**SoS-Reúso: um SoS do tipo direcionado para facilitar o
reúso de software**

Iohan Gonçalves Vargas

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

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Iohan Gonçalves Vargas

SoS-Reúso: um SoS do tipo direcionado para facilitar o reúso de software

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
Novembro de 2017

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

V64s Vargas, Iohan Gonçalves
 SoS-Reúso: um SoS do tipo direcionado para
 facilitar o reúso de software / Iohan Gonçalves
 Vargas; orientadora Rosana Teresinha Vaccare Braga.
 -- São Carlos, 2017.
 187 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, 2017.

 1. Reúso de Software. 2. Sistemas-de-Sistemas.
 3. Ativos Reusáveis. 4. Comportamento Emergente. I.
 Braga, Rosana Teresinha Vaccare, orient. II. Título.

Iohan Gonçalves Vargas

**SoS-Reuse: a directed SoS aimed at facilitating software
reuse**

Master dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – 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
November 2017

*Dedico esta dissertação de mestrado primeiramente a Deus,
por ser essencial em minha vida, à minha família, por sua capacidade de acreditar em mim.
Mãe seu cuidado e dedicação me manteve firme para seguir.
Pai, seu apoio significou segurança e certeza de que não estava sozinho nessa caminhada.*

AGRADECIMENTOS

Se você está lendo este agradecimento é porque eu consegui chegar até aqui. E posso lhe dizer que não foi fácil subir degrau a degrau para aqui estar. Do processo seletivo, passando pela aprovação, qualificação e, por último a conclusão do mestrado, foi um caminho longo percorrido.

Primeiramente agradeço a Deus por ter me abençoado durante todo o tempo e ter me proporcionado saúde, inteligência e muitas coisas boas durante essa caminhada. Agradeço também pelas pessoas que o Senhor colocou em meu caminho. Algumas delas me inspiram, me ajudam, me desafiam e me encorajam a ser cada dia melhor.

Gostaria de imensamente agradecer também a meus pais, Adevenir Gonçalves e Hélida Vargas, meu irmão, Luan Gonçalves, que são MEUS exemplos de vida e me deram a oportunidade de estudar e buscar ser alguém na vida. A vocês meus pais meu infinito agradecimento, por acreditar na minha capacidade de vencer, por terem me dado educação, valores e ensinado a andar com minhas próprias pernas. Sem vocês nada disso teria acontecido, pois muitas vezes, renunciaram aos seus sonhos para que eu pudesse realizar o meu, partilho com vocês a alegria deste momento.

Agradeço também a todos os meus familiares, minha vó, primos, tios e amigos. Não citarei nomes, para não me esquecer de ninguém. Mas há aquelas pessoas especiais que diretamente fazem parte da sua história e que me incentivaram. Obrigado pelo otimismo e pela solidariedade perante minhas dificuldades.

Não posso aqui deixar de citar o agradecimento à minha orientadora deste trabalho de mestrado, Profa. Dra. Rosana Teresinha Vaccare Braga, que acolheu grandiosamente esse goiano em terras paulistas e me deu apoio necessário para realizar esse trabalho durante esses anos. Agradeço pela sua enorme ajuda na sistematização dessa pesquisa, por me ensinar um pouco mais desse ambiente acadêmico e diversos temas abordados nesta dissertação. Agradeço por isto e, em especial, pelas inúmeras horas gastas e dedicadas a correções de artigos, qualificação, dissertação e diversas outras tarefas que realizamos ao longo desta jornada. Você se tornou referência profissional e pessoal para meu crescimento. Obrigada por estar a meu lado e acreditar no meu potencial.

Agradeço, também, à CAPES pelo apoio financeiro, à FAPESP, à USP - São Carlos, em especial o ICMC e toda sua equipe de funcionários que se colocaram à disposição para solucionar problemas pelos quais, sempre estamos vulneráveis a passar.

Por fim, vale lembrar que: **NINGUÉM VENCE SOZINHO, OBRIGADO A TODOS.**

*“Sem sonhos, a vida não tem brilho.
Sem metas, os sonhos não tem alicerces.
Sem prioridades, os sonhos não se tornam reais. Sonhe, trace metas, estabeleça prioridades e
corra riscos para executar seus sonhos.
Melhor é errar por tentar do que errar por omitir ”
(Augusto Cury)*

RESUMO

VARGAS, G. I. **SoS-Reúso: um SoS do tipo direcionado para facilitar o reúso de software**. 2017. 187 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, 2017.

Reúso de software, principalmente quando apoiado por ferramentas computacionais, é uma forma de enfrentar os constantes desafios da Engenharia de Software em aumentar a produtividade e qualidade no desenvolvimento de software. Diversos ambientes de apoio ao reúso de software têm sido propostos, em sua maioria com objetivos específicos, por exemplo, repositórios de componentes, ferramentas para geração de aplicações e ferramentas para configuração de linhas de produtos. Entretanto, a integração desses ambientes é, muitas vezes, deficiente e deixada por conta do próprio desenvolvedor. Por outro lado, Sistemas-de-Sistemas (SoS) constituem uma forma de integrar sistemas independentes e seus relacionamentos, formando um todo maior que a soma das partes. Dessa forma, o SoS permite alcançar objetivos complexos que não poderiam ser facilmente alcançados individualmente pelos seus sistemas constituintes. Em um SoS, comportamentos emergentes podem surgir a qualquer momento ou terem sido previamente implementados. Por serem recentes, os conceitos de SoS ainda apresentam uma vasta gama de tópicos em aberto, entre eles sua aplicação a ambientes de reúso de software. Assim, neste trabalho, investigou-se como integrar ambientes de reúso com base nos conceitos de SoS. O SoS resultante, denominado SoS-Reúso, visa facilitar a implementação de comportamentos emergentes relacionados ao reúso de software. Com isso, pode-se potencializar a atividade de reúso, facilitando a busca por ativos reusáveis em diferentes sistemas constituintes. Para colocar em prática essa ideia, desenvolveu-se um simulador do SoS-Reúso, denominado P-SoS-Reúso. Realizou-se um estudo qualitativo da usabilidade e adequação funcional do P-SoS-Reúso, a fim de obter do usuário *feedbacks* em relação ao seu uso. A abordagem utilizada no desenvolvimento do SoS-Reúso poderá ser adaptada futuramente para integrar outros tipos de sistemas, os quais poderiam se beneficiar da abordagem baseada em SoS.

Palavras-chave: Reúso de Software, Ativos Reusáveis, Comportamento Emergente, RAS, Integração, Interoperabilidade.

ABSTRACT

VARGAS, G. I. **SoS-Reuse: a directed SoS aimed at facilitating software reuse**. 2017. 187 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, 2017.

Software reuse, specially when supported by computational tools, is a way to face the constant challenges of Software Engineering in increasing productivity and quality in software development. Several environments to support software reuse have been proposed, mostly for specific purposes, for example, component repositories, application generation tools, and tools for configuring product lines. However, the integration among these environments is often deficient and left to the developers's responsibility. On the other hand, Systems-of-Systems (SoS) is a way of integrating independent systems and their relationships, forming a whole greater than the sum of the parts. In this way, SoS allows us to achieve complex goals that could not be easily achieved individually by their constituent systems. In a SoS, emergent behaviors can arise at any time or have been previously implemented. Because they are recent, SoS concepts still feature a variety of open research topics, including their application to software reuse environments. Thus, in this work, we investigated how to integrate reuse environments based on SoS concepts. The resulting SoS, called SoS-Reuse, aims to facilitate the implementation of emergent behaviors related to software reuse. This allows to enhance the reuse activity, facilitating the search for reusable assets in different constituent systems. To implement this idea, it was developed a simulator of the SoS-Reuse, named P-SoS-Reuse. An qualitative study of the usability and functional adequacy of the P-SoS-Reuse was performed to obtain feedback from users regarding its use. The approach used to develop SoS-Reuse can be adapted in the future to integrate other types of systems, which could benefit from the SoS-based approach.

Keywords: Software Reuse, Reusable Asset, Systems-of-Systems, Emergent Behavior, RAS, Integration, Interoperability.

LISTA DE ILUSTRAÇÕES

Figura 1 – Metodologia Utilizada para Desenvolvimento deste Trabalho de Mestrado	30
Figura 2 – Técnicas de Reúso de Software	36
Figura 3 – Papéis Desempenhados por um Serviço Agregador	38
Figura 4 – Exemplo de Orquestração de Serviço	39
Figura 5 – Exemplo de Coreografia	40
Figura 6 – Definição Geral do Ativo	41
Figura 7 – Principais Seções do Core RAS	42
Figura 8 – Principais Seções do Modelo de Domínio do Core RAS	42
Figura 9 – Metamodelo da RAS	43
Figura 10 – Visão Geral da AIRES	46
Figura 11 – Processo de Seleção de Estudos Relevantes para o Mapeamento Sistemático	59
Figura 12 – Relação da Quantidade de Artigos Publicados por Ano e Tipo de Publicação do Mapeamento Sistemático	60
Figura 13 – Processo de Seleção de Estudos Relevantes	67
Figura 14 – Relação da Quantidade de Artigos Publicados por Ano e Tipo de Publicação da Revisão Sistemática	68
Figura 15 – Frequência de Palavras para o Tema da RS	68
Figura 16 – Processo para Engenharia de SoS	75
Figura 17 – Modelo Conceitual Ilustrando a Solução Proposta para SoS Direcionado	79
Figura 18 – Visão Geral da Solução Proposta	81
Figura 19 – Definição da Interface de Integração	84
Figura 20 – Casos de Uso das Funções dos Atores	85
Figura 21 – Figura Geral do SoS-Reuse-Sim	86
Figura 22 – Modelagem Conceitual do SoS-Reuse-Sim	89
Figura 23 – Diagrama de Sequência de Composição do SoS-Reuse-Sim	91
Figura 24 – Diagrama de Sequência para Sugerir o Melhor Ativo Reusável do SoS-Reuse- Sim	92
Figura 25 – Passos Realizados Durante a Execução de um Comportamento Emergente	93
Figura 26 – Informações do Banco de Dados	94
Figura 27 – Código em .xml	95
Figura 28 – Principais Pacotes Desenvolvidos no Simulador	95
Figura 29 – Principal Tela do Simulador do SoS	97
Figura 30 – Primeira Tela do SoS-User	98

Figura 31 – Informações Sobre o Melhor Ativo	99
Figura 32 – Ativos Relacionados ao Escolhido como o Melhor	100
Figura 33 – Tela Principal do SoS-Admin	101
Figura 34 – Tela de Visualização dos Ativos e Artefatos de um CS	102
Figura 35 – Relação da Quantidade de Participantes X Idade	108
Figura 36 – Nível de Escolaridade dos Participantes	108
Figura 37 – Nível de Conhecimento sobre SoS	109
Figura 38 – Nível de Conhecimento sobre Reúso de Software	109
Figura 39 – Nível de Conhecimento sobre RAS	110
Figura 40 – Experiência com Desenvolvimento de Software	110
Figura 41 – Nível de Satisfação com a Busca por Ativos	111
Figura 42 – Pergunta em Relação aos Atributos de Busca	112
Figura 43 – Visualização dos Atributos do Ativo	112
Figura 44 – Avaliação dos Atributos de ativo	113
Figura 45 – Avaliação de Quão Útil é o Simulador Desenvolvido	113
Figura 46 – Avaliação de Quão Fácil é Buscar Automaticamente os Ativos	114
Figura 47 – Avaliação da Importância de Sugerir o Melhor Ativo	115
Figura 48 – Avaliação da Facilidade de Visualização dos Ativos	115
Figura 49 – Avaliação da Hierarquia dos CS's	116
Figura 50 – Avaliação do Treinamento Realizado	116
Figura 51 – Avaliação do Quão Trabalhoso é Publicar um CS	118
Figura 52 – Avaliação das Funções de Adicionar/Remover/Editar	118
Figura 53 – Avaliação da Importância de Manter o Catálogo de Sistemas Confiáveis	119
Figura 54 – Processo de Seleção de Estudos Relevantes para o Mapeamento Sistemático	149
Figura 55 – Relação Quantidade por Máquina de Busca e Porcentagem do Mapeamento Sistemático	150
Figura 56 – Relação Quantidade de Artigos Publicados por Ano e Tipo de Publicação do Mapeamento Sistemático	150
Figura 57 – <i>Word Cloud</i> para o Tema do MS	151
Figura 58 – Processo de Seleção de Estudos Relevantes	163
Figura 59 – Relação da Quantidade por Máquina de Busca e Porcentagem	164
Figura 60 – Relação da Quantidade de Artigos Publicados por Ano e Tipo de Publicação	165
Figura 61 – Frequência de Palavras para o Tema da RS	169
Figura 62 – Visão Geral dos Artigos que Utilizaram SOA	184

LISTA DE TABELAS

Tabela 1 – Benefícios do Reúso de Software	35
Tabela 2 – Problemas do Reúso de Software	36
Tabela 3 – Casos de Interoperabilidade no Contexto de SoS	52
Tabela 4 – Nível de Esforço de Integração	54
Tabela 5 – Respostas Obtidas para FE1	69
Tabela 6 – Domínio de Aplicação	70
Tabela 7 – Vantagens Apontadas Pelos Estudos da RS	72
Tabela 8 – Atores Identificados na Abordagem para SoS Direcionado	82
Tabela 9 – Detalhamento das Etapas do Estudo Qualitativo	107
Tabela 10 – <i>Feedback</i> dos Usuários em Relação a Ideia do Simulador sob o Ponto de Vista do SoS-User	117
Tabela 11 – <i>Feedback</i> dos Usuários em Relação a Ideia do Simulador sob o Ponto de Vista do SoS-Admin	120
Tabela 12 – Sugestões de Melhorias dos Usuários no papel de SoS-Admin	120
Tabela 13 – Dificuldades Encontradas pelos Usuários no papel de SoS-Admin	121
Tabela 14 – Bases de Dados Utilizadas no Mapeamento Sistemático	147
Tabela 15 – Artigos Selecionados para Leitura Transversal com seus Respetivos Anos de Publicação	152
Tabela 16 – Bases de Dados Utilizadas na Revisão Sistemática	161
Tabela 17 – Formulário de Avaliação de Qualidade - Cobertura das Questões de Pesquisa	165
Tabela 18 – Formulário de Avaliação de Qualidade - Cobertura Sobre o Estudo	166
Tabela 19 – Pontuação dos Critérios de Avaliação de Qualidade	166
Tabela 20 – Artigos Selecionados para Leitura Completa com seus Respetivos <i>Score</i> de Avaliação de Qualidade	167
Tabela 21 – Visão Geral das Respostas da Revisão Sistemática	168
Tabela 22 – Respostas Obtidas para FE1 da Revisão Sistemática	170
Tabela 23 – Domínio de Aplicação	171
Tabela 24 – Vantagens Apontadas pelos Estudos da Revisão Sistemática	174
Tabela 25 – Padrão Arquitetural	180
Tabela 26 – Nível de Esforço de Integração	181

