e os demais detalhes do repositório, assim como a disponibilização em forma de código aberto.

O foco principal do repositório é englobar todos os usuários e pesquisadores que desejam disponibilizar serviços para serem reutilizados, em uma forma organizada e documentada. Em relação a empresas ou grupos privados, existe a possibilidade de adequação do repositório para tais usos. Dessa forma, os *assets* poderiam ser disponibilizados no repositório, mas teriam um termo ou uma licença de uso, de acordo com o critério de cada fornecedor. Outro caso possível é a disponibilização de uma instância privada para uso próprio, uma vez que a abordagem utilizada para a implementação da segurança é a TLS/SSL na forma de *Two way*. Em suma, uma instância privada pode ser criada facilmente. O desenvolvedor dela só precisa colocar um certificado no servidor e adicionar como confiável apenas os certificados do cliente que ele deseja permitir acesso.

Outro ponto são os órgãos governamentais que geralmente possuem sistemas verticais na cidade ou país (como serviços de tráfego, hospitais ou estacionamento). Esses sistemas poderiam fornecer serviços úteis para o restante da população, por meio de um repositório de fácil acesso. De fato, o repositório pode ser adaptado e visualizado de diferentes maneiras, de forma que seja útil não só para a comunidade de pesquisa, mas também para empresas e setores privados.

⁵ <<https://github.com/diegolnasc/DMasterProject>>

CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

5.1 Visão Geral da Pesquisa

A busca por qualidade e melhoria no desenvolvimento de software está abrindo novas portas no âmbito de pesquisa em engenharia de software. Um exemplo disso são os sistemas convencionais, que geralmente são planejados por um ou mais *stakeholders* e preocupam-se em solucionar problemas e/ou aperfeiçoar as atividades que ocorrem apenas em seu escopo. No entanto, há sempre a necessidade de desenvolver novas funcionalidades, e isso resulta na busca por outros sistemas que já forneçam esses recursos. Além disso, os *stakeholders* desejam fazer com que seus sistemas, muitas vezes, desenvolvidos em plataformas heterogêneas, interoperem de forma prática e sem perder controle gerencial de cada um, alcançando comportamentos emergentes não previstos durante o desenvolvimento, o que leva ao conceito de SoS.

Diante disso, neste trabalho de mestrado, procurou-se propor uma solução para parte desse problema, no contexto de SoS, que é um campo de pesquisa emergente que tem ganhado atenção de diversos setores, desde o governo e indústria, até a área acadêmica. Portanto, uma abordagem foi apresentada com o intuito de facilitar o reúso de capacidades já desenvolvidas, por meio de um repositório orientado a serviços.

Esse repositório apresenta os conceitos de reúso inspiradas na RAS e aborda o contexto de SoS, visando não só facilitar o processo de desenvolvimento, por meio de uma busca organizada, mas também contribuir com engenheiros e pesquisadores que trabalham no campo de interoperabilidade de sistemas.

5.2 Contribuições

A contribuição principal deste trabalho de mestrado foi a apresentação de uma abordagem para facilitar o reúso de software no contexto de SoS. A abordagem é composta por um modelo conceitual detalhado acompanhado de respectivo glossário, lista de atores e diretrizes sobre como cada ator atua no processo de desenvolvimento e um repositório de *assets* relativos a SoS, CS's e serviços. Como contribuições secundárias, destacam-se:

- Contribuição para o campo de reúso de software e SoS, disponibilizando um repositório em forma de código aberto para modificações e criações de instâncias próprias;
- Implementação de uma aplicação gráfica (também em código aberto) com o propósito de facilitar o uso do repositório; e
- Identificação de desafios no campo de SoS por meio de um MS.

5.3 Dificuldades e Limitações

A área de SoS tem muitos conceitos, definições e classificações de diferentes pesquisadores. Isso faz com que o processo de construção de um modelo seja extremamente exaustivo e desafiador, resultando em inúmeras reuniões entre o autor, orientadora e demais interessados, em que cada parte já definida podia precisar ser modificada ou removida. Um exemplo claro disso pode ser observado na avaliação conduzida neste trabalho, onde alguns participantes levantaram questionamentos sobre como o comportamento emergente funciona, e porque não foi abordado o dinamismo entre os tipos de SoS. Nessa perspectiva, o problema propende-se a ficar mais complexo, pois esses questionamentos também são abordados na literatura e são apresentados de diferentes maneiras. Alguns pesquisadores até questionam a existência de alguns tipos de SoS. Em nível de implementação isso torna-se ainda mais desafiador, já que não é possível saber ao certo os tipos e atributos que um SoS e seus CS's devem ter. Para que fosse possível a conclusão desta dissertação, foi necessário estabelecer um ponto de parada e deixar como trabalhos futuros o refinamento da abordagem.

Outro desafio ocasionado por isso é relacionado à avaliação do trabalho, visto que ela precisa ser conduzida por pesquisadores experientes no contexto de SoS e de diferentes áreas (arquitetura, reúso, qualidade, entre outros). Desse modo, é muito desafiador validar uma abordagem, quando as visões sobre a concepção de um SoS de cada pesquisador é diferente, sem mencionar a dificuldade de encontrar pessoas qualificadas para analisar uma abordagem nesse âmbito de pesquisa. Após implementado o repositório, a avaliação fica mais restrita ainda, por causa da falta de CS's e SoS's disponíveis para o cadastro. Consequentemente, o repositório é totalmente dependente da contribuição de outros usuários/pesquisadores que desejam compartilhar recursos ou incentivar o crescimento da área. Somente após isso será

possível avaliar melhor se a abordagem é de fato efetiva e útil para a melhoria do reuso no contexto de SoS.

Um outro ponto a ser observado é que o sucesso da abordagem depende da participação e interação da comunidade. Dessa forma, algumas barreiras organizacionais e de proteção à informação ainda estarão presentes, mas a divulgação e a permissão para reutilizar esses sistemas podem vir a fomentar um crescimento significativo na área de integração de sistemas.

5.4 Trabalhos Futuros

O foco principal desta pesquisa foram os constituintes e os serviços que eles fornecem, todavia, foi tentado abordar ao máximo o contexto de SoS (em nível de implementação) para o escopo de um trabalho de mestrado. Por isso, desde o início foi planejado que essa parte teria uma ênfase maior em trabalhos futuros, além de ser um dos pilares do projeto MB4SoSI, nas quais outros alunos já estão envolvidos. Dessa forma, destacam os seguintes trabalhos futuros:

- Implementar na aplicação gráfica, a invocação aos demais serviços que o repositório fornece: conforme apresentado na Subseção 4.3.2, a aplicação gráfica permite o acesso a 45,71% dos serviços implementados, portanto o restante ainda precisa ser incluído;
- Disponibilizar guias para a criação de instâncias do repositório para grupos privados: conforme mencionado na Seção 4.4, o repositório pode ser facilmente estendido para instâncias privadas, por meio do protocolo TLS/SSL. Dessa forma, guias/roteiros poderiam ser disponibilizadas como exemplo, desde a criação do certificado (no exemplo pode ser utilizado um *self-signed*) até a configuração da aplicação;
- Expandir o repositório para dar suporte à busca por comportamentos emergentes: no momento, a disponibilização de comportamentos emergentes está fora do escopo da abordagem. Entretanto, em trabalhos futuros, espera-se expandir a abordagem para que seja possível a busca de tais comportamentos. Para isso, o modelo atual deve ser adaptado para compor esses comportamentos, algo semelhante ao módulo *Services Managmt*;
- Implementar um sistema de recomendação para facilitar a busca por serviços similares, com base na sua classificação, para casos em que um CS se torne indisponível; e
- Incentivar *CSOwners* e *SoSDevelopers*: Conforme mencionado, a falta de CS's e SoS's cadastrados são uma das principais limitações. Portanto, é necessário incentivar os engenheiros a participarem e alimentarem o repositório, visto que só possível avaliar se o repositório melhorou o reuso, quando os *assets* estiverem disponíveis para os usuários usarem. Além disso, o *feedback* desses participantes são importantes para adequar a abordagem, caso seja necessário.

REFERÊNCIAS

ABDALLA, G. **Establishment of an ontology for Systems-of-Systems**. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, 2017. Citado nas páginas 47, 52 e 54.