SUMÁRIO

1	INTRODUÇÃO	25
1.1	Contextualização e Motivação	25
1.2	Objetivos	29
1.3	Metodologia	30
1.4	Organização	31
2	FUNDAMENTAÇÃO TEÓRICA	33
2.1	Considerações Iniciais	33
2.2	Conceitos Básicos de Reúso de Software	33
2.2.1	<i>Arquitetura Orientada a Serviços - SOA</i>	37
2.2.2	<i>Ativos Reusáveis e RAS</i>	40
2.2.2.1	<i>Ativos Reusáveis</i>	40
2.2.2.2	<i>Reusable Asset Specification - RAS</i>	41
2.2.3	<i>Ambientes de Reúso de Software</i>	43
2.2.4	<i>AIRES</i>	45
2.3	Sistemas-de-Sistemas	46
2.3.1	<i>Conceitos Básicos</i>	46
2.3.2	<i>Características de um SoS</i>	48
2.3.3	<i>Tipos de Sistemas-de-Sistemas</i>	49
2.3.4	<i>Interoperabilidade de SoS</i>	51
2.3.5	<i>Integração de Sistemas-de-Sistemas</i>	53
2.3.6	<i>Aplicação de SOA no Contexto de SoS</i>	55
2.4	Considerações Finais	55
3	MAPEAMENTO E REVISÃO SISTEMÁTICA DA LITERATURA	57
3.1	Considerações Iniciais	57
3.2	Integração de Ambientes de Reúso de Software - Um Mapeamento Sistemático	57
3.2.1	<i>String de Pesquisa e Máquinas de Busca</i>	58
3.2.2	<i>Resultados do MS</i>	59
3.2.3	<i>Resposta obtida para a subquestão FE 1- Existem definições de ambientes de reúso de software?</i>	60

3.2.4	<i>Resposta obtida para a subquestão FE 2- Como tem sido feita a integração entre ferramentas de reúso?</i>	61
3.2.5	<i>Resposta obtida para a subquestão FE 3- Utiliza-se alguma ferramenta de apoio à integração?</i>	63
3.2.6	<i>Resposta obtida para a subquestão FE 4- O artigo citou dificuldades e/ou desafios a serem superados?</i>	63
3.2.7	<i>Resposta obtida para a subquestão FE 5- SoS tem sido empregado como uma possível solução de integração entre ambientes de Reúso?</i>	64
3.2.8	<i>Conclusão Sobre as Questões de Pesquisa</i>	64
3.3	Abordagens para Integração em Sistemas-de-Sistemas (SoS) - Uma Revisão Sistemática	64
3.3.1	<i>String de Pesquisa e Máquinas de Busca</i>	65
3.3.2	<i>Resultados - Revisão Sistemática</i>	66
3.3.3	<i>Resultados - Revisão SoSI</i>	66
3.3.4	<i>Resposta obtida para a subquestão FE1 - Qual é a proposta do artigo em relação à integração de SoS?</i>	67
3.3.5	<i>Resposta obtida para a subquestão FE 1.1 - Qual contribuição de integração de SoS foi abordada pelo estudo?</i>	69
3.3.6	<i>Resposta Obtida para a Subquestão FE 2 - Qual o Domínio de Aplicação que Tem Sido Alvo no Contexto de Integração de SoS?</i>	69
3.3.7	<i>Resposta obtida para a subquestão FE 2.1 - Qual o tipo de SoS foi utilizado?</i>	69
3.3.8	<i>Resposta Obtida para a Subquestão FE 2.1.1 - Quais as Características do SoS Foram Detalhadas?</i>	70
3.3.9	<i>Resposta Obtida para a Subquestão FE 2.2 - Quais Problemas Referentes à Integração de SoS Foram Abordados?</i>	71
3.3.10	<i>Resposta obtida para a subquestão FE 2.3 - Quais as vantagens obtidas quando se utiliza o conceito de SoS para integrar sistemas?</i>	72
3.3.11	<i>Resposta obtida para a subquestão FE 3 - O estudo apresentou alguma ferramenta de integração?</i>	72
3.3.12	<i>Conclusão Sobre Questões de Pesquisa</i>	73
3.4	Trabalhos Relacionados	74
3.5	Considerações Finais	76
4	UMA ABORDAGEM PARA DESENVOLVIMENTO DE SOS DIRECIONADO - APLICAÇÃO NO SOS-REÚSO	77
4.1	Considerações Iniciais	77
4.2	Uma Abordagem para Desenvolvimento de SoS Direcionado	78
4.2.1	<i>Conceitos Básicos da Proposta de SoS Direcionado</i>	78
4.2.2	<i>Modelagem Conceitual e Visão Geral do SoS Direcionado</i>	79

4.2.3	<i>Casos de Uso para o SoS Direcionado</i>	81
4.3	SoS-Reúso: Concepção de um SoS Direcionado para Reúso de Software	84
4.4	SoS-Reuse-Sim: um Simulador do SoS-Reúso	85
4.5	Modelagem do SoS-Reuse-Sim	88
4.5.1	<i>Modelo Conceitual</i>	88
4.5.2	<i>Diagrama de Sequência de Composição do SoS-Reuse-Sim</i>	91
4.5.3	<i>Diagramas do Comportamento Emergente - Which is the best asset?</i>	92
4.6	Implementação	93
4.7	Visão Geral das Funcionalidades do Simulador	96
4.7.1	<i>Funcionalidades do SoS-User</i>	96
4.7.2	<i>Funcionalidades do SoS-Admin</i>	100
4.8	Considerações Finais	102
5	ESTUDO QUALITATIVO DA USABILIDADE E ADEQUAÇÃO FUNCIONAL DO P-SOS-REÚSO	103
5.1	Considerações Iniciais	103
5.2	Definição e Planejamento do Estudo Qualitativo	104
5.2.1	<i>Definição de Metas</i>	105
5.3	Operação do Estudo Qualitativo	106
5.4	Análise e Interpretação dos Resultados	107
5.4.1	<i>Questionário Pessoal</i>	107
5.4.2	<i>Questionário Sobre o Papel do SoS-User</i>	110
5.4.3	<i>Questionário Sobre o papel SoS-Admin</i>	117
5.5	Discussão dos Resultados	121
5.6	Ameaças à Validade	122
5.7	Considerações Finais	123
6	CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS	125
6.1	Visão Geral da Pesquisa	125
6.2	Contribuições	126
6.3	Trabalhos Futuros	126
6.4	Produção Científica	127
6.4.1	<i>Publicações Elaboradas</i>	127
6.4.2	<i>Publicações em Andamento</i>	127
	REFERÊNCIAS	129
	GLOSSÁRIO	141

APÊNDICE A	INTEGRAÇÃO DE AMBIENTES DE REÚSO DE SOFTWARE - UM MAPEAMENTO SISTEMÁTICO	143
A.1	Planejamento	144
A.1.1	<i>Background: Justificativa para o Mapeamento Sistemático.</i>	144
A.1.2	<i>Estratégia para Seleção dos Estudos Primários</i>	145
A.1.3	<i>String de Pesquisa e Máquinas de Busca</i>	146
A.2	Condução do Mapeamento	147
A.2.1	<i>Critérios de Inclusão e Exclusão</i>	147
A.2.2	<i>Procedimentos para Seleção dos Estudos</i>	148
A.3	Resultados - Mapeamento Sistemático	151
A.3.1	<i>Resposta obtida para a subquestão FE 1- Existem definições de ambientes de reúso de software?</i>	153
A.3.2	<i>Resposta obtida para a subquestão FE 2- Como tem sido feita a integração entre ferramentas de reúso?</i>	153
A.3.3	<i>Resposta obtida para a subquestão FE 3- Utiliza-se alguma ferramenta de apoio à integração?</i>	155
A.3.4	<i>Resposta obtida para a subquestão FE 4- O artigo citou dificuldades e/ou desafios a serem superados?</i>	155
A.3.5	<i>Resposta obtida para a subquestão FE 5- SoS tem sido empregado como uma possível solução de integração entre ambientes de Reúso?</i>	156
A.4	Conclusão sobre as Questões de Pesquisa	156
A.4.1	<i>RQ1. Como tem sido investigada a integração entre ambientes de reúso de software?</i>	156
A.4.2	<i>RQ2. SoS tem sido apontado como uma possível solução para integrar ambientes de reúso?</i>	156
APÊNDICE B	ABORDAGENS PARA INTEGRAÇÃO EM SISTEMAS-DE-SISTEMAS - UMA REVISÃO SISTEMÁTICA . .	157
B.1	Planejamento	158
B.1.1	<i>Background: Justificativa para a Revisão Sistemática.</i>	158
B.1.2	<i>Estratégia para Seleção dos Estudos Primários</i>	160
B.1.3	<i>Conduzindo a busca por Estudos Primários</i>	160
B.1.4	<i>Seleção dos Estudos Primários</i>	161
B.2	Condução da Revisão	162
B.2.1	<i>Procedimentos para Seleção dos Estudos</i>	162
B.2.2	<i>Critérios de Avaliação de Qualidade</i>	164
B.2.3	<i>Sumarização dos Dados</i>	166
B.2.4	<i>Mapeamento de Dados</i>	168
B.3	Resultados - Revisão SoSI	169

<i>B.3.1</i>	<i>Resposta obtida para a subquestão FE1 - Qual é a proposta do artigo em relação à integração de SoS?</i>	<i>170</i>
<i>B.3.2</i>	<i>Resposta obtida para a subquestão FE 1.1 - Qual contribuição de integração de SoS foi abordada pelo estudo?</i>	<i>170</i>
<i>B.3.3</i>	<i>Resposta obtida para a subquestão FE 2 - Qual o domínio de aplicação que tem sido alvo no contexto de integração de SoS?</i>	<i>171</i>
<i>B.3.4</i>	<i>Resposta obtida para a subquestão FE 2.1 - Qual o tipo de SoS foi utilizado?</i>	<i>171</i>
<i>B.3.5</i>	<i>Resposta obtida para a subquestão FE 2.1.1 - Quais as características do SoS foram detalhadas?</i>	<i>172</i>
<i>B.3.6</i>	<i>Resposta obtida para a subquestão FE 2.2 - Quais problemas referentes à integração de SoS foram abordados?</i>	<i>172</i>
<i>B.3.7</i>	<i>Resposta obtida para a subquestão FE 2.3 - Quais as vantagens obtidas quando utiliza o conceito de SoS para integrar sistemas? .</i>	<i>174</i>
<i>B.3.8</i>	<i>Resposta obtida para a subquestão FE 3 - O estudo apresentou alguma ferramenta de integração?</i>	<i>174</i>
<i>B.4</i>	<i>Contribuições dos artigos selecionados para extração de dados . . .</i>	<i>175</i>
<i>B.5</i>	<i>Conclusão sobre Questões de Pesquisa</i>	<i>183</i>
<i>B.5.1</i>	<i>RQ1. Como tem sido investigada a integração entre os membros de um SoS?</i>	<i>183</i>
<i>B.5.2</i>	<i>RQ2. Nesse tipo de estudo, tem sido utilizada alguma ferramenta que auxilia na integração dos sistemas constituintes?</i>	<i>183</i>
APÊNDICE C	INFORMAÇÕES COMPLEMENTARES	185
C.1	Questionário Inicial	185
C.2	Questionário do SoS-User	185
C.3	Questionário do SoS-Admin	186

INTRODUÇÃO

1.1 Contextualização e Motivação

Melhorar a produtividade e qualidade de processos de desenvolvimento, sem deixar de balancear os custos, é um dos desafios constantes da engenharia de software. Técnicas de reúso de software são essenciais para superar esses desafios, por facilitar a reutilização de artefatos de qualidade produzidos previamente, tais como documentos de requisitos, modelos de análise e projeto, código fonte, casos de teste e documentação. Um grande número de técnicas, processos e ferramentas para alavancar o reúso tem sido propostas na literatura, em geral voltadas a formas específicas de reúso, por exemplo, componentes, linha de produtos, *frameworks* e padrões (SOMMERVILLE, 2011).

Com o aumento da complexidade no desenvolvimento de software, é essencial a utilização de suporte computacional provido por ferramentas ou ambientes de desenvolvimento. Esse suporte ajuda a melhorar a produtividade e qualidade de processos de desenvolvimento, ao passo que contribui para a obtenção de menores custos e *time-to-market*. Sendo assim, são desafios constantes para a engenharia de software. Técnicas de reúso de software são essenciais para dar suporte a esses desafios, por facilitarem o reúso de artefatos que foram produzidos previamente (FRAKES; TERRY, 1996).

Desenvolvedores de software usualmente empregam mais de uma técnica de reúso, tendo que lidar com diferentes ambientes, ferramentas, processos e técnicas. O problema na maioria dos casos, é que essas técnicas de reúso são independentes e tratadas de forma isolada umas das outras, o que pode levar à dificuldade de integração entre artefatos produzidos em diferentes projetos (CHEN; CHENG, 1997). O uso isolado de ambientes de desenvolvimento faz com que a rastreabilidade entre os ativos reusáveis fique sob responsabilidade do engenheiro de software. Para alcançar a cooperação entre as diversas técnicas de reúso e potencializar o reúso de software, precisa-se realizar alguma forma de integração. Porém, a falta de uma integração apropriada

dessas técnicas e a falta de um suporte computacional impõem obstáculos significativos que impedem de alcançar o reuso efetivo durante o processo de desenvolvimento de software (CHEN; CHENG, 1997).

A integração é amplamente utilizada em diferentes áreas. Integração de aplicação é uma abordagem estratégica para vincular muitos sistemas de informação, tanto no nível de serviço quanto no nível de informação, apoiando a capacidade de trocar informações e alavancar processos em tempo real. A integração pode assumir várias formas, incluindo a integração de aplicações internas (do inglês, *Enterprise Application Integration* - EAI) ou integração de aplicações externas (do inglês, *Business-to-Business Application Integration* - B2B), sendo que cada organização e comunidade comercial tem seu próprio conjunto de problemas de integração. Por isso, é quase impossível encontrar um conjunto de soluções tecnológicas que podem ser aplicadas universalmente (LINTHICUM, 2003).

Uma tendência clara é a movimentação de integração baseada em informação para a baseada em serviços (LINTHICUM, 2003). A integração orientada a informação fornece um mecanismo de baixo custo porque, na maioria dos casos, não há necessidade de alterar as aplicações. A integração com base em serviços não é uma abordagem nova, e fornece um mecanismo que integra aplicações por meio de serviços da Web, sendo capaz de aproveitar o poder da internet para fornecer acesso a serviços de aplicações remotas por meio de um serviço bem definido de interface e diretório: Descrição Universal, Descoberta e Integração (do inglês, *Universal Description, Discovery and Integration* - UDDI) (LINTHICUM, 2003). A seguir são listadas algumas formas de integração (LINTHICUM, 2003):

- **Orientada à Informação:** a abordagem orientada à informação (do inglês, *Information-Oriented Application Integration* - IOAI) deve ocorrer entre os bancos de dados (ou APIs proprietárias que produzem informações) - ou seja, bancos de dados ou API's produtoras de informação devem ser vistos como os principais pontos de integração. É importante entender que essa abordagem lida com informações simples, e não com o processo ou serviços da aplicação.
- **Processos de Negócios Orientado à Integração:** (do inglês, *Business Process Integration*) é uma camada superior de processos definidos e gerenciados centralmente dentro do conjunto de aplicações da empresa.
- **Integração de Aplicação Orientada a Serviços:** essa abordagem permite que as aplicações compartilhem métodos ou lógica de negócios comuns. O princípio fundamental é que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de *Web Services*. Esse estilo arquitetural permite desenvolver os sistemas com independência de linguagem de programação e flexibilidade quanto à plataforma, uma vez que funcionalidades oferecidas por serviços são definidas por meio de linguagens de descrição padronizadas.

Diante das diversas formas de integração, combinar diferentes interesses individuais ou do grupo, em diferentes níveis e ferramentas, ainda é uma tarefa difícil (FICHMAN; KEMERER, 2001). O processo de desenvolvimento de software é colaborativo e requer a coordenação de atividades e tarefas, além do contínuo compartilhamento de informação sobre o estado do projeto. Trabalha-se com tarefas fortemente acopladas, como edição de documentos e programas. Essa natureza colaborativa motiva a construção de uma variedade de ferramentas e ambientes para apoiar o reúso de software (WERNER *et al.*, 2003a).

Conforme mencionado anteriormente, desenvolvedores podem utilizar diferentes ambientes, de acordo com as necessidades de cada ciclo de desenvolvimento. Ferramentas dão apoio a várias tarefas do dia-a-dia do desenvolvedor; porém, ele tem que atuar como coordenador desses sistemas, fazendo as integrações necessárias por meio de operações, muitas vezes, complexas e sujeitas a erros, além de levarem à redundância de informações que ficam espalhadas em diversos sistemas.

Por outro lado, a utilização de mais de um ambiente para alcançar um objetivo maior é comum. Por exemplo, após ter especificado a arquitetura do sistema pode-se fazer buscas em um repositório de componentes até encontrar componentes prontos que satisfaçam aos requisitos elicitados; em seguida, ao projetar algumas soluções, pode-se precisar buscar padrões a serem empregados; depois, ao codificar as soluções, pode-se fazer busca em repositórios de código reusável e finalmente, ao produzir o código, ele vai sendo armazenado em gerenciadores de versão. Esses ambientes são independentes tanto operacionalmente quanto gerencialmente e poderiam ser integrados para compor um Sistemas-de-Sistemas (SoS) para atingir comportamentos emergentes, como por exemplo: identificar e sugerir o melhor ativo a ser reusado, ou identificar conflitos no uso de artefatos de diferentes ferramentas.

Atualmente, a orientadora está coordenando o desenvolvimento de uma arquitetura integrada de reúso de software, denominada AIRES. A AIRES possui funcionalidades relacionadas ao reúso de ativos, e pode ser considerada uma tentativa de possibilitar o reúso em diversos níveis de abstração (BRAGA *et al.*, 2016). Essa arquitetura baseia-se no desenvolvimento dirigido por modelos e tem arquitetura baseada em serviços, de forma que quaisquer funcionalidades para possível integração com ambientes externos serão fornecidas por meio de serviços. Sendo assim, além de outros ambientes isolados, pode-se ter instâncias da AIRES distribuídas em diversas instituições para apoiar o reúso de software. Porém, não há uma integração efetiva desses ambientes e, portanto a atividade de reúso continua impondo desafios e exigindo do engenheiro de software um controle manual de tudo isso. Como uma tentativa de superar esses desafios, acredita-se que ao utilizar SoS, pode-se realizar a integração de diversos ambientes de reúso e promover o desenvolvimento de software global apoiado por um suporte computacional. Esse suporte pode facilitar a atividade de reúso, por integrar ambientes que antes estavam espalhados por toda a parte, e conseqüentemente, permite reusar mais artefatos, ao invés de produzi-los do zero.

Sistemas-de-Sistemas (do inglês, *Systems-of-Systems* - SoS) são uma composição de

sistemas independentes e interoperáveis que visam alcançar objetivos únicos de forma coletiva (BOARDMAN; SAUSER, 2006). Um dos usos mais comuns de SoS é para compor sistemas complexos, mantendo sua independência gerencial e operacional. No entanto, há poucos indícios do uso de SoS no contexto de integração de ambientes de reúso de software.

De fato, Vargas, Gottardi e Braga (2016) realizaram uma revisão sistemática da literatura na qual evidenciaram diversos domínios em que SoS estão sendo explorados. Porém, não foram encontradas aplicações de SoS no domínio de ambientes de apoio ao reúso de software. A Integração de Sistemas-de-Sistemas (do inglês, *Systems-of-Systems Integration* - SoSI) tem se tornado uma preocupação chave na área de SoS. A SoSI destina-se a criar uma nova capacidade geral por meio da composição de sistemas que contribuem para essa capacidade geral (MADNI; SIEVERS, 2014a). Ela é uma das atividades que compõe a engenharia de SoS e tem por objetivo criar interfaces, assim como habilitar as interações entre os sistemas constituintes, a fim de criar uma nova capacidade para cumprir objetivos de negócios.

A integração de SoS é um desafio fundamental para engenharia de software, pois os diferentes sistemas não foram projetados para trabalharem juntos. Assim, garantir que eles alcancem um comportamento maior do que a soma das partes não é uma tarefa simples. A heterogeneidade dos sistemas, na maioria dos casos, pode fazer com que o processo de integração seja ainda mais desafiador (MADNI; SIEVERS, 2014a).

Integrar os ambientes de reúso de software em um SoS é mais vantajoso, pois pode-se manter a independência operacional e gerencial de cada um dos sistemas constituintes, além disso, pode-se ter diversos sistemas em diferentes posições geográficas e juntos eles podem prover uma funcionalidade que não é possível se obter nos sistemas constituintes de forma individual. Integração tradicional de sistemas geralmente significa a troca de informações entre dois ou mais sistemas sem visibilidade nos processos internos, movendo informações entre os sistemas a nível de dados. Em alguns casos os sistemas são fortemente acoplados e apresentam dependência entre eles (LINTHICUM, 2003). O que pretendeu-se alcançar neste trabalho de mestrado foi além de uma simples troca de dados, necessitou-se que os sistemas constituintes estivessem aptos a oferecer serviços, trabalhassem subordinados a um coordenador central, fossem independentes de plataforma de aplicação e pudessem juntos contribuir para a obtenção de um comportamento emergente.

Em trabalhos anteriores da orientadora deste projeto de mestrado, foi proposta uma abordagem para engenharia de SoS (RAMOS, 2014; RAMOS *et al.*, 2013; RAMOS *et al.*, 2012a) com diversas perspectivas, incluindo requisitos, análise, projeto e reengenharia. Essa abordagem foi utilizada neste trabalho como base para: (1) entender o desenvolvimento de SoS baseado em serviços; (2) entender a forma de integração realizada entre os sistemas constituintes de um SoS; (3) analisar a implementação de comportamentos emergentes; e (4) entender o papel de cada um dos atores que foram propostos neste trabalho.

Neste projeto de mestrado também manteve-se o uso de serviços como forma de integra-

ção entre os ambientes de reúso. Foi proposta uma abordagem para desenvolvimento de SoS direcionado, composto por instâncias da AIRES, que por sua vez disponibilizam ativos e contém um ou mais artefatos de software reusáveis.

Por fim, uma motivação importante para este projeto de mestrado é contribuir com um projeto de pesquisa em andamento, coordenado pela orientadora, em que está sendo proposta a MBA4SoSI (do inglês *Model-Based Architecture for System-of-Systems Integration*) (BRAGA, 2016). Ela é uma solução arquitetural que visa facilitar a integração de sistemas (SoS), utilizando SOA para favorecer a interação dos sistemas constituintes e desenvolvimento baseado em modelos (MDD – do inglês *Model-Driven Development*) (BRAMBILLA; CABOT; WIMMER, 2012; SCHMIDT, 2006), para facilitar a união de diferentes plataformas de *software* e futura evolução do SoS e seus constituintes.

Este trabalho de mestrado visa contribuir com um dos sub-objetivos da MBA4SoSI, averiguando a união dos conceitos de SoS e SOA. Em outros trabalhos pertencentes ao projeto MBA4SoSI, estão sendo investigados os objetivos referentes a MDD.

1.2 Objetivos

Este trabalho de mestrado investigou, como objetivo principal, como ambientes de reúso de software independentes podem ser compostos em um SoS para apoiar seu uso integrado, cumprindo objetivos maiores do que simplesmente a soma dos objetivos de cada ambiente individualmente. Essa investigação resultou em uma abordagem para desenvolvimento de SoS direcionado, que foi aplicada a um cenário específico de reúso de software, aqui denominado SoS-Reúso, o qual facilita a implementação de comportamentos emergentes relacionados ao reúso de software por meio da integração de ambientes já existentes. Dessa forma, é possível integrar ambientes que estavam espalhados por toda parte e em diversas localizações. O SoS-Reúso aborda os conceitos de SoS, Reúso de Software, RAS (do inglês, *Reusable Asset Specification*) e SOA (do inglês, *Service Oriented Architecture*).

Um segundo objetivo deste trabalho foi implementar um simulador do SoS-Reúso, aqui denominado SoS-Reuse-Sim, utilizando instâncias da AIRES como sistemas constituintes, e como exemplo de comportamento emergente a busca pelo melhor ativo de software. Esse simulador foi avaliado para obter *feedback* dos usuários em relação a sua adequação funcional e usabilidade.

Como objetivos complementares, a revisão sistemática e o mapeamento sistemático conduzidos neste trabalho podem ser úteis a outros pesquisadores das áreas de SoS e reúso de software. Além disso, objetivou-se a disponibilização de um novo exemplo de uso de SoS, bem como a documentação de todo o processo de desenvolvimento de uma abordagem geral para integrar sistemas e formar SoS direcionado.

1.3 Metodologia

Conforme apresentado na Figura 1, o desenvolvimento deste trabalho de mestrado dividiu-se em quatro partes, são elas: Mapeamento Sistemático (MS) e Revisão Sistemática da Literatura (RS), Desenvolvimento da Abordagem, Implementação do Simulador SoS-Reuse-Sim e Avaliação do Simulador SoS-Reúso.

Figura 1 – Metodologia Utilizada para Desenvolvimento deste Trabalho de Mestrado

Fonte: Elaborada pelo autor.

Na primeira parte, realizou-se um MS para identificar abordagens de integração de Ambientes de Reúso de Software e uma RS com o intuito de identificar abordagens para integração em Sistemas-de-Sistemas (VARGAS; GOTTARDI; BRAGA, 2016). Por meio desses

dois trabalhos, foram identificados os principais tópicos sobre reúso de software, SoS, SOA e integração de sistemas.

Na segunda parte, após caracterizar o estado da arte referente aos tópicos mencionados anteriormente e analisar os resultados do MS e da RS, observou-se a necessidade de desenvolver novas ferramentas de apoio à atividade de reúso de software com base em SoS. Então, iniciou-se a concepção, modelagem, definição de conceitos, diagramas, escopo e cenário de aplicação para desenvolvimento de um SoS do tipo direcionado para reúso de software, denominado SoS-Reúso.

Como terceira parte, desenvolveu-se um simulador do SoS-Reúso que visa dar suporte computacional ao desenvolvedor e potencializar o reúso de ativos de software, denominado de SoS-Reuse-Sim. Por fim, na quarta parte, conduziu-se um estudo experimental, para avaliar a abordagem proposta, com base nas diretrizes apresentadas por [Wohlin et al. \(2012\)](#). O experimento consistiu-se em quatro etapas: definição, planejamento, operação do experimento e análise e interpretação dos resultados. O objetivo do estudo experimental foi avaliar os dois diferentes papéis de usuário presente no SoS-Reuse-Sim, sob o ponto de vista de usabilidade e adequação funcional.