ALEXANDER, R.; KELLY, T. Supporting systems of systems hazard analysis using multi-agent simulation. **Safety Science**, v. 51, n. 1, p. 302 – 318, 2013. Citado na página 102.

ALI, N. B.; PETERSEN, K.; MÄNTYLÄ, M. V. Testing highly complex system of systems: An industrial case study. In: **Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement**. Lund, Sweden: IEEE, 2012. p. 211–220. Citado nas páginas 34 e 102.

AUSTIN, D.; BARBIR, A.; FERRIS, C.; GARG, S. Web services architecture requirements. **W3C Working Group Notes**, p. 22, 2004. Citado na página 37.

AUTRAN, F.; AUZELLE, J.-P.; CATTAN, D.; GARNIER, J.-L.; LUZEAUX, D.; MAYER, F.; PEYRICHON, M.; RUAULT, J.-R. Coupling component systems towards systems of systems. **Annual International Symposium of INCOSE**, 2008. Citado na página 24.

BATISTA, T. Challenges for sos architecture description. In: **Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems**. Montpellier, France: ACM, 2013. (SESoS '13), p. 35–37. Citado nas páginas 33, 34 e 102.

BLOMSTEDT, F.; FERREIRA, L. L.; KLISICS, M.; CHRYSOULAS, C.; SORIA, I. M. de; MORIN, B.; ZABASTA, A.; ELIASSON, J.; JOHANSSON, M.; VARGA, P. The arrowhead approach for soa application development and documentation. In: **IECON - 40th Annual Conference of the IEEE Industrial Electronics Society**. Dallas, TX, USA: IEEE, 2014. p. 2631–2637. Citado na página 35.

BOARDMAN, J.; SAUSER, B. System of systems - the meaning of of. In: **IEEE/SMC International Conference on System of Systems Engineering**. Los Angeles, CA, USA: IEEE, 2006. p. 6. Citado nas páginas 27 e 28.

BRAGA, R. T. V. MBA4SoSi: Uma arquitetura baseada em modelos para integração de sistemas em sistema-de-sistemas. 2016. Projeto de pesquisa FAPESP n. 2016/05129-0. Citado nas páginas 42 e 43.

BRAGA, R. T. V.; FELONI, D.; PACINI, K.; FILHO, D. S.; GOTTARDI, T. Aires: An architecture to improve software reuse. In: _____. **Software Reuse: Bridging with Social-Awareness: 15th International Conference, ICSR 2016, Limassol, Cyprus, June 5-7, 2016, Proceedings**. Cham: Springer International Publishing, 2016. p. 231–246. Citado na página 47.

BRAGA, R. T. V.; VARGAS, I. G.; GOTTARDI, T. A service-based architecture for virtual and collaborative system of systems. In: **X Workshop em Desenvolvimento Distribuído de Software, Ecossistemas de Software e Sistemas-de-Sistemas (WDES), CBSoft Proceedings**. Maringá-PR, Brazil: [s.n.], 2016. p. 51–60. Citado na página 54.

- CAVALCANTE, E.; CACHO, N.; LOPES, F.; BATISTA, T.; OQUENDO, F. Thinking smart cities as systems-of-systems: A perspective study. In: **Proceedings of the 2Nd International Workshop on Smart**. Trento, Italy: ACM, 2016. (SmartCities '16), p. 9:1–9:4. Citado na página 51.
- CHEN, P.; CLOTHIER, J. Advancing systems engineering for systems-of-systems challenges. **Systems Engineering**, Wiley Subscription Services, Inc., A Wiley Company, v. 6, n. 3, p. 170–183, 2003. Citado na página 102.
- CHIPRIANOV, V.; FALKNER, K.; GALLON, L.; MUNIER, M. Towards modelling and analysing non-functional properties of systems of systems. In: **9th International Conference on System of Systems Engineering (SOSE)**. Adelaide, SA, Australia: IEEE, 2014. p. 289–294. Citado na página 101.
- CHIPRIANOV, V.; FALKNER, K.; SZABO, C.; PUDDY, G. Architectural support for model-driven performance prediction of distributed real-time embedded systems of systems. In: _____. **Software Architecture: 8th European Conference, ECSA 2014, Vienna, Austria, August 25-29, 2014. Proceedings**. Cham: Springer International Publishing, 2014. p. 357–364. Citado na página 101.
- CHIPRIANOV, V.; GALLON, L.; SALAMEH, K.; MUNIER, M.; HACHEM, J. E. Towards security software engineering the smart grid as a system of systems. In: **2015 10th System of Systems Engineering Conference (SoSE)**. San Antonio, TX, USA: IEEE, 2015. p. 77–82. Citado na página 101.
- CHOWN, B. The design team as a “System of Systems”. In: **IEEE International Systems Engineering Symposium (ISSE)**. Vienna, Austria: IEEE, 2017. p. 1–5. Citado nas páginas 29 e 30.
- COLE, R. The changing role of requirements and architecture in systems engineering. In: **IEEE/SMC International Conference on System of Systems Engineering**. Los Angeles, CA, USA: IEEE, 2006. p. 5 pp.–. Citado na página 102.
- COLOMBO, A. W.; BANGEMANN, T.; KARNOUSKOS, S. A system of systems view on collaborative industrial automation. In: **IEEE International Conference on Industrial Technology (ICIT)**. Cape Town, South Africa: IEEE, 2013. p. 1968–1975. Citado na página 18.
- DAHMAN, J.; LANE, J. A.; REBOVICH, G.; LOWRY, R. Systems of systems test and evaluation challenges. In: **5th International Conference on System of Systems Engineering**. Loughborough, United Kingdom, United Kingdom: IEEE, 2010. p. 1–6. Citado na página 102.
- DAHMAN, J. S.; BALDWIN, K. J. Understanding the current state of us defense systems of systems and the implications for systems engineering. In: **2nd Annual IEEE Systems Conference**. Montreal, Quebec, Canada: IEEE, 2008. p. 1–7. Citado nas páginas 26 e 102.
- DESPOTOU, G.; ALEXANDER, R.; KELLY, T. Addressing challenges of hazard analysis in systems of systems. In: **3rd Annual IEEE Systems Conference**. Vancouver, BC, Canada: IEEE, 2009. p. 167–172. Citado na página 102.
- DOD. **Systems Engineering Guide for Systems of Systems**. 2008. Disponível em: <<http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf>>. Citado nas páginas 25, 29 e 54.

ENDREI, M.; ANG, J.; ARSANJANI, A.; CHUA, S.; COMTE, P.; KROGDAHL, P.; LUO, M.; NEWLING, T. **Patterns: service-oriented architecture and web services**. [S.l.]: IBM Corporation, International Technical Support Organization, 2004. Citado na página 37.

ERL, T. **Service-oriented architecture: a field guide to integrating XML and web services**. [S.l.]: Prentice hall, 2004. Citado nas páginas 37 e 39.

_____. **Service-oriented architecture (SOA): concepts, technology, and design**. [S.l.]: Prentice Hall, 2005. Citado nas páginas 35, 36 e 38.

_____. **SOA design patterns**. [S.l.]: Pearson Education, 2008. Citado na página 35.

FANG, Z.; DELAURENTIS, D. Multi-stakeholder dynamic planning of system of systems development and evolution. **Procedia Computer Science**, v. 44, p. 95 – 104, 2015. 2015 Conference on Systems Engineering Research. Citado na página 101.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. 180 p. Tese (PHD em Information and Computer Science) — UNIVERSITY OF CALIFORNIA IRVINE, Irvine, California, 2000. Disponível em: <<https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>>. Acesso em: 15/12/2017. Citado na página 70.

FIRESMITH, D. **Profiling systems using the defining characteristics of systems of systems (SoS)**. Pennsylvania, US, 2010. Relatório Técnico. Citado nas páginas 25, 26 e 28.

FIRESMITH, D. G.; CAPELL, P.; FALKENTHAL, D.; HAMMONS, C. B.; IV, D. T. L.; MERENDINO, T. **The method framework for engineering system architectures**. [S.l.]: CRC Press, 2008. Citado na página 29.

FRIEDMAN, T. L. **The world is flat: A brief history of the twenty-first century**. [S.l.]: Macmillan, 2005. Citado na página 17.

GONZALEZ, A.; PIEL, E.; GROSS, H. G.; GLANDRUP, M. Testing challenges of maritime safety and security systems-of-systems. In: **Testing: Academic Industrial Conference - Practice and Research Techniques (taic part 2008)**. Lodge, Windsor, UK: IEEE, 2008. p. 35–39. Citado na página 102.

GRIENDLING, K.; SALMON, J.; MAVRIS, D. Elements of a decision-making framework for early-phase system of systems acquisition. In: **IEEE International Systems Conference SysCon 2012**. Vancouver, BC, Canada: IEEE, 2012. p. 1–8. Citado na página 102.

GROTZINGER, J. P.; VASAVADA, A. R.; RUSSELL, C. T. **Mars Science Laboratory**. [S.l.]: Springer, 2013. Citado na página 18.

GUTIERREZ-GARCIA, J. O.; RAMOS-CORCHADO, F. F.; KONING, J. L. Obligations as constrainers, descriptors, and linkers of open system of systems. In: **IEEE International Conference on System of Systems Engineering (SoSE)**. Albuquerque, NM, USA: IEEE, 2009. p. 1–6. Citado na página 18.

HACHEM, J. E. Towards model driven architecture and analysis of system of systems access control. In: **Proceedings of the 37th International Conference on Software Engineering - Volume 2**. Florence, Italy: IEEE Press, 2015. (ICSE '15), p. 867–870. Citado na página 101.

HALEY, C. B.; NUSEIBEH, B. Bridging requirements and architecture for systems of systems. In: **International Symposium on Information Technology**. Kuala Lumpur, Malaysia: IEEE, 2008. v. 4, p. 1–8. Citado na página 18.

HALLERSTEDE, S.; HANSEN, F. O.; HOLT, J.; LAURITSEN, R.; LORENZEN, L.; PELESKA, J. Technical challenges of sos requirements engineering. In: **7th International Conference on System of Systems Engineering (SoSE)**. Genova, Italy: IEEE, 2012. p. 573–578. Citado na página 102.

HENSHAW, M.; SIEMIENIUCH, C.; SINCLAIR, M.; HENSON, S.; BAROT, V.; JAMSHIDI, M.; DELAURENTIS, D.; NCUBE, C.; LIM, S. L.; DOGAN, H. Systems of systems engineering: A research imperative. In: **International Conference on System Science and Engineering (ICSSE)**. Budapest, Hungary: IEEE, 2013. p. 389–394. Citado nas páginas 33 e 102.

JAMSHIDI, M. **Systems of systems engineering: principles and applications**. [S.l.]: CRC press, 2008. Citado na página 25.

_____. From large-scale systems to system of systems – control challenges for the 21st century1. **IFAC Proceedings Volumes**, v. 43, n. 8, p. 13 – 19, 2010. 12th IFAC Symposium on Large Scale Systems: Theory and Applications. Citado na página 102.

JOSUTTIS, N. M. **SOA in practice: the art of distributed system design**. [S.l.]: "O'Reilly Media, Inc.", 2007. Citado na página 36.

KALAWSKY, R. S.; JOANNOU, D.; TIAN, Y.; FAYOUMI, A. Using architecture patterns to architect and analyze systems of systems. **Procedia Computer Science**, v. 16, p. 283 – 292, 2013. 2013 Conference on Systems Engineering Research. Citado nas páginas 34 e 102.

KAMPERT, D.; NAZARI, S.; SONNTAG, C.; EPPLE, U.; ENGELL, S. A framework for simulation, optimization and information management of physically-coupled systems of systems. **IFAC-PapersOnLine**, v. 48, n. 3, p. 1553 – 1558, 2015. 15th IFAC Symposium on Information Control Problems in Manufacturing. Citado na página 101.

KAUR, N.; MCLEOD, C. S.; JAIN, A.; HARRISON, R.; AHMAD, B.; COLOMBO, A. W.; DELSING, J. Design and simulation of a soa-based system of systems for automation in the residential sector. In: **IEEE International Conference on Industrial Technology (ICIT)**. Cape Town, South Africa: IEEE, 2013. p. 1976–1981. Citado na página 18.

KAZMAN, R.; SCHMID, K.; NIELSEN, C. B.; KLEIN, J. Understanding patterns for system of systems integration. In: **8th International Conference on System of Systems Engineering**. Maui, HI, USA: IEEE, 2013. p. 141–146. Citado na página 18.

KEATING, C. B. Governance implications for meeting challenges in the system of systems engineering field. In: **9th International Conference on System of Systems Engineering (SOSE)**. Adelaide, SA, Australia: IEEE, 2014. p. 154–159. Citado na página 101.

KEWLEY, R.; COOK, J.; GOERGER, N.; HENDERSON, D.; TEAGUE, E. Federated simulations for systems of systems integration. In: **Winter Simulation Conference**. Miami, FL, USA, USA: IEEE, 2008. p. 1121–1129. Citado na página 18.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007. Citado nas páginas 30 e 103.

KRUEGER, I. H.; MEISINGER, M.; MENARINI, M.; PASCO, S. Rapid systems of systems integration - combining an architecture-centric approach with enterprise service bus infrastructure. In: **IEEE International Conference on Information Reuse Integration**. Waikoloa Village, HI, USA: IEEE, 2006. p. 51–56. Citado na página 18.

KRYGIEL, A. J. **Behind the Wizard's Curtain. An Integration Environment for a System of Systems**. [S.l.], 1999. Citado na página 24.

KUMAR, B. **Web Services**. [S.l.]: Tata McGraw-Hill Education, 2004. Citado na página 38.

LANE, J. A. **What is a system of systems and why should i care?** Relatório técnico, University of Southern California, Los Angeles, CA, USA, 2013. Citado nas páginas 17, 26 e 54.

LARSSON, J.; BORG, M.; OLSSON, T. Testing quality requirements of a system-of-systems in the public sector - challenges and potential remedies. **CoRR**, abs/1602.05618, 2016. Disponível em: <http://arxiv.org/abs/1602.05618>. Citado na página 101.

LEWIS, G.; MORRIS, E.; SIMANTA, S.; SMITH, D. Service orientation and systems of systems. **IEEE Software**, v. 28, n. 1, p. 58–63, Jan 2011. Citado nas páginas 18, 43, 44, 48, 52 e 54.

LEWIS, G. A.; MORRIS, E.; PLACE, P.; SIMANTA, S.; SMITH, D. B. Requirements engineering for systems of systems. In: **3rd Annual IEEE Systems Conference**. Vancouver, BC, Canada: IEEE, 2009. p. 247–252. Citado na página 102.

LINTHICUM, D. S. **Next generation application integration: from simple information to Web services**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2003. Citado na página 19.

LIU, S. Employing system of systems engineering in china's emergency management. **IEEE Systems Journal**, v. 5, n. 2, p. 298–308, June 2011. Citado nas páginas 28 e 102.

LOWE, P. N.; CHEN, M. W. System of systems complexity: modeling and simulation issues. In: **Proceedings of the 2008 Summer Computer Simulation Conference**. Edinburgh, Scotland: Society for Modeling & Simulation International, 2008. p. 36. Citado nas páginas 27 e 29.

LUKASIK, S. J. Systems, systems of systems, and the education of engineers. **Artificial Intelligence for Engineering Design, Analysis and Manufacturing**, Cambridge University Press, v. 12, n. 1, p. 55–60, 1998. Citado na página 24.

MA, K. J. Web services: what's real and what's not? **IT professional**, IEEE, v. 7, n. 2, p. 14–21, 2005. Citado na página 35.

MADNI, A. Integrating humans with and within software and systems: Challenges and opportunities. (**Invited Paper**) **CrossTalk, The Journal of Defense Software Engineering**, 2011. Citado na página 19.

MADNI, A. M. Integrating humans with software and systems: Technical challenges and a research agenda. **Systems Engineering**, Wiley Online Library, v. 13, n. 3, p. 232–245, 2010. Citado na página 19.

MADNI, A. M.; SIEVERS, M. System of systems integration: Key considerations and challenges. **Systems Engineering**, Wiley Online Library, v. 17, n. 3, p. 330–347, 2014. Citado nas páginas 18, 19, 27, 33, 98 e 101.