1.4 Organização

O restante do documento está organizado da seguinte forma: No Capítulo 2 é feita uma revisão dos principais conceitos relacionados ao reúso de software e Sistemas-de-Sistemas (SoS). No Capítulo 3 são sinalizados os resultados dos dois estudos de revisão da literatura que fundamentam esse trabalho de mestrado. No Capítulo 4 apresenta-se a visão geral da abordagem desenvolvida neste trabalho de mestrado. No Capítulo 5 apresenta-se o estudo qualitativo realizado para avaliar o simulador do SoS-Reúso. No Capítulo 6 apresentam-se as conclusões sobre o projeto de mestrado, contribuições, publicações e possíveis trabalhos futuros. Nos Apêndices A e B são detalhados o mapeamento e a revisão sistemática conduzidos para fundamentar e desenvolver este trabalho de mestrado.

FUNDAMENTAÇÃO TEÓRICA

2.1 Considerações Iniciais

Alguns conceitos relacionados ao reúso de software e SoS se fazem necessários para o entendimento deste trabalho. Na Seção 2.2 são apresentados os conceitos básicos de Reúso de Software com suas principais definições e terminologias. Na Seção 2.2.1 são apresentadas as definições sobre SOA. Na Seção 2.2.2 são apresentados os conceitos de RAS e as principais definições de ativos reusáveis utilizadas neste trabalho. Na Seção 2.2.3 são apresentados alguns exemplos de ambientes de reúso de software. Mais especificamente, na Seção 2.2.4 é apresentada a AIRES, uma arquitetura para melhorar o reúso de software.

Sistemas de software estão presentes em nosso cotidiano e uma classe emergente de sistemas, denominada Sistemas-de-Sistemas - SoS (do inglês *Systems-of-Systems*), apresenta fundamental importância para este trabalho de mestrado. De forma ampla, pode-se dizer que um SoS contém pelo menos uma ordem de complexidade maior do que seus sistemas constituintes (SAGE; CUPPAN, 2001).

Os conceitos de SoS são explorados neste trabalho como uma maneira de integrar ambientes de reúso de software. Dessa forma, na Seção 2.2 descrevem-se os tópicos básicos para o entendimento de Reúso de Software. Na Seção 2.3 são apresentados os conceitos relacionados a SoS, integração de SoS e termos relacionados. sinalizados os resultados da revisão sistemática conduzida para este trabalho. Por fim, na Seção 2.4 são apresentadas as considerações finais do capítulo.

2.2 Conceitos Básicos de Reúso de Software

A engenharia de software baseada em reúso é uma estratégia em que o processo de desenvolvimento é orientado para o reúso de artefatos de software existentes. A aceitação do

reúso de software foi uma resposta às exigências de menores custos de produção e manutenção, entregas mais rápidas e software de maior qualidade (BALDASSARRE *et al.*, 2005). O reúso é uma das áreas da engenharia de software que propõe um conjunto sistemático de processos, técnicas e ferramentas para obter produtos com alta qualidade, que sejam economicamente viáveis e que possibilitem aumentar o retorno sobre os investimentos de software (SOMMERVILLE, 2011).

De acordo com Krueger (1992), reúso de software é o processo de implementar ou atualizar sistemas de software usando ativos de software existentes. Baldassarre *et al.* (2005) definem como uma técnica da engenharia de software que, quando adotada sistematicamente, pode melhorar e até mesmo garantir a qualidade de software, visto que artefatos podem ser reutilizados em menor tempo do que se fossem desenvolvidos a partir do zero. Em alto nível, reúso de software é composto por dois tipos de atividades: (1) a gestão de componentes de software, incluindo a especificação, classificação e recuperação de componentes existentes e (2) o componente de integração que agrega os componentes reutilizáveis para uma aplicação.

O reúso de software é mais eficaz quando está previsto como parte de um programa de reúso de toda a organização. Um programa de reúso envolve a criação de ativos reusáveis e a adaptação de processos de desenvolvimento para incorporar esses ativos no novo software. Pode-se reusar um sistema por completo incorporando-o a outros, assim como pode-se reusar com novos objetivos, reusando componentes, classes de objeto, funções, bibliotecas, modelos, serviços, entre outros.

Um grande número de técnicas têm sido desenvolvidas para abordar questões de reúso de software (KRUEGER, 1992). No entanto, a falta de uma integração eficiente dessas técnicas impõe obstáculos significativos para alcançar a reutilização em grande escala. Diante da grande diversidade de técnicas, as unidades de software reusadas podem ser de tamanhos diferentes (SOMMERVILLE, 2011), por exemplo:

1. Reúso de sistema: um sistema pode ser reutilizado em sua totalidade sem alterações ou pela customização para diferentes clientes. Desenvolve-se então, um núcleo comum de software com diferentes adaptações para clientes específicos.
2. Reúso de componentes: componentes de diferentes tamanhos podem ser reusados.
3. Reúso de objetos e funções: componentes de software que implementam uma única função podem ser reusados. Essa forma de reúso, baseada em bibliotecas padrão, tem se tornado mais comum nos últimos anos. Pode-se reutilizar essas bibliotecas ligando-as com o código da aplicação recém desenvolvido.

Como fonte complementar, existe o reúso de conceitos, em que ao invés de reusar um componente de software, é possível reusar uma ideia, uma forma, um trabalho ou um

algoritmo. O conceito reusado é representado em uma notação abstrata que não inclui detalhes de implementação. Uma vantagem óbvia do reúso de software é a redução dos custos globais de desenvolvimento. Menos componentes de software precisam ser especificados, concebidos, implementados e validados (BALDASSARRE *et al.*, 2005).

De acordo com Sommerville (2011), existem diversos benefícios do reúso de software, alguns deles são apresentados na Tabela 1. Apesar das vantagens, o reúso de software também possui desvantagens, como as descritas na Tabela 2.

Tabela 1 – Benefícios do Reúso de Software

Benefício	Explicação
Confiança aumentada	Softwares que foram reusados, experimentados e testados provavelmente são mais confiáveis do que um novo software.
Risco de processo reduzido	O custo do software existente já é conhecido. Tal fator é muito importante para o gerenciamento do projeto, pois reduz a margem de erro na estimativa de custos.
Uso eficaz de especialistas	Envolve o desenvolvimento de softwares reusáveis que encapsulem o conhecimento de especialistas.
Conformidade com padrões	Quando nota-se um padrão utilizado em diversas aplicações, como menus em uma interface de usuário, ele pode ser implementado como um conjunto de componentes reusáveis.
Desenvolvimento acelerado	Entregar o sistema mais rápido possível, é de extrema importância. O reúso de software pode acelerar a produção do sistema, pois pode reduzir o tempo de desenvolvimento e validação.

Fonte: Adaptada de Sommerville (2011).

Ao longo dos últimos vinte anos, muitas técnicas foram desenvolvidas para oferecer suporte ao reúso de software, explorando características comuns e seus potenciais para reúso em vários níveis de aplicação, desde funções simples até aplicações completas. Existe uma grande variedade de técnicas de reúso de software, algumas delas são mostradas na Figura 2. No entanto, a decisão de reusar é um fator mais gerencial do que técnico, o que pode desmotivar o reúso, uma vez que os riscos associados podem ser desconhecidos em relação ao desenvolvimento de um novo software.

A seguir, são listados alguns fatores que devem ser considerados ao planejar o reúso:

1. Cronograma de desenvolvimento para o software: caso seja necessário um rápido desenvolvimento, deve-se tentar reusar sistemas de prateleira em vez de componentes individuais.
2. A expectativa de duração do software: se estiver desenvolvendo um sistema de vida longa, a manutenção deve ser considerada como parte primordial pois trará implicações a longo prazo.

Tabela 2 – Problemas do Reúso de Software

Problema	Explicação
Maiores custos de manutenção	Se o código fonte não estiver disponível, os custos de manutenção podem ser maiores, pois os elementos reusados podem tornar-se cada vez mais incompatíveis com as alterações do sistema.
Falta de ferramentas de suporte	Algumas ferramentas de software não apoiam o desenvolvimento com reúso. Pode ser difícil ou impossível integrar essas ferramentas com um sistema de biblioteca de componentes.
Criação, manutenção e uso de uma biblioteca de componentes	Preencher uma biblioteca de componentes reusáveis e garantir que desenvolvedores de software possam utilizar essa biblioteca podem ser ações caras. Processos de desenvolvimento precisam ser adaptados para garantir que a biblioteca seja usada.
Encontrar, compreender e adaptar os componentes reusáveis	Componentes de software precisam ser descobertos em uma biblioteca, compreendidos e, às vezes, adaptados para trabalhar em um novo ambiente. Os engenheiros precisam estar confiantes de que encontrarão, na biblioteca, um componente, antes de incluírem a pesquisa de componente como parte de seu processo normal de desenvolvimento.

Fonte: Adaptada de [Sommerville \(2011\)](#).

Figura 2 – Técnicas de Reúso de Software

Fonte: Adaptada de [Sommerville \(2011\)](#).

3. Expertise do desenvolvimento da equipe: diante da variedade de técnicas de reúso de software é necessário muito tempo para compreendê-las e usá-las de forma eficaz. Portanto, se a equipe tem conhecimento aprofundado em uma técnica específica de reúso, provavelmente essa é a técnica em que deve-se concentrar os esforços.
4. O domínio da aplicação: em alguns domínios de aplicação existem diversos produtos genéricos que podem ser reusados e podem ser configurados para uma situação específica.
5. A plataforma em que o sistema será executado: alguns ativos podem ser específicos de plataforma e o seu reúso apenas será possível se utilizar plataformas compatíveis.

A prática de reusar pode trazer vários benefícios mas, conforme citado na Tabela 2, existem diversos problemas. Sendo assim, implantar a prática de reúso pode ser trabalhosa, visto que é necessário estabelecer uma maneira efetiva para realizar a busca, compreender e adaptar os itens reusáveis à aplicação (SOMMERVILLE, 2011). Dentre as técnicas de reúso ilustradas na Figura 2, neste trabalho utiliza-se o desenvolvimento orientado a serviços, por isso, na próxima seção a Arquitetura Orientada a Serviços é apresentada com mais detalhes.

2.2.1 *Arquitetura Orientada a Serviços - SOA*

A Arquitetura Orientada a Serviços - SOA (do inglês, *Service Oriented Architecture*) é utilizada para integração de sistemas distribuídos que podem estar em vários domínios e visa prover diversas vantagens quanto ao reúso, produtividade e evolução de sistemas (PAPAZOGLU; HEUVEL, 2007). Esse estilo arquitetural permite desenvolver os sistemas com independência de linguagem de programação e flexibilidade quanto à plataforma, uma vez que funcionalidades oferecidas por serviços são definidas por meio de linguagens de descrição padronizadas. Nesta seção, são discutidos os conceitos referentes à SOA e sua aplicabilidade, uma vez que estão relacionados a este trabalho de mestrado.

Basicamente, SOA é um estilo arquitetural de construção e integração de software que estrutura aplicações em serviços, que são elementos computacionais que tem o propósito de desempenhar uma função específica e que podem ser utilizados por um cliente. Esta arquitetura é útil em organizações que utilizam variados tipos de sistemas, com diferentes tecnologias e fornecedores, sendo imprescindível o atendimento a requisitos como escalabilidade, flexibilidade e interoperabilidade. Além disso, SOA é uma solução interessante para lidar com sistemas legados que, geralmente, apresentam pouca adaptabilidade às tendências tecnológicas. Isso porque pode-se encapsular funções realizadas por esses sistemas em novos serviços que podem ser criados para realizar funções específicas. Dessa forma, SOA utiliza o conceito de composição de aplicações, por meio da descoberta de interfaces invocáveis para a realização de tarefas mais completas (PAPAZOGLU; HEUVEL, 2007).

A Arquitetura Orientada a Serviços fornece uma estrutura flexível que unifica processos de negócios ao modelar as aplicações em serviços. Um cliente a partir de qualquer dispositivo, usando qualquer sistema operacional, em qualquer linguagem de programação, pode acessar um serviço SOA para criar um novo processo de negócio. Em outras palavras, SOA cria um conjunto de serviços que podem se comunicar uns com os outros usando interfaces de serviços para passar mensagens de um serviço para outro, ou por meio de um coordenador de atividades entre um ou mais serviços (PAPAZOGLU; HEUVEL, 2007).

Serviços são módulos independentes, bem definidos e auto-contidos que fornecem funcionalidades de negócio e são independentes do estado ou contexto de outros serviços, ocultando detalhes de implementação e expondo apenas informações relativas às suas interfaces (ERL, 2005). Descrições de serviços são disponibilizadas com detalhamentos formais que estabelecem regras para a garantia de interoperabilidade entre os serviços, podendo assumir diferentes papéis de acordo com o contexto no qual eles são empregados.

De acordo com Papazoglou e Heuvel (2007), serviços podem desempenhar dois papéis diferentes: consumidor e provedor. Quando provedor, expõe sua interface, publicando sua descrição para ser posteriormente invocado. Quando consumidor, ele utiliza as funcionalidades disponibilizadas pelo provedor de acordo com suas necessidades de negócio. Também é possível uma abordagem alternativa, denominado serviço agregador, que provê funcionalidades compostas desempenhando o papel de consumidor e provedor, conforme ilustrado na Figura 3.

Figura 3 – Papéis Desempenhados por um Serviço Agregador

Fonte: Adaptada de Papazoglou e Heuvel (2007).

A comunicação entre serviços pode ser feita de duas maneiras: direta ou por meio de serviços intermediários. Na direta, o consumidor possui conhecimento prévio do endereço do serviço que deseja utilizar, enquanto que por meio de intermediários, o consumidor informa a um registro sobre qual funcionalidade deseja utilizar. Consecutivamente, o registro consulta um conjunto de serviços nele publicados, informando ao consumidor o endereço de um serviço que

atenda às especificações requeridas.

Serviços compostos podem ser obtidos por meio da agregação de informações sobre diversos serviços, fornecendo funcionalidades mais completas, construídas a partir de funcionalidades oferecidas por serviços mais simples. De acordo com [Kazhamiakin, Pistore e Santuari \(2006\)](#), serviços compostos viabilizam o reaproveitamento de componentes, acelerando o desenvolvimento de aplicações. Pode-se descrever os modelos de orquestração e coreografia para a composição de serviços.

Figura 4 – Exemplo de Orquestração de Serviço

Fonte: Adaptada de [Peltz \(2003\)](#).