- MAIA, P.; CAVALCANTE, E.; GOMES, P.; BATISTA, T.; DELICATO, F. C.; PIRES, P. F. On the development of systems-of-systems based on the internet of things: A systematic mapping. In: **Proceedings of the European Conference on Software Architecture Workshops**. Vienna, Austria: ACM, 2014. (ECSAW '14), p. 23–8. Citado na página 101.
- MAIER, M. W. Architecting principles for systems-of-systems. **Systems Engineering**, John Wiley & Sons, Inc., v. 1, n. 4, p. 267–284, 1998. ISSN 1520-6858. Citado nas páginas 24, 25, 26, 29 e 100.
- MOURA, D. de S. **Software Profile RAS : estendendo a padronização do Reusable Asset Specification e construindo um repositório de ativos**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação., 2013. Disponível em: <<http://hdl.handle.net/10183/87582>>. Citado na página 47.
- NAKAGAWA, E. Y.; OQUENDO, F.; AVGERIOU, P.; CUESTA, C. E.; DRIRA, K.; MALDONADO, J. C.; ZISMAN, A. Foreword: towards reference architectures for systems-of-systems. In: **Proceedings of the Third International Workshop on Software Engineering for Systems-of-Systems**. Florence, Italy: IEEE Press, 2015. (SESos '15), p. 1–4. Citado nas páginas 17 e 101.
- NASCIMENTO, D. L.; BRAGA, R. T. V. Challenges in system of systems development: A systematic mapping. In: **XXI Ibero-American Conference on Software Engineering (Cibse-ESELAW)**. Bogotá, Colombia: [s.n.], 2018. Citado na página 20.
- NASCIMENTO, D. L.; FELONI, D. F. G. F.; BRAGA, R. T. V. **A Service-Oriented Approach for a System of Systems Repository**. 2017. Artigo submetido a conferência internacional, em análise. Citado na página 66.
- NATIONS, U. **Urban and Rural Areas 2009**. 2009. Disponível em: <<http://www.un.org/en/development/desa/population/publications/urbanization/urban-rural.shtml>>. Acesso em: 07/12/2017. Citado na página 19.
- NCUBE, C. On the engineering of systems of systems: Key challenges for the requirements engineering community. In: **Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems**. Trento, Italy: IEEE, 2011. p. 70–73. Citado na página 102.
- NEWCOMER, E.; LOMOW, G. **Understanding SOA with Web services**. [S.l.]: Addison-Wesley, 2005. Citado na página 35.
- NIELSEN, C. B.; LARSEN, P. G.; FITZGERALD, J.; WOODCOCK, J.; PELESKA, J. Systems of systems engineering: Basic concepts, model-based techniques, and research directions. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 48, n. 2, p. 18–41, set. 2015. Citado nas páginas 17 e 101.
- OMG, O. M. G. **RAS - Reusable Asset Specification**. 2005. Disponível em: <<http://www.omg.org/spec/RAS/2.2/>>. Citado nas páginas 39, 40, 41, 42, 52 e 54.
- PAPATHEOCHAROUS, E.; AXELSSON, J.; ANDRERSSON, J. Issues and challenges in ecosystems for federated embedded systems. In: **Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems**. Montpellier, France: ACM, 2013. (SESos '13), p. 21–24. Citado na página 101.

PAPAZOGLU, M. P.; HEUVEL, W.-J. Service oriented architectures: approaches, technologies and research issues. **The VLDB Journal—The International Journal on Very Large Data Bases**, Springer-Verlag New York, Inc., v. 16, n. 3, p. 389–415, 2007. Citado na página 35.

PEI, R. S. System of systems integration (sosi)-a "smart" way of acquiring army c4i2ws systems. In: SOCIETY FOR COMPUTER SIMULATION INTERNATIONAL; 1998. **Summer Computer Simulation Conference**. Vancouver, Canada, 2000. p. 574–579. Citado na página 24.

PENNOCK, M. J.; WADE, J. P. The top 10 illusions of systems engineering: A research agenda. **Procedia Computer Science**, v. 44, p. 147 – 154, 2015. 2015 Conference on Systems Engineering Research. Citado na página 101.

PENZENSTADLER, B.; ECKHARDT, J. A requirements engineering content model for cyber-physical systems. In: **2012 Second IEEE International Workshop on Requirements Engineering for Systems, Services, and Systems-of-Systems (RESS)**. Chicago, IL, USA: IEEE, 2012. p. 20–29. Citado na página 102.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: **12th international conference on evaluation and assessment in software engineering**. Italy: BCS Learning & Development Ltd., 2008. v. 17, n. 1. Citado nas páginas 30 e 97.

RABISER, R.; VIERHAUSER, M.; GRÜNBACHER, P. Variability management for a runtime monitoring infrastructure. In: **Proceedings of the Ninth International Workshop on Variability Modelling of Software-intensive Systems**. Hildesheim, Germany: ACM, 2015. (VaMoS '15), p. 35–42. Citado na página 101.

REN, H. m.; YAN, Z. y.; ZHANG, J. z. Design and implementation of ras-based open source software repository. In: **2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery**. Tianjin, China: IEEE, 2009. v. 2, p. 219–223. Citado na página 47.

RING, J.; MADNI, A. M. Key challenges and opportunities in 'system of systems' engineering. In: **2005 IEEE International Conference on Systems, Man and Cybernetics**. Waikoloa, HI, USA: IEEE, 2005. v. 1, p. 973–978. Citado na página 102.

RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. **Unified Modeling Language Reference Manual, The (2nd Edition)**. [S.l.]: Pearson Higher Education, 2004. ISBN 0321245628. Citado na página 55.

SAFWAT, A.; SENOUSY, M. Addressing challenges of ultra large scale system on requirements engineering. **Procedia Computer Science**, v. 65, p. 442 – 449, 2015. International Conference on Communications, management, and Information technology (ICCMIT'2015). Citado na página 101.

SAGE, A. P.; CUPPAN, C. D. On the systems engineering and management of systems of systems and federations of systems. **Inf. Knowl. Syst. Manag.**, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 2, n. 4, p. 325–345, dez. 2001. Citado na página 24.

SECURITY, U. D. H. **The System of Systems Approach for Interoperable Communications**. 2004. Disponível em: <http://www.npstc.org/download.jsp?tableId=37&column=217&id=2458&file=SOSApproachforInteroperableCommunications_02.pdf>. Citado na página 28.

SMITH, D.; LEWIS, G. Systems of systems: New challenges for maintenance and evolution. In: **Frontiers of Software Maintenance**. Beijing, China: IEEE, 2008. p. 149–157. Citado na página 102.

SOMMERVILLE, I. **Software Engineering**. [S.l.]: Pearson, 2011. (International Computer Science Series). ISBN 9780137053469. Citado nas páginas 18, 20 e 39.

SOUZA, V. **Uma arquitetura orientada a serviços para desenvolvimento, gerenciamento e instalação de serviços de rede. 2006. 93 f.** Dissertação (Mestrado) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2006. Citado na página 35.

STUTZKE, R. D. Factors affecting effort to integrate and test a system of systems. 2005. Citado na página 24.

VARGA, P.; BLOMSTEDT, F.; FERREIRA, L. L.; ELIASSON, J.; JOHANSSON, M.; DELSING, J.; SORIA, I. Martinez de. Making system of systems interoperable the core components of the arrowhead framework. **J. Netw. Comput. Appl.**, Academic Press Ltd., London, UK, UK, v. 81, n. C, p. 85–95, 2017. Citado nas páginas 18, 19 e 46.

VARGAS, I. G. **SoS-Reúso: um SoS do tipo direcionado para facilitar o reúso de software**. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, 2017. Citado nas páginas 45, 46, 48, 52 e 54.

VARGAS, I. G.; GOTTARDI, T.; BRAGA, R. T. V. Approaches for integration in system of systems: a systematic review. In: **Proceedings of the 4th International Workshop on Software Engineering for Systems-of-Systems**. Austin, Texas: ACM, 2016. p. 32–38. Citado nas páginas 42, 98 e 101.

VIERHAUSER, M.; RABISER, R.; GRÜNBACHER, P.; DANNER, C.; WALLNER, S. Evolving systems of systems: Industrial challenges and research perspectives. In: **Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems**. Montpelier, France: ACM, 2013. (SESoS '13), p. 1–4. Citado na página 102.

VIERHAUSER, M.; RABISER, R.; GRÜNBACHER, P. Monitoring requirements in systems of systems. **IEEE Software**, v. 33, n. 5, p. 22–24, Sept 2016. Citado na página 17.