A orquestração é uma abordagem baseada no modelo de fluxo de trabalho, contendo nós e o fluxo de dados entre eles. Um processo central controla os serviços envolvidos e coordena a execução das diferentes operações. Na Figura 4, é apresentado um exemplo de orquestração na qual o coordenador central (orquestrador) recebe uma solicitação do Serviço 1. Em seguida, ele invoca o Serviço 2. Ao receber esta resposta, ele invoca outros serviços caso seja necessário. Ao concluir o processamento, o controlador central retorna uma resposta para o Serviço 1, portanto deixa-se a cargo do orquestrador central a invocação e retorno das mensagens. O modelo de orquestração apresenta benefícios, tais como: a facilidade de implementação, administração e monitoramento centralizado. Na orquestração, o processo é sempre controlado a partir do ponto de vista do coordenador central ([ERL, 2005](#)).

Ao contrário da orquestração, na coreografia, os serviços conhecem uns aos outros e cada um conhece sua função dentro do fluxo de trabalho. A coreografia é realizada pela troca de mensagens entre os serviços, sem a presença de um coordenador central. Na Figura 5, é ilustrado como funciona. Cada serviço realiza uma atividade e inicia um ou mais serviços que realizam outras atividades.

Comparando coreografia em relação à orquestração, existe maior dificuldade para implementação, administração e monitoramento de coreografias. Problemas decorrentes da execução

Figura 5 – Exemplo de Coreografia

Fonte: Adaptada de [Peltz \(2003\)](#).

são difíceis de gerenciar e a localização dinâmica de um serviço não é fácil de ser obtida.

2.2.2 Ativos Reusáveis e RAS

Dentre os problemas do reúso de software, pode-se dizer que encontrar, compreender e adaptar ativos reusáveis é uma tarefa árdua. Na maioria dos casos eles estão em diversos formatos, com diferentes padronizações, espalhados por toda parte e sem nenhuma forma de integração entre eles. O intuito deste trabalho de mestrado está na interoperabilidade entre diversos ambientes de reúso, de forma que eles estejam aptos a disponibilizar, de forma padronizada, ativos reusáveis. Para isso, é necessário estabelecer o que se entende por ativos reusáveis, assim como, sua estrutura e descrições. Nas seções a seguir, detalham-se os conceitos básicos sobre ativos reusáveis e RAS. A AIRES, adotada como parte fundamental deste trabalho, também utiliza e segue as diretrizes da RAS. Na Seção 2.2.4, são apresentados mais detalhes sobre a AIRES.

2.2.2.1 Ativos Reusáveis

Ativos de software reusáveis (do inglês *reusable software assets*) fornecem uma solução para um problema em um determinado contexto. Eles são compostos de uma coleção de produtos de software que podem ser reusados em diferentes aplicações ([EZRAN; MORISIO; TULLY, 2002](#)). Em outras palavras, um ativo é um conjunto de informações e artefatos que contém ao menos uma solução para um problema específico ([OMG, 2005](#)). Os ativos podem abranger quaisquer artefatos reusáveis, tais como código fonte, componentes, serviços, *frameworks*, padrões de projeto, interfaces de usuários, documentos de requisitos e casos de testes.

Os ativos relacionados a este trabalho são aqueles desenvolvidos e projetados para serem reusados em desenvolvimento de software. Na Figura 6, ilustra-se um ativo com três artefatos e um ponto de variabilidade para um determinado contexto, com regras para seu uso e direcionado

a resolver um problema. Por meio dessa figura, pode-se perceber que os ativos se tratam de algo abstrato e o artefato é o que realmente será reusado.

Figura 6 – Definição Geral do Ativo

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

2.2.2.2 Reusable Asset Specification - RAS

A *Reusable Asset Specification* - RAS é um conjunto de diretrizes e recomendações sobre a estrutura, o conteúdo e as descrições de ativos reusáveis de software ([OMG, 2005](#)). A RAS foi desenvolvida pelo *Object Management Group* - OMG em parceria com grandes empresas de software, tais como IBM e *Blueprint Technologies*. Tem como objetivo a padronização consistente do empacotamento de ativos, para reduzir o conflito das transações na reutilização de software. Desenvolvido em diagramas UML, essa padronização identifica categorias para ativos e *Profiles* para os ativos, que encapsulam o conhecimento do negócio em diversos níveis de abstração ([OMG, 2005](#)).

A RAS é dividida em duas categorias principais: o Core RAS e seus perfis (*Profiles*). O Core RAS representa os elementos fundamentais e os *Profiles* descrevem extensões para esses elementos essenciais, nunca alterando a definição ou a semântica do Core RAS, que não pode ser instanciado. As principais partes do Core RAS são ilustradas na Figura 7. Um ativo contém uma descrição, o estado, a versão e o Profile utilizado, junto das categorias que representam sua Classificação, sua Solução com artefatos, seu Uso com instruções e atividades e por último seus Ativos Relacionados.

Na Figura 8, é apresentado o modelo de domínio do Core RAS e suas principais categorias. A RAS especifica os modelos dos ativos para o Core RAS e seus Profiles. Fornece detalhes de suas classes e atributos, definindo se são obrigatórios e se há restrições semânticas que os diagramas não podem descrever.

Figura 7 – Principais Seções do Core RAS

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

Figura 8 – Principais Seções do Modelo de Domínio do Core RAS

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

Na Figura 9, é mostrado o metamodelo da RAS. As partes principais desse modelo foram incorporadas ao modelo conceitual deste trabalho de mestrado, com algumas adaptações. Com o uso da RAS, espera-se facilitar a interoperabilidade com outras ferramentas que a utilizam, além

de aproveitar o corpo de conhecimento embutido na própria especificação.

Figura 9 – Metamodelo da RAS

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

2.2.3 Ambientes de Reúso de Software

Ambientes de reúso de software fornecem aos usuários um conjunto de funcionalidades que apoiam a atividade de reúso ([MASCENA et al., 2006](#)). O conhecimento disponível, na maioria dos casos, é desestruturado e espalhado, sendo assim, um ambiente de reúso pode potencializar a atividade de reúso por conter diversas ferramentas integradas que podem interoperar e prover maior eficácia no reúso de diversos ativos, durante o desenvolvimento de software.

Atualmente, vários ambientes de desenvolvimento de software incluem, de alguma forma, funcionalidades relacionadas ao reúso. Por exemplo, o Eclipse fornece *plugins* para permitir o reúso de código. Em [Sawadsky e Murphy \(2011\)](#), foi proposto um protótipo de *plugin*, denominado *Fishtail*, que auxilia o programador na descoberta de exemplos de código fonte

e documentação relevantes para a tarefa atual a partir da web. *Fishtail* utiliza um histórico de interações do programador com o código fonte de um programa e artefatos relacionados para conduzir a busca automatizada de exemplos. Com este *plugin*, alivia-se o programador de comutar do IDE para uma janela do navegador e manualmente digitar palavras chave para uma pesquisa. Em vez disso, o programador vê, em uma janela dentro da IDE, os exemplos de códigos relacionados que são periodicamente atualizados.

O ambiente OdysseyShare (WERNER *et al.*, 2003b) propõe um ambiente de desenvolvimento de software colaborativo baseado em componentes e permite o reúso de modelos conceituais, arquiteturas de software e modelos de implementação para um domínio específico. Ele apoia estas atividades fornecendo um conjunto de componentes reusáveis que estão disponíveis em pacotes que incluem como por exemplo: diagramas, código fonte, descrições textuais e código binário.

A ferramenta Peônia (CHAN, 2008; CHAN *et al.*, 2007; BRAGA; CHAN, 2008) permite o reúso de processos, padrões de software e requisitos de teste associados. Ela fornece um repositório de padrões de software associados a cada fase ou atividade do processo. Cada padrão possui requisitos de teste para validar sua utilização. O engenheiro de software pode criar projetos em que ele escolhe o processo de desenvolvimento a ser adotado e recebe orientação durante a aplicação do processo, que pode ser adaptado a interesses específicos.

Sistemas de controle de versões também potencializam a atividade de reúso, visto que facilitam a recuperação de ativos para serem utilizados em outros projetos. Eles podem ser utilizados, por exemplo, durante o processo de desenvolvimento de software para controlar os diferentes estágios e versões do desenvolvimento de código fonte. Várias versões são armazenadas no sistema de controle de versão, de modo que qualquer versão desejada possa ser recuperada e reusada por diferentes equipes de desenvolvimento, podendo estar em diferentes posições geográficas (SPINELLIS, 2005).

Em (MAHMOOD; AHMED; ALSHAYEB, 2013), foi feito um estudo sobre ambientes de reúso disponíveis na literatura. Nesse estudo, foram detalhados nove ambientes de reúso, em que três deles se destacam: AsmiaÅ,ek *et al.* (2010) definiram um processo, linguagem e ferramenta de reúso na qual o reúso de software é baseado em construir e recuperar artefatos de software que podem ser reusados por outros sistemas em domínios diferentes. Rubin e Chechik (2012) apresentaram um modelo de dados e uma técnica de representação para transformar produtos individuais em representações de linha de produtos de software. Robinson e Woo (2004) apresentaram um ambiente de reúso denominado REUSER que auxilia a recuperar artefatos semelhantes para reutilização. REUSER utiliza técnica de agrupamento baseada em grafos para ajudar os analistas a reusar diagramas de sequência UML (do inglês *Unified Modeling Language*). Com base na análise desses ambientes de reúso, é possível observar que há necessidade de oferecer suporte à integração, ou seja, dos nove ambientes citados nenhum oferece apoio à integração com ferramentas existentes, o que implica apenas no uso isolado dos ambientes.

Durante o desenvolvimento de software, é desejável querer se beneficiar de mais de uma forma de reúso, mas isso leva o desenvolvedor a lidar com vários ambientes, ferramentas, processos e técnicas diferentes, o que pode gerar desestímulo ao reúso e impor obstáculos significativos. Além disso, integrar técnicas de reúso em uma forma *ad-hoc* consome muito tempo e é propensa a erros (CHEN; CHENG, 1997).

2.2.4 AIRES

O reúso de software, apesar de seus benefícios, também apresenta diversos problemas. De acordo com (MAHMOOD; AHMED; ALSHAYEB, 2013) e (MASCENA *et al.*, 2006), um passo em direção a solução desses problemas é a obtenção de um ambiente integrado de reúso, que pode maximizar a produtividade e oferecer reúso de outras ferramentas integradas ao ambiente para permitir a descoberta e reutilização de diversos serviços.

Nesse sentido, a orientadora deste trabalho está desenvolvendo uma arquitetura integrada de reúso de software, denominada AIRES (do inglês (*ArchItecture for REuse using Services*), em que o principal foco é permitir o reúso em diversos níveis de abstração, desde processos até artefatos de código fonte (BRAGA *et al.*, 2016). As partes principais do modelo RAS foram incorporadas ao modelo conceitual da AIRES, com algumas adaptações. Com o uso da RAS, espera-se facilitar a interoperabilidade com outras ferramentas que a utilizam, além de aproveitar o corpo de conhecimento embutido na própria especificação.

De acordo com Braga *et al.* (2016), pode-se citar cinco motivações para a AIRES: (1) Importância de facilitar o desenvolvimento de software por meio de metodologias de reúso. (2) Suporte computacional para reúso durante todo o ciclo de desenvolvimento. (3) Diversidade de métodos, processos e ferramentas utilizados com um grande número de formatos de representação para processos e ativos reusáveis. (4) Flexibilidade diante da variedade de plataformas de desenvolvimento. (5) Potencial das arquiteturas orientadas a serviços para resolver alguns dos problemas mencionados.

Cada instância da AIRES disponibiliza serviços de forma pública. Isso facilita a criação de aplicações que utilizam esses serviços, além de facilitar a integração com as ferramentas existentes. Considerando que a AIRES pode ser instanciada por diferentes instituições, geograficamente distribuídas em diferentes locais e operando de forma independente, novos serviços podem ser desenvolvidos no futuro para integrar essas diferentes instâncias.

Na Figura 10 fornece-se uma visão geral da AIRES, em que se destaca o uso de uma infraestrutura baseada em serviços para permitir a integração entre seus vários módulos. A AIRES pode ser instanciada em contextos específicos, por exemplo em empresas ou instituições que queiram iniciar um programa de reúso de software. Para isso, a empresa pode adaptar suas aplicações-cliente para que se integrem à AIRES por meio da invocação de serviços disponibilizados, ou podem utilizar diretamente uma instância da AIRES disponível na Web

para uso público.

Neste trabalho de mestrado, procurou-se explorar instâncias da AIRES como sistemas que compõem o SoS, bem como poderão ser adaptadas outras ferramentas de reúso, como por exemplo gerenciadores de versões, que podem ser integrados e então formar um SoS. No entanto, como a AIRES ainda não está implementada, baseou-se apenas em seu projeto, considerando as interfaces de serviços que permitem acessar seus ativos reusáveis.

Figura 10 – Visão Geral da AIRES

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

2.3 Sistemas-de-Sistemas

2.3.1 Conceitos Básicos

Uma das principais características dos grandes sistemas é que eles podem ser formados a partir de uma variedade de sistemas constituintes e podem estar em diferentes posições geográficas, sob o controle de diversas organizações. Há um interesse crescente na obtenção de sinergia entre sistemas independentes para atingir um objetivo geral desejado (AZARNOUSH *et al.*, 2006). Com isso, SoS têm recebido notável atenção das pesquisas na literatura de Engenharia

de Software. De acordo com [Fitzgerald, Bryans e Payne \(2012\)](#), SoS são colaborações sinérgicas entre sistemas que são operacional e gerencialmente independentes, distribuídos, evoluem dinamicamente e exibem um comportamento emergente. Um SoS oferece capacidades e serviços que não podem ser alcançados nos sistemas constituintes (do inglês *Constituent Systems* - CS) de forma isolada.