VIERHAUSER, M.; RABISER, R.; GRÜNBACHER, P.; DANNER, C.; WALLNER, S.; ZEISEL, H. A flexible framework for runtime monitoring of system-of-systems architectures. In: **IEEE/IFIP Conference on Software Architecture**. Sydney, NSW, Australia: IEEE, 2014. p. 57–66. Citado na página 101.

XIAO, B.; WANG, Y.; MA, L.; CAO, Z. Research on the history and perspective of system of systems. In: **The Proceedings of 9th International Conference on Reliability, Maintainability and Safety**. Guiyang, China: [s.n.], 2011. p. 1262–1266. Citado na página 27.

ZAMBONI, A.; THOMMAZO, A.; HERNANDES, E.; FABBRI, S. Start uma ferramenta computacional de apoio à revisão sistemática. In: **Proc.: Congresso Brasileiro de Software (CBSOFT'10), Salvador, Brazil**. Salvador, Brazil: [s.n.], 2010. Citado na página 97.

MAPEAMENTO SISTEMÁTICO

Nesta Seção, o Mapeamento Sistemático (MS) é apresentado em detalhes, seguindo um protocolo estratégico para auxiliar no processo do mapeamento (PETERSEN *et al.*, 2008). Para uma melhor apresentação, o processo foi dividido nas seguintes etapas:

- **Planejamento:** Nessa fase é investigada a necessidade de realizar o mapeamento sistemático, definir suas etapas e um protocolo do estudo;
- **Condução:** Nessa fase é conduzido o mapeamento, em que são identificadas as principais bases de dados e são realizadas as buscas por estudos nessas bases, ressaltando que os estudos relacionados ao escopo são filtrados para a próxima etapa.
- **Sumarização:** Nessa fase é feita uma coleta de dados dos estudos selecionados.

A.1 Planejamento

Para realizar o MS, foi utilizada a ferramenta de revisão sistemática *State of the Art through Systematic Review* (StArt) (ZAMBONI *et al.*, 2010), desenvolvida pelo Laboratório de Pesquisa em Engenharia de Software da Universidade Federal de São Carlos. O escopo desse estudo foi planejado conforme os critérios descritos a seguir:

- **População:** Será observado o desenvolvimento no contexto de SoS, por meio de estudos primários e secundários encontrados nas bases de dados utilizadas, mais especificamente teses, dissertações, livros e artigos;
- **Intervenção:** Serão investigados no MS os desafios/problemas com o uso de SoS, que correspondem também no uso ferramentas, técnicas, *frameworks*, etc;

- **Controle:** Estudos iniciais que são do conhecimento do autor e que abrangem o escopo do MS, serão utilizados como forma de evidenciar que o MS está pertinentes aos objetivos. Para esse MS foi utilizado como controle: 1) (MADNI; SIEVERS, 2014) e 2) (VARGAS; GOTTARDI; BRAGA, 2016);
- **Resultados:** São os resultados obtidos após a análise dos estudos. Nesse MS espera-se como o resultado a identificação dos principais problemas enfrentados durante o desenvolvimento de SoS, ferramentas e sugestões a serem consideradas; e
- **Aplicação:** Corresponde a quem o estudo se destina, seja profissionais, pesquisadores ou indústrias. Nesse contexto, esse MS destina-se a engenheiros de software, pesquisadores, desenvolvedores e acadêmicos que utilizam SoS.

Seguindo o protocolo, esse MS foi conduzido com o objetivo de responder duas RQ (*Research Question*) direcionadas ao ambiente de desenvolvimento de SoS, cujo intuito é obter e categorizar os problemas que ocorrem no contexto de SoS. As RQ são:

1. **RQ1 - Quais são os desafios enfrentados durante o desenvolvimento de SoS?** -Apontar dificuldades durante o desenvolvimento de SoS, indiferentemente do domínio ou ferramenta utilizada-;
2. **RQ2: Foi evidenciada alguma solução para os desafios encontrados?** -Verificar se foi apontada alguma solução/consideração para os problemas encontrados durante a análise dos estudos-.

Os critérios utilizados para definir se um estudo é pertinente ao escopo do MS são expostos a seguir:.

Critérios de Inclusão:

- Estudos que destacam explicitamente problemas/desafios durante o uso de SoS.

Critérios de Exclusão:

- O estudo não está em português ou inglês;
- O estudo não está completo ou o pesquisador não tem acesso;
- O estudo não é relacionado a SoS ou não aponta os problemas/desafios.

A.2 Condução

Foram escolhidas as bases mais relevantes para computação e engenharia, visto que, dentre as diversas bases, essas indexam as principais conferências do tema de pesquisa, visando também a disponibilidade dos estudos para o pesquisador. Na Tabela 12 é apresentada a lista escolhida.

Tabela 12 – Base de Dados Seleccionadas

Base	Endereço
IEEEExplore	http://ieeexplore.ieee.org
ACM DigitalLibrary	http://dl.acm.org
Scopus	http://www.scopus.com
Web of Science	http://sub3.webofknowledge.com
Science Direct	http://www.sciencedirect.com

Fonte: Elaborada pelo autor.

Foi utilizado um conjunto de palavras-chaves e seus sinônimos para um retorno mais preciso dos estudos, de tal maneira que os resultados sejam mais pertinentes ao escopo do problema ou área a ser pesquisada. Frente a isso, foram utilizadas as palavras apresentadas na Tabela 13.

Tabela 13 – Palavras-chaves e seus Sinônimos

Termo	Sinônimo
System of Systems	Systems of Systems; System-of-System; Systems-of-Systems
Software Development	Software Engineering; Development; Engineering
Problem	Difficulty; Challenge

Fonte: Elaborada pelo autor.

Com o auxílio desse conjunto, foi possível formular uma *string* de busca e utilizar operadores booleanos (AND e OR) para calibrar nas cinco bases de busca. A *string* genérica utilizada como base é apresentada a seguir.

("System-of-Systems"OR "System-of-System"OR "Systems-of-Systems") AND ("Development"OR "Software Development"OR "Software Engineering") AND ("Problem"OR "Difficulty"OR "Challenge")

O procedimento para leitura dos 900 estudos foi iniciado logo após a última calibragem da *string*, em agosto de 2016. Esse processo foi dividido em duas etapas, primeiramente a ideia era apenas a leitura do título e *abstract* e classificá-los de acordo com sua relevância. Caso positivo, o estudo primário passaria para a próxima etapa, destacando também que o estudo deve ser pertinente ao critério de inclusão para ser aceito. Caso contrário, o estudo é excluído por algum critério de exclusão. Entretanto, foi feita uma mudança após a leitura de alguns estudos, pois era difícil identificar os desafios/problemas apenas com a leitura do título e *abstract*.

Consequentemente, foi incluído também a leitura da conclusão com o intuito de mitigar esse problema.

A segunda etapa, denominada extração, foi executada com os mesmos critérios de inclusão e exclusão, entretanto, foi feita uma leitura transversal dos estudos, visto que, essa técnica de leitura tem um foco maior nas partes mais relevantes do texto, mais especificamente introdução e conclusão.

É comum obter uma quantidade substancial de estudos repetidos, porque algumas bases de busca indexam a mesma conferência ou periódico. No caso desse MS, foi encontrado um total de 194 estudos duplicados e, dos 706 restantes, 580 foram rejeitados por algum critério de exclusão. Portanto, a maioria dos estudos não passaram pelo primeiro critério de inclusão porque não deixavam claro qual era o desafio/problema enfrentado durante o desenvolvimento de SoS. Diante disso, 126 estudos foram aprovados pela etapa inicial, entretanto, restaram apenas 39 após a leitura da segunda etapa.

A.2.1 Sumarização dos Dados

Nesse MS não foi aceito nenhum estudo para sumarização dos resultados com data anterior à 2003, mesmo sem a utilização de filtragem de ano para auxiliar a verificação de algum acontecimento específico ou de como está a área nos últimos anos. O porquê de não terem sido encontrados estudos anteriores à 2003, é justificado pelo fato que, apesar de sistemas com essas características serem comentados em uma data um pouco mais distante, suas definições e particularidades foram definidas inicialmente por [Maier \(1998\)](#). Consequentemente, é comum os desafios/problemas surgirem ou serem discutidos a partir de 2003, como é evidenciado na Figura 43. A Tabela 16 mostra a lista completa com os respectivos estudos em ordem cronológica inversa. Vale ressaltar que a busca foi feita em meados de 2016, portanto a queda do número de artigos em 2016 deve-se ao fato desse ano não ter sido considerado em sua totalidade, além do atraso natural para que artigos recém publicados sejam indexados pelas máquinas de busca.

A correspondência dos estudos com a classificação dos domínios de aplicação e com os tipos são apresentados na Tabela 14 e Tabela 15.

Tabela 14 – Classificação dos Domínios de Aplicação (Com ID dos estudos)

ID	Quantidade	Domínio
S1, S3, S7, S9, S10, S11, S12, S14, S15, S17, S20, S21, S22, S23, S26, S27, S28, S31, S32, S33, S34, S37, S38, S39	24	Sem domínio específico
S5, S18, S19, S24	4	Automação industrial
S13, S16	2	Sistema de observação da terra
S30, S36	2	Departamento de defesa
S2, S29	2	Setor público
S4, S35	2	Marítimo
S6, S8	2	Fisicamente acoplado
S25	1	Automotivo, Estrada de ferro e Aviônica

Fonte: Elaborada pelo autor.

Figura 43 – Distribuição dos Estudos

Fonte: Elaborada pelo autor.

Tabela 15 – Classificação dos Tipos de Estudos (Com ID dos estudos)

ID	Quantidade	Tipo
S1, S12	2	RS e MS
S6, S8, S10, S14, S16, S23, S27, S30, S35, S36	10	Framework
S3, S21, S22, S37	4	Arquitetura
S7, S9, S11, S15, S17, S18, S20, S28, S32, S33, S34, S38, S39	13	Abordagem
S5, S24, S26, S29, S31	5	Estudo de caso
S13	1	Ontologia

Fonte: Elaborada pelo autor.

ID	Estudo	Ano
S1	Approaches for Integration in System of Systems: A Systematic Review (VARGAS; GOTTARDI; BRAGA, 2016)	2016
S2	Testing quality requirements of a system-of-systems in the public sector - Challenges and potential remedies (LARS-SON; BORG; OLSSON, 2016)	2016
S3	Foreword: Towards Reference Architectures for Systems-of-systems (NAKAGAWA <i>et al.</i> , 2015)	2015
S4	Towards Model Driven Architecture and Analysis of System of Systems Access Control (HACHEM, 2015)	2016
S5	Variability Management for a Runtime Monitoring Infrastructure (RABISER; VIERHAUSER; GRÜNBACHER, 2015)	2015
S6	Towards security Software engineering the Smart Grid as a System of Systems (CHIPRIANOV <i>et al.</i> , 2015)	2015
S7	Addressing Challenges of Ultra Large Scale System on Requirements Engineering (SAFWAT; SENOUSY, 2015)	2015
S8	A Framework for Simulation, Optimization and Information Management of Physically-Coupled Systems of Systems (KAMPERT <i>et al.</i> , 2015)	2015
S9	The Top 10 Illusions of Systems Engineering: A Research Agenda (PENNOCK; WADE, 2015)	2015
S10	Multi-stakeholder Dynamic Planning of System of Systems Development and Evolution (FANG; DELAURENTIS, 2015)	2015
S11	Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions (NIELSEN <i>et al.</i> , 2015)	2015
S12	On the Development of Systems-of-Systems Based on the Internet of Things: A Systematic Mapping (MAIA <i>et al.</i> , 2014)	2014
S13	System of systems integration: Key considerations and challenges (MADNI; SIEVERS, 2014)	2014
S14	Towards modelling and analysing non-functional properties of systems of systems (CHIPRIANOV <i>et al.</i> , 2014a)	2014
S15	Architectural support for model-driven performance prediction of distributed real-time embedded systems of systems (CHIPRIANOV <i>et al.</i> , 2014b)	2014
S16	A flexible framework for runtime monitoring of system-of-systems architectures (VIERHAUSER <i>et al.</i> , 2014)	2014
S17	Governance Implications for Meeting Challenges in the System of Systems Engineering Field (KEATING, 2014)	2014
S18	Issues and Challenges in Ecosystems for Federated Embedded Systems (PAPATHEOCHAROUS; AXELSSON; ANDRERSSON, 2013)	2013

Tabela 16 – Estudos Aceitos.

S19	Evolving Systems of Systems: Industrial Challenges and Research Perspectives (VIERHAUSER <i>et al.</i> , 2013)	2013
S20	Challenges for SoS Architecture Description (BATISTA, 2013)	2013
S21	Systems of Systems Engineering: A research Imperative (HENSHAW <i>et al.</i> , 2013)	2013
S22	Using Architecture Patterns to Architect and Analyze Systems of Systems (KALAWSKY <i>et al.</i> , 2013)	2013
S23	Supporting systems of systems hazard analysis using multi-agent simulation (ALEXANDER; KELLY, 2013)	2013
S24	Testing Highly Complex System of Systems: An Industrial Case Study (ALI; PETERSEN; MÄNTYLÄ, 2012)	2012
S25	A Requirements Engineering content model for Cyber-Physical Systems (PENZENSTADLER; ECKHARDT, 2012)	2012
S26	Technical challenges of SoS requirements engineering (HALLERSTEDE <i>et al.</i> , 2012)	2012
S27	Elements of a Decision-Making Framework for Early-Phase System of Systems Acquisition (GRIENDLING; SALMON; MAVRIS, 2012)	2012
S28	On the engineering of systems of systems: Key challenges for the requirements engineering community (NCUBE, 2011)	2011
S29	Employing system of systems engineering in China's emergency management (LIU, 2011)	2011
S30	Systems of systems test and evaluation challenges (DAHMAN <i>et al.</i> , 2010)	2010
S31	From Large-Scale Systems to System of Systems – Control Challenges for the 21st Century (JAMSHIDI, 2010)	2010
S32	Requirements Engineering for Systems of Systems (LEWIS <i>et al.</i> , 2009)	2009
S33	Addressing challenges of hazard analysis in systems of systems (DESPOTOU; ALEXANDER; KELLY, 2009)	2009
S34	Systems of systems: New challenges for maintenance and evolution (SMITH; LEWIS, 2008)	2008
S35	Testing challenges of maritime safety and security systems-of-systems (GONZALEZ <i>et al.</i> , 2008)	2008
S36	Understanding the current state of US defense systems of systems and the implications for systems engineering (DAHMAN; BALDWIN, 2008)	2008
S37	The changing role of requirements and architecture in systems engineering (COLE, 2006)	2006
S38	Key challenges and opportunities in 'system of systems' engineering (RING; MADNI, 2005)	2005
S39	Advancing systems engineering for systems-of-systems challenges (CHEN; CLOTHIER, 2003)	2003

Fonte: Elaborada pelo autor.

Por fim, a Tabela 17, apresenta os estudos que têm desafios em aberto e os que apresentam considerações para as respectivas categorias.

Tabela 17 – Categorização dos Problemas (Completa)

ID	Desafio	Considerações
S1, S5, S8, S9, S11, S12, S13, S17, S18, S21, S28, S32, S35, S38, S39	Integração	S27
S1, S5, S10, S17, S18, S21, S29, S32, S34, S36, S39	Gerenciamento	S7
S1, S3, S12, S13, S20	Documentação	
S1, S3, S11, S20, S21, S22, S32, S34, S37, S39	Evolução e Manutenção	S19, S27
S1, S3, S11, S20, S22, S31, S37, S39	Arquitetura e Modelagem	S4, S6, S14, S15, S16, S23, S27, S33
S20, S26, S28, S32, S37	Requisitos e Qualidade	S2, S7, S25
S24, S35	Teste	S30
S11, S20, S35	Reconfiguração Dinâmica e Monitoramento	S5, S16, S19
S1, S5, S11, S21, S28	Comportamento	

Fonte: Elaborada pelo autor.

A.3 Ameaças à Validade

Esse MS foi conduzido em agosto de 2016 e está limitado a estudos que discutem o desenvolvimento de SoS. É importante enfatizar que nenhum filtro foi usado nas bases de busca

e nenhum tipo de literatura cinza foi incluída. Portanto, as principais ameaças à validade desta pesquisa estão relacionadas ao viés de seleção e extração dos estudos.