Recentemente, observou-se um interesse crescente pelos conceitos e estratégias de SoS que tem se tornado o foco de vários domínios de aplicação, tais como: militares, segurança, aeroespacial, sistemas de gestão de catástrofes, entre outros ([LOPEZ, 2006](#); [WOJCIK; HOFFMAN, 2006](#)). Com base na literatura existente sobre SoS, percebe-se que há várias definições espalhadas e divulgadas, algumas delas dependem da particularidade de uma área de aplicação ([BOURQUE; FAIRLEY et al., 2014](#); [JAMSHIDI, 2008](#)). Algumas definições mais alinhadas aos objetivos deste trabalho são apresentadas a seguir.

De acordo com ([FEILER et al., 2006](#)), SoS é um sistema de grande escala que pode ser formado por sistemas que pertencem a diferentes domínios. Essa heterogeneidade tem sido apontada como uma das propriedades de um SoS. Sendo assim, para que um SoS funcione efetivamente, seus sistemas constituintes precisam trabalhar em direção a um objetivo comum. Isto significa que diferentes sistemas constituintes precisam se comunicar uns com os outros e entender suas perspectivas. A heterogeneidade e distribuição podem estar presentes em um SoS, o que faz com que a comunicação seja extremamente desafiadora. Além disto, influências externas precisam ser entendidas e levadas em consideração quando se refere ao comportamento final do SoS.

[Mayk e Madni \(2006\)](#) definem um SoS como um conjunto de sistemas que foram originalmente concebidos como sistemas autônomos para fins específicos e diferentes, mas que foram reunidos dentro de um SoS para criar uma nova capacidade necessária para um determinado objetivo.

Um SoS é usualmente descrito como uma composição de sistemas, em que os elementos são projetados separadamente, descobertos, selecionados e integrados de modo a formar um sistema maior (mais complexo), emergindo um novo comportamento ou função não disponível em qualquer sistema individual que o compõe ([MAIER, 1998](#)).

[Norman e Kuras \(2006\)](#) ressaltam alguns pontos importantes sobre os sistemas constituintes de um SoS: (1) Geralmente não compartilham uma base conceitual comum; (2) Compartilham um ambiente de aquisição que os mantém autônomos; e (3) Apresentam tecnologias facilitadoras de integração e a integração é realizada após a entrega dos sistemas constituintes.

Há uma outra definição mais abrangente sobre SoS dada por [Boardman e Sauser \(2006\)](#), [Firesmith \(2010\)](#), que é adotada como base para o desenvolvimento deste trabalho de mestrado: “SoS é uma coleção de entidades independentes e seus relacionamentos reunidos para formar um todo, maior do que a soma das partes. Então, nossa atenção é direcionada às partes (entidades),

relacionamentos (entre as partes) e o próprio todo, o qual entendemos em algum sentido ser melhor do que a coleção das partes e relacionamentos”.

Uma comunicação bem sucedida entre os CS's é essencial para o desenvolvimento do SoS. Os CS's são muitas vezes desenvolvidos fora do contexto do SoS e, como resultado, os sistemas podem ser incapazes de interagir totalmente e de se adaptarem a diversos comportamentos emergentes. Assim, é necessário estabelecer uma linguagem comum envolvendo os dados que são enviados e recebidos entre os CS's do SoS (PARISI; SAHIN; JAMSHIDI, 2008).

Ghose e DeLaurentis (2009) resumem algumas das causas comuns de falhas dentro do processo de desenvolvimento do SoS, são elas: (a) o desalinhamento dos objetivos entre os sistemas, (b) controle limitado dos sistemas que compõem o SoS, (c) evolução do SoS, (d) inflexibilidade de projeto dos sistemas constituintes, (e) comportamento emergente, em alguns casos, revela dependências ocultas dentro dos sistemas, (f) complexidade dos sistemas e por último (g) desafios na representação do sistema.

2.3.2 Características de um SoS

SoS se destacam por um conjunto de oito principais dimensões (ANDREWS *et al.*, 2013; FITZGERALD; BRYANS; PAYNE, 2012; MAIER, 1998; PÉREZ *et al.*, 2013): autonomia, independência, distribuição, evolução, comportamento emergente, comportamento dinâmico, interdependência e interoperabilidade. Dessas oito características de SoS, cinco foram destacadas como essenciais por Maier (1998): a independência operacional e a independência gerencial dos constituintes, o comportamento emergente, os processos de desenvolvimento evolutivo (evolução) e a distribuição geográfica. Além disso, arquitetura dinâmica e os aspectos de auto gerenciamento (ANDREWS *et al.*, 2013; FITZGERALD; BRYANS; PAYNE, 2012) são características inerentes e esperadas em um SoS. Portanto, quanto mais um sistema apresentar as seguintes características, mais apto ele está para ser classificado como um SoS (MADNI; SIEVERS, 2014a).

- **Autonomia:** a capacidade de completar os próprios objetivos dentro de seu limites, sem o controle de outra entidade. Esta característica implica que cada sistema constituinte do SoS pode funcionar como um sistema livre e autônomo, podendo tomar decisões individualmente (BALDWIN; SAUSER, 2009).
- **Independência:** de acordo com Maier (1998), considerando a independência operacional, os sistemas constituintes devem ser capazes de operar de forma independente e útil. O SoS é formado por sistemas que são independentes e úteis individualmente. Com relação à independência gerencial, os sistemas constituintes não somente podem operar independentemente, mas também podem ser gerenciados independentemente. Eles são adquiridos e integrados separadamente, mas mantêm sua existência operacional contínua e independente do SoS.

- **Distribuição Geográfica:** a extensão geográfica dos sistemas constituintes pode ser grande, isso significa que eles devem interoperar mesmo quando estão geograficamente distribuídos. Grande é um conceito nebuloso e relativo à medida que as capacidades de comunicação aumentam, mas, no mínimo, significa que os sistemas constituintes podem trocar facilmente informações, uma vez que eles não estão necessariamente localizadas em uma mesma posição geográfica (MAIER, 1998; FIRESMITH, 2010; NAKAGAWA *et al.*, 2013).
- **Desenvolvimento evolucionário:** o desenvolvimento e existência do SoS é evolutivo, com funções e propósitos adicionados, removidos e/ou modificados de acordo com a experiência e necessidade (MAIER, 1998).
- **Comportamento emergente:** o SoS executa funções/comportamentos que não estão presentes em nenhum outro sistema constituinte. Esses comportamentos são propriedades emergentes de todo o SoS e não podem ser localizados em qualquer sistema constituinte de forma separada (MAIER, 1998; FIRESMITH, 2010).
- **Comportamento Dinâmico:** Os sistemas constituintes podem entrar e sair do contexto de um SoS a qualquer momento, caso julgue necessário. Esta característica compreende a capacidade do SoS de mudar as relações entre os sistemas constituintes e ajustar o número de sistemas constituintes presente no SoS, o que inerentemente afeta a dinâmica da arquitetura do SoS (NIELSEN *et al.*, 2013).
- **Interdependência:** significa que há uma dependência mútua entre os sistemas constituintes que formam o SoS. Isso acontece devido ao fato de que eles precisam confiar um no outro para cumprir o objetivo comum dos SoS. Portanto, as ações dos sistemas constituintes podem afetar os outros (NIELSEN *et al.*, 2013).
- **Arquitetura Dinâmica:** arquiteturas de SoS são fundamentalmente dinâmicas, pois os sistemas constituintes são independentes. Portanto, eles podem mudar ao longo do tempo, de uma forma não prevista quando foram originalmente compostos. O SoS como um todo evolui continuamente para responder às mudanças dos objetivos, comportamentos emergentes ou mudança nos elementos internos. Como consequência, os sistemas constituintes e as conexões não são estáveis (sob uma perspectiva arquitetônica) e podem ser adicionados ou removidos a qualquer momento (BATISTA, 2013).

2.3.3 Tipos de Sistemas-de-Sistemas

Há uma grande discussão sobre a classificação dos tipos de um SoS. Na literatura é possível encontrar vários tipos, definidos por diferentes autores. No entanto, SoS tem sido usualmente diferenciado pelo nível de controle gerencial (DAHMANN; JR; LANE, 2008), por sua autoridade central e nível de colaboração entre os CS's (BRYANS *et al.*, 2013; PÉREZ

et al., 2013). Com base nessa diretriz, pode-se destacar quatro tipos essenciais: direcionado, colaborativo, reconhecido e virtual (DAHMANN; JR; LANE, 2008; MAIER, 1998).

- **Direcionado:** este tipo de SoS é composto por sistemas integrados que são construídos e gerenciados para um propósito específico. Os sistemas constituintes são gerenciados centralmente e mantêm a capacidade de operar de forma independente para cumprir um objetivo comum, mas seu modo de operação quando integrado a outros sistemas é subordinado ao propósito central. Em suma, os sistemas constituintes funcionam subordinados a um coordenador central (MAIER, 1998; PÉREZ *et al.*, 2013).
- **Colaborativo:** neste tipo de SoS os sistemas constituintes interagem relativamente de forma voluntária para cumprir um objetivo comum. Eles são colaboradores e decidem coletivamente como fornecer ou negar um determinado serviço. Um SoS colaborativo difere de um SoS direcionado, na medida em que a entidade de gerenciamento central não possui poder absoluto para tomar decisões, sendo feitos acordos entre as entidades para fornecer ou negar algum mecanismo (MAIER, 1998).
- **Reconhecido:** este tipo de SoS tem objetivos reconhecidos e uma equipe de gestão e recursos designados para o SoS. É composto por sistemas existentes, juntamente com novos desenvolvimentos. Os sistemas constituintes mantêm sua independência e seus objetivos. Mudanças nos sistemas baseiam-se na colaboração entre o SoS e os CS's (DAHMANN; JR; LANE, 2008).
- **Virtual:** este tipo de SoS não tem autoridade administrativa centralizada sobre os sistemas constituintes, o comportamento emergente surge e pode ser desejável. A interação entre os sistemas constituintes acontece livremente e diretamente entre eles. Este tipo de SoS invoca mecanismos de forma invisível para se manter, sem a presença de um coordenador central. Os CS's não são subordinados ao coordenador central e não há propósitos claros do SoS. A engenharia de requisitos é baseada em exigências menos concretas. Os sistemas devem ser construídos com flexibilidade e adaptabilidade para trabalhar com as capacidades existentes, dispondo de características como: auto-configuração e auto-organização, para alcançar uma tarefa comum (LANE, 2013; MAIER, 1998; JAMSHIDI, 2008).

Implementar e integrar diferentes tipos de sistemas se torna mais complexo em razão da natureza evolutiva de um SoS. Embora muitos sistemas possam ser dependentes de outros durante as fases de implementação ou de integração, o processo de desenvolvimento de um SoS muitas vezes não é controlado centralmente. Isto requer que os desenvolvedores dos sistemas constituintes tenham uma preocupação em colaborar uns com os outros. Estas questões são apenas um sub-conjunto dos desafios enfrentados durante o processo de desenvolvimento no contexto de SoS (MADNI; SIEVERS, 2014a). Outros problemas e desafios podem ser vistos em detalhe no Apêndice B.

2.3.4 Interoperabilidade de SoS

Interoperabilidade é a capacidade de sistemas distintos compartilhar informações semanticamente compatíveis e, em seguida, processar/gerenciar as informações, permitindo aos usuários executar tarefas desejadas (MADNI; SIEVERS, 2014b; ZEIGLER; MITTAL, 2008). Este conceito implica que é necessário fazer muito mais do que os sistemas se comunicarem uns com os outros. Interoperar exige que as organizações envolvidas empreguem algum grau de semântica compatível e interpretações comuns das informações trocadas (MADNI; SIEVERS, 2014b). Também é importante ressaltar, na maioria dos casos, que os sistemas são projetados e implementados antes que exista uma necessidade reconhecida para interoperar. Essa necessidade, quando surge, estende a interoperabilidade além do escopo original do sistema. Portanto, existem preocupações internas que precisam ser abordadas por qualquer sistema, se algum dia ele necessitar interagir com outros.

Alcançar interoperabilidade no contexto de SoS é difícil mesmo quando considerada durante o projeto, e ainda mais difícil quando ela não é levada em consideração desde o princípio. Na Tabela 3, apresentam-se os casos de interoperabilidade em ordem crescente de dificuldade (MADNI; SIEVERS, 2014a). Os dois primeiros casos são pertinentes a SoS projetados desde o início, uma vez que envolvem a concepção e implementação de um novo sistema que será projetado intencionalmente com questões de interoperabilidade. Os últimos três casos, no entanto, são mais problemáticos, uma vez que envolvem várias combinações de adaptação dos sistemas existentes para alcançar a interoperabilidade.

De acordo com Madni e Sievers (2014a), para um SoS, a interoperabilidade, muitas vezes, serve como um substituto para a integração se desenvolvida de forma independente. Esse conceito deve retratar tanto a interoperabilidade sintática quanto a interoperabilidade semântica. A interoperabilidade sintática é a capacidade de dois ou mais sistemas se comunicarem e trocarem dados. Nesse caso, formatos de dados específicos e protocolos de comunicação são fundamentais. Padrões, como *Extensible Markup Language* (XML) e *Structured Query Language* (SQL) apoiam a interoperabilidade sintática. A interoperabilidade sintática é um pré-requisito para a interoperabilidade semântica, que é a capacidade de dois ou mais sistemas interpretar automaticamente o significado das informações trocadas e produzir os resultados úteis para os usuários finais (de ambos os sistemas). Nesse caso, quaisquer dois sistemas dentro de um SoS devem chegar a um acordo sobre um modelo comum de troca de informações.

Examinar e planejar a interoperabilidade é complexo e algumas vezes difícil de ser alcançada pelos seguintes motivos (HAN; CHEN, 2002):

- Falta de maturidade, no nível de projeto, para reconhecer os níveis crescentes de sofisticação em relação à troca de informações entre os sistemas.
- Falta de entendimento dos atributos de interoperabilidade dos sistemas.