Primeiramente, todo o processo foi estruturado a partir das RQ's e o protocolo definido na seguindo os critérios rigorosamente. Esse protocolo foi escrito pelo proponente, mas também foi revisado e refinado pela orientadora desta pesquisa, melhorando a coerência e precisão do MS com as RQ's. A seleção dos estudos foi realizada estrategicamente (seguindo os critérios de inclusão e exclusão), usando a ferramenta de revisão sistemática “State of the Art through Systematic Review” (StArt), que visa auxiliar a execução de um MS e organizar a seleção de estudos e remover automaticamente estudos duplicados.

É importante enfatizar que cada estudo foi submetido a duas fases de classificação rigorosas, na qual os 39 estudos foram aceitos duas vezes pelo critério de inclusão. No entanto, embora as bases de dados sejam as mais relevantes na área ([KITCHENHAM; CHARTERS, 2007](#)), alguns estudos talvez não tenham sido analisados devido às limitações dos mecanismos de pesquisa ou à falta de acesso dos autores ao trabalho.

O viés na extração dos estudos também foi controlado pelo estabelecimento de um protocolo baseado nas RQ's. No entanto, nem todos os estudos tiveram respostas bem definidas as RQ's do MS, por isso eles precisavam ser interpretados e categorizados pelos autores desse trabalho.

QUESTIONÁRIO

Qual o seu nível de conhecimento em Sistema de Sistemas?

- ☐ Muito alto
- ☐ Alto
- ☐ Regular
- ☐ Baixo
- ☐ Muito baixo

Você tem algum experiência com Sistema de Sistemas?

- ☐ Desenvolvimento
- ☐ Pesquisa

1a) - O modelo conceitual está de acordo com o objetivo de criar um repositório para disponibilizar e buscar ativos (Serviços, Constituintes e SoS)?

- ☐ Concordo fortemente
- ☐ Concordo
- ☐ Não tenho certeza
- ☐ Discordo
- ☐ Discordo fortemente

1b) - Caso necessário, utilize este espaço para adicionar comentários sobre sua resposta.

2a) - A RAS (Reusable Asset Specification) é utilizada com o objetivo de fomentar o reúso de ativos (assets), por isso, alguns elementos foram adaptados e utilizados no modelo desta proposta. Na sua opinião, essa adaptação/inspiração na RAS auxilia a promover o reúso de recursos na abordagem?

- ☐ Concordo fortemente
- ☐ Concordo
- ☐ Não tenho certeza
- ☐ Discordo
- ☐ Discordo fortemente
- ☐ Não conheço a RAS suficiente para responder esta pergunta

2b) - Caso necessário, utilize este espaço para adicionar comentários sobre sua resposta.

3a) - Os casos de uso estão de acordo com o objetivo (criar um repositório) da abordagem?

- ☐ Concordo fortemente
- ☐ Concordo
- ☐ Não tenho certeza
- ☐ Discordo

☐ Discordo fortemente

3b) - Caso necessário, utilize este espaço para adicionar comentários sobre sua resposta.

4a) - De acordo com os casos de uso, os papéis de cada ator (User, CSOwner, SoS-Developer) estão bem definidos?

☐ Concordo fortemente

☐ Concordo

☐ Não tenho certeza

☐ Discordo

☐ Discordo fortemente

4b) - Caso necessário, utilize este espaço para adicionar comentários sobre sua resposta.

5a) - Na sua opinião, existe algum ator que não foi apresentado, mas deveria estar presente na abordagem?

☐ Sim

☐ Não

☐ Não tenho certeza

5b) - Caso necessário, utilize este espaço para adicionar comentários sobre sua resposta.

6a) - Cada elemento está bem definido e detalhado no glossário do modelo conceitual?

- ☐ Concordo fortemente
- ☐ Concordo
- ☐ Não tenho certeza
- ☐ Discordo
- ☐ Discordo fortemente

6b) - Caso necessário, utilize este espaço para adicionar comentários sobre sua resposta.

7a) - O caso de uso (Encontrar asset) auxiliou a compreender a abordagem?

- ☐ Concordo fortemente
- ☐ Concordo
- ☐ Não tenho certeza
- ☐ Discordo
- ☐ Discordo fortemente

7b) - Caso necessário, utilize este espaço para adicionar comentários sobre sua resposta.

SERVIÇOS DO MÓDULO USER MANAGMT

Tabela 18 – Serviços do Módulo User Managmt

User Managmt @URI("http://sosrepository-repositorydoc.1d35.starter-us-east-1.openshiftapps.com/ServiceOrientedSoS/rest/user")			
Serviço	HTTPVerb	Consome/Produz	Retorno (Sucesso/Falha)
@Path("create") createUser(User);	POST	XML	Cód.Status = 201 UserManagmt->New user created;
		JSON	
		TEXT_PLAIN XML	Cód.Status = 400 UserManagmt->error : Details
@Path("edit") editUser(User);	PUT	JSON	Cód.Status = 200 UserManagmt->User edited;
		XML	
		TEXT_PLAIN XML	Cód.Status = 404 UserManagmt->error : Details
@Path("delete") deleteUser();	DELETE		Cód.Status = 200 UserManagmt->User deleted;
		TEXT_PLAIN XML	Cód.Status = 404 UserManagmt->error : Details
@Path("search") searchUser();	GET		Cód.Status = 200 return User;
		XML JSON	Cód.Status = 404 UserManagmt->error : Details
@Path("recoverPassword") recoverPassword(Email);	GET		Cód.Status = 200 UserManagmt->E-mail send
		TEXT_PLAIN XML	Cód.Status = 404 UserManagmt->error : Details

Fonte: Elaborada pelo autor.

SERVIÇOS DO MÓDULO CS MANAGMT

Tabela 19 – Serviços do Módulo CS Managmt

CS Managmt @URI("http://sosrepository-repositorydoc.1d35.starter-us-east-1.openshiftapps.com/ ServiceOrientedSoS/rest/constituent")			
Serviço	HTTPVerb	Consome/Produz	Retorno (Sucesso/Falha)
@Path("create")	POST	XML	Cód.Status = 201
		JSON	CSManagmt->New CS created;
		TEXT_PLAIN XML	Cód.Status = 400 CSManagmt->error: Details
@Path("addModel")	POST	XML	Cód.Status = 201
		JSON	CSManagmt->New model added;
		TEXT_PLAIN XML	Cód.Status = 400 CSManagmt->error: Details
@Path("edit")	PUT	XML	Cód.Status = 200
		JSON	CSManagmt-> CS edited;
		TEXT_PLAIN XML	Cód.Status = 404 CSManagmt->error: Details
@Path("editModel")	PUT	XML	Cód.Status = 200
		JSON	CSManagmt->New model edited;
		TEXT_PLAIN XML	Cód.Status = 400 CSManagmt->error: Details
@Path("changeState")	PUT		Cód.Status = 200 CSManagmt->CS state edited;
		TEXT_PLAIN	Cód.Status = 400
		XML	Cód.Status = 404 CSManagmt->error: Details
@Path("delete")	DELETE		Cód.Status = 200 CSManagmt->CS deleted;
		TEXT_PLAIN	Cód.Status = 404
		XML	CSManagmt->error: Details
@Path("deleteModel")	DELETE		Cód.Status = 200 CSManagmt->Model deleted;
		TEXT_PLAIN	Cód.Status = 404
		XML	CSManagmt->error: Details
@Path("search")	GET		Cód.Status = 200 return CS;

		XML JSON	Cód.Status = 404 CSManagmt->error: Details
@Path("searchAll")	GET		Cód.Status = 200 return List<CS>;
		XML JSON	Cód.Status = 404 CSManagmt->error: Details
@Path("searchAll ByStatus")	GET		Cód.Status = 200 return List<CS>;
		XML JSON	Cód.Status = 400 Cód.Status = 404 CSManagmt->error: Details
@Path("searchMy Constituents")	GET		Cód.Status = 200 return List<CS>;
		XML JSON	Cód.Status = 404 CSManagmt->error: Details
@Path("searchMy ConstituentsByStatus")	GET		Cód.Status = 200 return List<CS>;
		XML JSON	Cód.Status = 400 Cód.Status = 400 CSManagmt->error: Details
@Path("searchModels")	GET		Cód.Status = 200 return List<Model>;
		XML JSON	Cód.Status = 404 CSManagmt->error: Details
@Path("searchAll MyModels")	GET		Cód.Status = 200 return List<Model>;
		XML JSON	Cód.Status = 404 CSManagmt->error: Details
@Path("searchBy Keyword")	GET		Cód.Status = 200 return List<CS>;
		XML JSON	Cód.Status = 404 CSManagmt->error: Details
@Path("searchBy ReuseScope")	GET		Cód.Status = 200 return List<CS>;
		XML JSON	Cód.Status = 404 CSManagmt->error: Details
@Path("searchBy Context")	GET		Cód.Status = 200 return List<CS>;
		XML JSON	Cód.Status = 404 CSManagmt->error: Details
@Path("searchBy Domain")	GET		Cód.Status = 200 return List<CS>;
		XML JSON	Cód.Status = 404 CSManagmt->error: Details

Fonte: Elaborada pelo autor.

SERVIÇOS DO MÓDULO SERVICE MANAGMT

Tabela 20 – Serviços do Módulo Service Managmt

Services Managmt @URI("http://sosrepository-repositorydoc.1d35.starter-us-east-1.openshiftapps.com/ ServiceOrientedSoS/rest/service")			
Serviço	HTTPVerb	Consome/Produz	Retorno (Sucesso/Falha)
@Path("create")	POST	XML JSON	Cód.Status = 201 ServiceManagmt->New service created;
		TEXT_PLAIN XML	Cód.Status = 400 Cód.Status = 404 ServiceManagmt->error: Details
@Path("addModel")	POST	XML JSON	Cód.Status = 201 ServiceManagmt->New model added;
		TEXT_PLAIN XML	Cód.Status = 400 ServiceManagmt->error: Details
@Path("edit")	PUT	XML JSON	Cód.Status = 200 ServiceManagmt-> Service edited;
		TEXT_PLAIN XML	Cód.Status = 404 ServiceManagmt->error: Details
@Path("editModel")	PUT	XML JSON	Cód.Status = 200 ServiceManagmt->New model edited;
		TEXT_PLAIN XML	Cód.Status = 400 ServiceManagmt->error: Details
@Path("changeState")	PUT		Cód.Status = 200 ServiceManagmt->Service state edited;
		TEXT_PLAIN XML	Cód.Status = 400 Cód.Status = 404 ServiceManagmt->error: Details
@Path("delete")	DELETE		Cód.Status = 200 ServiceManagmt->Service deleted;

		TEXT_PLAIN XML	Cód.Status = 404 ServiceManagmt->error: Details
@Path("deleteModel")	DELETE		Cód.Status = 200 ServiceManagmt->Model deleted;
		TEXT_PLAIN XML	Cód.Status = 404 ServiceManagmt->error: Details
@Path("search")	GET		Cód.Status = 200 return Service;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchService Type")	GET		Cód.Status = 200 return ServiceType;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchService ByName")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchAll")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchAll ByStatus")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 400 Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchMy Services")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchMy ServicesByStatus")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 400 Cód.Status = 400 ServiceManagmt->error: Details
@Path("searchModels")	GET		Cód.Status = 200 return List<Model>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchAll MyModels")	GET		Cód.Status = 200 return List<Model>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchBy Keyword")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchBy ReuseScope")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchBy Context")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchBy Domain")	GET		Cód.Status = 200 return List<Service>;

		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchBy ServiceType")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchAll ServiceTypes")	GET		Cód.Status = 200 return List<ServiceType>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details
@Path("searchBy Constituent")	GET		Cód.Status = 200 return List<Service>;
		XML JSON	Cód.Status = 404 ServiceManagmt->error: Details

Fonte: Elaborada pelo autor.

SERVIÇOS DO MÓDULO SOS MANAGMT

Tabela 21 – Serviços do Módulo SoS Managmt

SoS Managmt @URI("http://sosrepository-repositorydoc.1d35.starter-us-east-1.openshiftapps.com/ ServiceOrientedSoS/rest/sos")			
Serviço	HTTPVerb	Consome/Produz	Retorno (Sucesso/Falha)
@Path("create")	POST	XML JSON	Cód.Status = 201 SoSManagmt->New SoS created;
		TEXT_PLAIN XML	Cód.Status = 400 SoSManagmt->error: Details
@Path("addModel")	POST	XML JSON	Cód.Status = 201 SoSManagmt->New model added;
		TEXT_PLAIN XML	Cód.Status = 400 SoSManagmt->error: Details
@Path("addCatalogOf TrustableSystem")	POST	XML JSON	Cód.Status = 201 SoSManagmt-> New catalog added;
		TEXT_PLAIN XML	Cód.Status = 400 SoSManagmt->error: Details
@Path("addStakeHolders")	POST	XML JSON	Cód.Status = 201 SoSManagmt-> Stakeholders added;
		TEXT_PLAIN XML	Cód.Status = 400 SoSManagmt->error: Details
@Path("edit")	PUT	XML JSON	Cód.Status = 200 SoSManagmt-> SoS edited;
		TEXT_PLAIN XML	Cód.Status = 404 SoSManagmt->error: Details
@Path("editModel")	PUT	XML JSON	Cód.Status = 200 SoSManagmt->New model edited;
		TEXT_PLAIN XML	Cód.Status = 400 SoSManagmt->error: Details
@Path("changeState")	PUT		Cód.Status = 200 SoSManagmt->SoS state edited;

		TEXT_PLAIN XML	Cód.Status = 400 Cód.Status = 404 SoSManagmt->error: Details
@Path("changeStateCatalogOfTrustableSystem")	PUT	XML JSON	Cód.Status = 200 SoSManagmt-> Catalog edited;
		TEXT_PLAIN XML	Cód.Status = 400 SoSManagmt->error: Details
@Path("editStakeHolders")	PUT	XML JSON	Cód.Status = 200 SoSManagmt-> Stakeholders edited;
		TEXT_PLAIN XML	Cód.Status = 400 SoSManagmt->error: Details
@Path("delete")	DELETE		Cód.Status = 200 SoSManagmt->SoS deleted;
		TEXT_PLAIN XML	Cód.Status = 404 SoSManagmt->error: Details
@Path("deleteModel")	DELETE		Cód.Status = 200 SoSManagmt->Model deleted;
		TEXT_PLAIN XML	Cód.Status = 404 SoSManagmt->error: Details
@Path("deleteCatalogOfTrustableSystem")	DELETE		Cód.Status = 200 SoSManagmt-> Catalog deleted;
		TEXT_PLAIN XML	Cód.Status = 404 SoSManagmt->error: Details
@Path("deleteStakeHolders")	DELETE		Cód.Status = 200 SoSManagmt-> Stakeholders deleted;
		TEXT_PLAIN XML	Cód.Status = 404 SoSManagmt->error: Details
@Path("search")	GET		Cód.Status = 200 return SoS;
		XML JSON	Cód.Status = 404 SoSManagmt->error: Details
@Path("searchAll")	GET		Cód.Status = 200 return List<SoS>;
		XML JSON	Cód.Status = 404 SoSManagmt->error: Details
@Path("searchAllByStatus")	GET		Cód.Status = 200 return List<SoS>;
		XML JSON	Cód.Status = 400 Cód.Status = 404 SoSManagmt->error: Details
@Path("searchMySoSs")	GET		Cód.Status = 200 return List<SoS>;
		XML JSON	Cód.Status = 404 SoSManagmt->error: Details
@Path("searchMySoSsByStatus")	GET		Cód.Status = 200 return List<SoS>;
		XML JSON	Cód.Status = 400 Cód.Status = 400 SoSManagmt->error: Details
@Path("searchModels")	GET		Cód.Status = 200 return List<Model>;

		XML JSON	Cód.Status = 404 SoSManagmt->error: Details
@Path("searchAll MyModels")	GET		Cód.Status = 200 return List<Model>;
		XML JSON	Cód.Status = 404 SoSManagmt->error: Details
@Path("searchBy Keyword")	GET		Cód.Status = 200 return List<SoS>;
		XML JSON	Cód.Status = 404 SoSManagmt->error: Details
@Path("searchBy ReuseScope")	GET		Cód.Status = 200 return List<SoS>;
		XML JSON	Cód.Status = 404 SoSManagmt->error: Details
@Path("searchBy Context")	GET		Cód.Status = 200 return List<SoS>;
		XML JSON	Cód.Status = 404 SoSManagmt->error: Details
@Path("searchBy Domain")	GET		Cód.Status = 200 return List<SoS>;
		XML JSON	Cód.Status = 404 SoSManagmt->error: Details

Fonte: Elaborada pelo autor.