Tabela 3 – Casos de Interoperabilidade no Contexto de SoS

ID	Casos de Interoperabilidade no Contexto de SoS
1	<p>Novos sistemas são desenvolvidos como parte de um novo SoS intencionalmente projetado:</p> <ul style="list-style-type: none"> • Coordenador central deve reforçar o plano de integração do SoS. • O desenvolvimento de cada sistema deve ser forçado a aderir aos guias de interoperabilidade recomendados pelo coordenador central.
2	<p>Novos sistemas são intencionalmente projetados para trabalhar em um SoS existente:</p> <ul style="list-style-type: none"> • Forçar os sistemas a aderir quaisquer que sejam os guias de interoperabilidade do SoS.
3	<p>Novos sistemas são desenvolvidos para ser parte de um SoS, que não foi intencionalmente projetado como um SoS:</p> <ul style="list-style-type: none"> • Podem não haver guias de interoperabilidade. • Novos guias podem ser desenvolvidos pelo coordenador central ou via colaboração com desenvolvedores e usuários de outros sistemas do SoS.
4	<p>Um sistema existente não projetado para ser parte de um SoS, é subsequentemente requisitado para interoperar com outros sistemas em um SoS <i>ad-hoc</i>:</p> <ul style="list-style-type: none"> • O sistema existente deve ser adaptado conforme os guias de interoperabilidade.
5	<p>Um sistema existente que não é parte de um SoS é requerido para interoperar com outros sistemas em um SoS <i>ad-hoc</i>:</p> <ul style="list-style-type: none"> • Não somente o SoS não foi projetado intencionalmente como um SoS, mas nenhum esforço de desenvolvimento de sistema está em andamento para incorporar interoperabilidade como novo conceito. • Sistemas existentes devem ser adaptados para interoperar, com mínima modificação. • Sistemas existentes não foram projetados como parte de um SoS. • No mundo real sistemas existentes são chamados a interoperar em formas que nunca foram vislumbradas por seus gerentes de projeto.

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

- Dificuldade em identificar um conjunto de capacidades necessárias e associá-las a opções de implementação para permitir que cada nível de interoperabilidade seja alcançado.
- Falta de suporte adequado para avaliar a interoperabilidade de um dado sistema ou par de sistemas.

Apesar das dificuldades, a interoperabilidade também oferece vantagens, incluindo: (1) maior flexibilidade, permitindo a heterogeneidade de sistemas, (2) criação de novas capacidades, desenvolvendo novas funções a partir das existentes e (3) aumento do custo-benefício, permitindo o reúso de sistemas e capacidades existentes. A interoperabilidade é normalmente realizada por

meio de uma interface de usuário comum, com semânticas, políticas e procedimentos uniformes. Apesar das vantagens, a interoperabilidade também pode gerar desvantagens decorrentes do aumento da complexidade técnica e do projeto de sistema (ROTHENBERG, 2008).

Ao verificar a interoperabilidade dos sistemas, todos os aspectos da interface arquitetural precisam ser examinados. A verificação de compatibilidade garante que os objetivos de um sistema coincidam com os objetivos de outro sistema constituinte e vice-versa. As interfaces dos sistemas constituintes, em algum ponto em comum, devem corresponder entre si, de forma que os recursos fornecidos satisfaçam às capacidades exigidas (MADNI; SIEVERS, 2014a).

2.3.5 Integração de Sistemas-de-Sistemas

A Engenharia de Sistema de Sistemas (do inglês *System-of-Systems Engineering - SoSE*), é um campo de pesquisa que insere uma gama de desafios para os pesquisadores de Engenharia de Sistemas ou de Software, que lidam com sistemas complexos de larga escala. A Integração de Sistema de Sistemas (do inglês *System-of-Systems Integration- SoSI*) é um desafio pontual para SoSE e está presente em muitas tarefas requeridas pelos profissionais da área. De acordo com o guia *System engineering guide for System-of-Systems Engineering* (DOD, 2008), integração é o processo de incorporar elementos de baixo nível do sistema a elementos de alto nível do sistema.

A interoperabilidade é um processo intrínseco na integração, pois implica que cada sistema pode se comunicar e interagir com o SoS. Isso significa que, quando estão integrados, eles precisam ter a capacidade de se comunicar com o SoS ou com uma parte do SoS sem problemas de compatibilidade, como sistemas operacionais, hardware de comunicação, entre outros. Para esse efeito, um SoS necessita de uma linguagem comum com os seus sistemas constituintes. A integração também implica que os sistemas precisam entender uns aos outros (PARISI; SAHIN; JAMSHIDI, 2008; MADNI; SIEVERS, 2014a).

Existem algumas situações fundamentalmente distintas na qual necessita-se integrar. De acordo com Mazzuchi e Albakri (2011), são três casos de sistemas que precisam ser integrados: (1) Novos sistemas com novos sistemas. (2) Sistemas legados com novos sistemas. (3) Sistemas legados com sistemas legados.

Embora relacionadas, essas situações podem trazer diferentes preocupações. Em particular, a integração no contexto de SoS é muito mais ampla do que apenas a troca de informações. Isso pode acontecer em níveis diferentes, como a camada de banco de dados (por exemplo, dados distribuídos), a interface do usuário (por exemplo, interface gráfica compartilhada), a lógica de negócios (por exemplo, a funcionalidade de tomada de decisões) (UNDERSTANDING..., 2013).

Estudos na literatura existente indicam que um dos fatores que contribuem para a complexidade de integração dos sistemas é a contínua mudança de elementos software ou de hardware, tais como atualizações e funcionalidades adicionais. Como resultado, os sistemas

devem ser compatíveis com os produtos existentes e compatíveis com a tecnologia emergente (JOVEL; JAIN, 2009). Conforme mencionado anteriormente, outro fator que contribui para o aumento da complexidade é a integração de sistemas legados. A documentação para esses sistemas nem sempre é detalhada, coerente, ou mesmo completa. Uma documentação adequada é necessária para entender a tecnologia existente, e consecutivamente para cooperar com a integração. Muitos engenheiros que trabalham em sistemas legados migram para novos projetos em diferentes empresas, ou até na mesma empresa, e não deixam o seu conhecimento em forma de documentação (JAIN *et al.*, 2008).

Em Jovel e Jain (2009), é possível observar alguns problemas de integração e suas possíveis soluções. Foi conduzido um *survey* com cinco engenheiros que avaliaram o nível do esforço de integração. Na Tabela 4 é apresentado o nível de esforço de integração, em que quanto maior o nível de esforço, maior a complexidade de integração.

Tabela 4 – Nível de Esforço de Integração

Questão	Possível Solução	Fase	Nível de Esforço(1-10)
Falta de foco, nas futuras capacidades do sistema, durante a fase de desenvolvimento	Necessidade de líderes do projeto direcionarem o foco para as potenciais capacidades	Requisitos	8
Interoperabilidade dos sistemas existentes com a nova tecnologia integrada, causada por constantes mudanças	O projeto do sistema precisa ser compatível com todas as fases do projeto	Projeto da Arquitetura	10
Compatibilidade do Sistema Fornecedor	Compartilhamento de informação dos sistemas Fornecedores	Projeto da Arquitetura	1
Mudanças contínuas no sistema	Documentação Sincronizada	Integração	7.5
Melhorias do sistema causam problemas imprevisíveis em outros subsistemas	Tipo da arquitetura do sistema faz as melhorias serem fáceis de serem adicionadas	Projeto da Arquitetura	3
Novos Engenheiros de Software não estão familiarizados com os sistemas legados	Documentação completa dos sistemas legados	Integração	10

Fonte: Adaptada de Jovel e Jain (2009).

Observa-se que a integração de sistemas legados e contínuas mudanças de projeto (novas tecnologias) são os fatores que requerem maior esforço de integração. De acordo com a opinião dos especialistas envolvidos no *survey*, esses fatores exigem esforço máximo da equipe de desenvolvimento, pois receberam nota 10 no nível de esforço.

2.3.6 Aplicação de SOA no Contexto de SoS

Por meio de SOA, pode-se criar SoS mapeando os sistemas existentes em serviços, e posteriormente, orquestrando a comunicação entre os serviços. O comportamento emergente, característica intrínseca de um SoS, pode ser alcançado por qualquer adição de novos serviços ou pela modificação da comunicação entre os serviços existentes. Em razão dessas particularidades, os projetos de SOA são particularmente propícios ao desenvolvimento de SoS (MADNI; SIEVERS, 2014a).

Alcançar todo o potencial de uma abordagem orientada a serviços requer muito mais do que simplesmente dominar a tecnologia de *Web Services*. Sem dúvida, o movimento em direção a *Web Services* ajudou a resolver muitos desafios e tentativas anteriores de implementar *middleware* de integração (LINTHICUM, 2003). Ao mesmo tempo, as principais preocupações de integração do sistema não desapareceram. Pelo contrário, a maioria dos projetos de integração apresentam desafios que vão desde preocupações de negócios a exigências de infra-estrutura. Compor um SoS a partir de sistemas constituintes reusáveis ajuda o SoS resultante a ser mais barato, confiável, flexível e escalável. Reutilização e interoperabilidade são estratégias convergentes em SoS, no qual o comportamento e as interfaces dos sistemas constituintes são definidos formalmente, permitindo o reúso e interação uns com os outros (MADNI; SIEVERS, 2014a).

Em geral, a arquitetura de um SoS pode aumentar ou inibir a capacidade dos sistemas constituintes de interagirem, ou seja, algumas arquiteturas promovem a interoperabilidade, enquanto outras não. SOA é um exemplo de arquitetura que promove a interoperabilidade entre os seus sistemas constituintes, desde que estejam em conformidade com seus paradigmas e princípios. Portanto, SOA é tipicamente projetada como interoperável e tem surgido como adequada para implementar/projetar SoS, pois os sistemas constituintes, quando representados por serviços, apresentam mais flexibilidade, adaptabilidade e conseqüentemente melhor interoperabilidade (MADNI; SIEVERS, 2014a).

2.4 Considerações Finais

Neste capítulo foi apresentada uma introdução dos conceitos básicos e terminologias relacionados a uma das principais áreas que este trabalho de mestrado envolve, o Reúso de Software. Foram contextualizados alguns ambientes e a área reúso de software, evidenciando-se os elementos principais para o desenvolvimento deste trabalho. Também foi apresentada uma introdução aos principais conceitos e terminologias relacionados a SoS. O desenvolvimento deste capítulo objetivou-se a contextualizar os campos de reúso de software e SoS, assim como suas definições, tipos, terminologia e aplicabilidade, de forma a evidenciar os elementos de estudo principais deste trabalho. No Capítulo seguinte são apresentados um Mapeamento Sistemático (MS) e uma Revisão Sistemática (RS) da Literatura que fundamentam este trabalho de mestrado.

MAPEAMENTO E REVISÃO SISTEMÁTICA DA LITERATURA

3.1 Considerações Iniciais

Com o intuito de fundamentar este trabalho de mestrado, desenvolveram-se dois trabalhos de revisão sobre o estado da arte em integração de ambientes de reúso de software e abordagens de integração em SoSI. Na Seção 3.2 são sinalizados os resultados do MS. Na Seção 3.3 são apresentados os resultados obtidos na RS. Na Seção 3.4 são apresentados alguns trabalhos relacionados encontrados durante a realização desses dois estudos da literatura e, por fim, na Seção 3.5 são apresentadas as considerações finais do capítulo.

3.2 Integração de Ambientes de Reúso de Software - Um Mapeamento Sistemático

Para auxiliar a condução deste trabalho de mestrado, realizou-se um MS, apresentado em detalhes no Apêndice A. O objetivo foi investigar melhor como ambientes de reúso de software tem sido integrados, e quais soluções têm sido propostas para alcançar a integração entre eles. Por meio do MS, evidenciou-se que há necessidade de integrar ambientes de reúso de software, pois muitos estudos apresentaram essa lacuna de integração. Conforme mencionado na Seção 2.2, com o passar dos anos várias técnicas de reúso de software foram desenvolvidas; porém, a falta de uma integração efetiva entre elas impõe obstáculos para alcançar um reúso efetivo. Nas Seções a seguir são apresentados as strings de pesquisa e os resultados obtidos. A definição do protocolo (planejamento), condução da revisão sistemática e sumarização dos dados podem ser vistos, em detalhes, no Apêndice A.

3.2.1 String de Pesquisa e Máquinas de Busca

Com base nos objetivos do MS, definiram-se as seguintes questões de pesquisas:

1. RQ1. Como tem sido investigada a integração entre ambientes de reuso de software?

Investigar se ferramentas de Engenharia de Software, especificamente ambientes de reuso, tem sido integradas e evidenciar como essa integração tem sido alcançada.

2. RQ2. SoS tem sido apontado como uma possível solução para integrar ambientes de reuso?

Evidenciar se o conceito de SoS tem sido proposto como uma possível solução para alcançar a integração entre ambientes de reuso de software

As questões de pesquisa foram utilizadas para construção da *string* de busca. A partir delas, extríram-se termos relevantes que correspondem aos três tópicos de interesse deste mapeamento:

1. *Software Reuse*: utilizou-se o termo reusable component, suas derivações e sinônimos, tais como: reusable artifact, reusable assets, reusable items e reusable code.
2. *Tool support*: utilizou-se o termo: support, suas derivações e sinônimos, tais como: environment e tool.
3. *Integration*: utilizou-se o termo: integration, suas derivações e sinônimos, tais como: integrated e composition.

Operadores *booleanos* **AND** e **OR** foram utilizados para a construção da *string* de busca, que é descrita de maneira geral a seguir: (sofre pequenas alterações para se adequar às restrições de cada base de busca indexada, não perdendo a lógica estrutural).

String de busca:((*integration OR integrated OR composition*) AND (*support OR environment OR tool*) AND ({*reusable component*} OR {*software reuse*} OR {*reusable artifact*} OR {*reusable assets*} OR {*reusable item*} OR {*reusable code*})).

Neste estudo foram consideradas bases de busca relevantes para a área de Ciência da Computação e Engenharia de Software, que indexam as principais conferências e periódicos dessas áreas, são elas: *ACM Digital library*, *IEEExplore*, *Scopus*, *Springer*, *Science Direct* e *Web of Science*.

3.2.2 Resultados do MS

Incluíram-se 47 estudos primários de um total de 1224 retornados pelas bases de dados. Na Figura 11, ilustra-se a quantidade de artigos obtidos quando executou-se a *string* de busca nas seis máquinas detalhadas anteriormente.

Figura 11 – Processo de Seleção de Estudos Relevantes para o Mapeamento Sistemático

Fonte: Dados da pesquisa.

Encontraram-se 330 artigos repetidos, ou seja, artigos que foram indexados por mais de uma máquina de busca. Ao avaliar os 894 artigos restantes após a exclusão dos repetidos, por meio da leitura do Título e Resumo, selecionaram-se 47 artigos para leitura transversal.

Na etapa final do processo de seleção dos estudos relevantes, realizou-se a leitura transversal dos 47 artigos e selecionaram-se 28 artigos para extração de dados. Na Figura 12 é apresentada a relação da quantidade de artigos publicados por ano e o tipo de publicação

(conferência ou *journal*) dos 28 artigos finais. A seguir, são apresentados os resultados para as questões que compuseram o questionário do MS.

No Apêndice A, na Tabela 15 são apresentados os 28 artigos selecionados para extração de dados com seus respectivos anos de publicação. Classificaram-se os artigos como S1, S2, e assim sucessivamente até o S28, para referenciar de forma abreviada um determinado artigo.

Figura 12 – Relação da Quantidade de Artigos Publicados por Ano e Tipo de Publicação do Mapeamento Sistemático

Fonte: Dados da pesquisa.

3.2.3 Resposta obtida para a subquestão FE 1- Existem definições de ambientes de reúso de software?

Nessa questão objetivou-se evidenciar definições sobre ambiente de reúso de software. Dentre os 28 artigos selecionados para extração de dados, não foram encontradas definições que respondem diretamente a esta pergunta. Contudo, conforme citado em Mascena *et al.* (2006), pôde-se coletar indícios de que são ambientes que fornecem aos usuários um conjunto de funcionalidades que apoiam a atividade de reúso. O conhecimento disponível na maioria dos casos é desestruturado e espalhado, sendo assim, um ambiente de reúso pode potencializar a atividade de reúso.

3.2.4 Resposta obtida para a subquestão FE 2- Como tem sido feita a integração entre ferramentas de reúso?

Nessa questão objetivou-se caracterizar, como e quais aspectos referentes à Engenharia de Software tem sido utilizados na integração de ferramentas de reúso e quando possível exemplificar a necessidade de integração.

Em **S28** ([BASSO; WERNER; OLIVEIRA, 2014](#)), utilizou-se a RAS para especificar os *assets* reusáveis e permitir a representação de seus elementos como tarefas e funções que instruem os usuários finais a adaptar os *assets* para uma necessidade específica. Após a especificação em RAS é possível descobrir inconsistências de requisitos e alcançar uma integração de *assets* reusáveis para um propósito específico.

Em **S12** ([THIMMEGOWDA et al., 2014](#)), afirma-se que o primeiro passo na integração de duas ferramentas é definir os conceitos de CDD (do inglês *Concern-Driven Development*) em um metamodelo, guiado por quatro passos: (1) *Concern*: representa o domínio de interesse do modelo; (2) Interface do *Concern*: a interface de variação, a interface de personalização e a interface de uso; (3) Reúso do *Concern*: um conceito que é usado para armazenar os recursos selecionados e (4) Características, com as associações para conectar os modelos que implementam a característica.

Em **S17** ([PONNALAGU; NARENDRA; GANGADHARAN, 2012](#)), foi proposto um processo de busca por potenciais *assets* reusáveis integrados em um repositório, a fim de minimizar o custo e o esforço por construir soluções baseadas em SOA, em vez de se desenvolver a partir do zero. Vale observar que nesse caso não se trata da integração propriamente dita, pois trabalha-se com um repositório integrado de *assets* reusáveis.

Em **S2** ([LÓPEZ et al., 2009](#)), foi proposta a criação de componentes para acessar fontes heterogêneas, um componente para combinar dados de diferentes fontes e outro componente para a construção de uma interface gráfica. O usuário constrói graficamente a aplicação *mashup*, selecionando, personalizando, e interligando componentes.

Em **S1** ([ROSELLÓ et al., 2007](#)), foi proposta uma abordagem de integração de componentes com uma interface simples e fácil de usar orientada a objetos, que dá acesso a toda a funcionalidade do Matlab, que podem ser integrados a ferramentas de desenvolvimento, permitindo projetar seus próprios aplicativos de forma simples e intuitiva. Isso significa que o usuário não precisa nem usar Matlab nem aprendê-lo, e os desenvolvedores não necessitam usar diretamente o Matlab para o desenvolvimento de software.

Em **S24 e S16** ([HSIEH; TEMPERO, 2006](#); [SAWADSKY; MURPHY, 2011](#)), foi proposto suporte ao reúso de código integrando *plugin* de desenvolvimento ao IDE Eclipse.

Em **S4** ([CHEN; XIE; SHI, 2005](#)), foi proposta a integração de componentes baseada em modelos reusáveis. Os componentes são projetados e usados como blocos de construção para a

integração, e cada um deles é modelado de forma abstrata, com reações e ações de comunicação. Ele usa composição hierárquica para esconder os detalhes de implementação dos componentes, e mantém os componentes em um mesmo nível de hierarquia interagindo sob um modelo bem definido de integração.

Em **S10** (SHI *et al.*, 2003), foi proposto um ambiente de suporte ao reúso de código de componente, que integra ferramenta de desenvolvimento, ferramenta de simulação de execução e uma biblioteca de código de componente. Então, o ambiente pode fornecer suporte abrangente para muitas atividades de reúso de código de componente, tais como: desenvolvimento, composição, visualização, prototipação, consulta, navegação e transmissão.

Em **S2** (LÓPEZ *et al.*, 2009), foi proposto um *framework* que fornece um conjunto de componentes reusáveis para facilitar a integração de projetos distribuídos. Ferramentas de software legadas podem ser integradas por meio da implementação de adaptadores, representando um conjunto de componentes e interfaces por meio de uma camada de serviços de domínio comum.

Em **S13** (RANA; SUN, 1995), foi proposto um ambiente distribuído para software cooperativo (DECS). O objetivo do DECS é proporcionar uma plataforma flexível para integrar mecanismos variantes e libera os desenvolvedores de aspectos de protocolos de comunicação de baixo nível. Para auxiliar o processo de desenvolvimento, DECS fornece um conjunto de ferramentas de desenvolvimento e *frameworks* reusáveis. Essas ferramentas e estruturas destinam-se a apoiar as diversas atividades do desenvolvimento de software cooperativo.

Em **S23** (HOCHMÜLLER, 1995), foi proposto uma abordagem com foco no aspecto de reúso de componentes que permite a instanciação de programas a partir de componentes genéricos e a composição da aplicação completa baseado em uma estrutura de aplicação genérica. Permite-se então o desenvolvimento de software baseado em componente de integração, prototipagem e reutilização.

Em **S20** e **S21** (BRAGA; WERNER; MATTOSO, 2006; WERNER *et al.*, 2003a), foi proposto um ambiente de desenvolvimento baseado em componentes. Os componentes são desenvolvidos de acordo com um processo específico de Engenharia de Domínio, e são disponibilizados em “pacotes” que incluem diagramas, código fonte, descrições textuais e código binário, entre outros artefatos. Esses artefatos podem ser reusáveis em todas as fases de um desenvolvimento de aplicações, de acordo com um processo de Engenharia de Aplicação.

Em **S18** (HADDAD, 2006), foi proposta uma abordagem que define coleções de componentes reusáveis que estão integrados no ambiente de desenvolvimento. O conceito de interface genérica também é utilizado para definir um mecanismo de interface padrão e controlar a interface das coleções reusáveis no domínio de interesse. A metodologia de coleções reusáveis baseia-se na ideia de integrar componentes relacionados e comumente relacionados, também chamado de famílias de componentes. Com a abordagem de integração, os componentes de

software comuns são agrupados em conjuntos, de modo que cada conjunto está associado a uma interface.

3.2.5 Resposta obtida para a subquestão FE 3- Utiliza-se alguma ferramenta de apoio à integração?

Nessa questão objetivou-se evidenciar se ferramentas suporte tem sido utilizadas para o processo de integração. Por meio da análise dos 28 artigos selecionados para extração de dados, pôde-se perceber que os poucos estudos retrataram que não utilizaram ferramentas de integração. Geralmente, a integração é realizada de maneira *ad-hoc*, sem suporte de ferramenta de integração.

3.2.6 Resposta obtida para a subquestão FE 4- O artigo citou dificuldades e/ou desafios a serem superados?

Nessa questão objetivou-se evidenciar possíveis problemas de integração e desafios para engenharia de software ao integrar ambientes de reúso de software.

- **Ferramentas proprietárias:** Quando a integração envolve ferramentas de reúso que são proprietárias, se o usuário não possuir a licença de utilização, não poderá ter acesso às funcionalidades da interface (S1).
- **Falta de ferramentas de integração:** as formas para integrar técnicas de reúso podem ter sido limitadas pela falta de ferramentas e técnicas adequadas que facilitem a tarefa, exceto de uma forma *ad-hoc* (S25).
- **Identificação de conflitos:** os conflitos não são fáceis de identificar, pois são normalmente escondidos na aplicação. A adaptação de componentes pode ser bastante trabalhosa, o que pode restringir a forma com que o componente será usado (S15).
- **Teste:** alguns problemas podem ocorrer no teste de integração de sistemas como: integração de software aplicativo, testes de integração e integração errada(S26).
- **Incompatibilidade:** As ferramentas envolvidas podem ter formatos incompatíveis de dados, interfaces diferentes, diferentes paradigmas de interação, diferentes níveis de acessibilidade para funcionalidades internas e diferentes direitos de acesso (S7).

3.2.7 Resposta obtida para a subquestão FE 5- SoS tem sido empregado como uma possível solução de integração entre ambientes de Reúso?

Nessa questão objetivou-se evidenciar se os conceitos de SoS estão sendo empregados quando ambientes de reúso de software são integrados. Porém, nenhum dos estudos apontou SoS como possível solução de integração entre ambientes de reúso.

3.2.8 Conclusão Sobre as Questões de Pesquisa

- **RQ1. Como tem sido investigada a integração entre ambientes de reúso de software?**

Nessa questão objetivou-se investigar se ferramentas de Engenharia de Software, especificamente ambientes de reúso, tem sido integrados e evidenciar como essa integração tem sido alcançada. Ao analisar os 28 artigos selecionados para extração de dados é possível perceber que ainda são incipientes os estudos em direção à integração de ambientes de reúso. Existem alguns estudos, como por exemplo (SHI *et al.*, 2003), que estão focados na maioria dos casos em proporcionar o reúso de código e ainda assim de forma isolada de outras ferramentas.

- **RQ2. Sistemas-de-Sistemas tem sido proposto como solução para integrar ambientes de reúso de software?**

Nessa questão objetivou-se evidenciar, quando possível, se o conceito de SoS tem sido proposto como uma possível solução para alcançar a integração entre ambientes de reúso de software. Contudo, ao realizar a análise dos 28 artigos selecionados para extração de dados, foi possível perceber que os conceitos de SoS não tem sido utilizados para integrar ambientes de reúso, o que fomenta a motivação desta proposta de mestrado por objetivar alcançar um ambiente integrado de reúso de software com características de SoS.

O estudo foi importante para entender melhor o estado da arte em relação à integração de ambientes de reúso, como por exemplo os problemas e desafios, bem como extrair ideias e soluções que auxiliaram na condução deste trabalho de mestrado.

3.3 Abordagens para Integração em Sistemas-de-Sistemas (SoS) - Uma Revisão Sistemática

Conforme discutido até aqui, SoS, e mais especificamente SoSI, são áreas que apresentam inúmeras lacunas de pesquisa a serem aprofundadas e ampliadas para diversos outros domínios de aplicação, e engenharia de software é um deles. Nas revisões sistemáticas apresentadas por Nakagawa *et al.* (2013) e Klein e Vliet (2013), não há evidências de que os conceitos de SoS

estão sendo utilizados para integração de ferramentas de engenharia de software, em particular ambientes de reúso. Nesses estudos, é possível notar alguns domínios de aplicação em que SoS tem sido empregado, como por exemplo: Militar, AeroEspacial, Aviônico, Automotivo e Comércio. Contudo, as duas pesquisas citadas anteriormente (Nakagawa *et al.* (2013) e Klein e Vliet (2013)) estão relacionadas à Arquitetura de Software no contexto de SoS e não possuem como foco de estudo a integração dos sistemas constituintes.

Sendo assim, a RS conduzida para este trabalho de mestrado baseou-se na premissa de descobrir/evidenciar métodos de engenharia de software que auxiliam na integração entre os membros de um SoS. A seguir, são apresentadas as questões de pesquisas referentes a RS e seus resultados. Mais detalhes podem ser vistos no Apêndice B, o qual contém a definição do protocolo (planejamento), a condução da revisão sistemática e a sumarização dos dados.

3.3.1 *String de Pesquisa e Máquinas de Busca*

Com base nos objetivos da Revisão Sistemática, definiram-se as seguintes questões de pesquisa:

1. **RQ1. Como tem sido investigada a integração entre os membros de um SoS?**

Nessa questão objetivou-se evidenciar trabalhos que investigam as formas de integração no contexto de um SoS, identificando por exemplo, os problemas, dificuldades, abordagens, técnicas, soluções e vantagens para integrar membros de um SoS.

2. **RQ2. Nesse tipo de estudo, tem sido utilizada alguma ferramenta que auxilia na integração dos sistemas constituintes?**

Nessa questão objetivou-se caracterizar, quando possível, alguma ferramenta que auxilia na integração dos sistemas constituintes de um SoS.

As questões de pesquisa foram utilizadas para construção da *string* de busca, ou seja, a partir das questões foram extraídos termos relevantes que correspondem aos dois tópicos de interesse desta revisão:

1. *System of Systems*: utilizou-se o termo *system of systems*, suas derivações e sinônimos, tais como: *systems of systems*, *Systems-of-systems*, *systems-of-systems*.
2. *Integration*: utilizou-se o termo: *integration*, suas derivações e sinônimos, tais como: *collaboration* e *composition* e *interoperability*.

Operadores *booleanos* AND e OR foram utilizados para a construção da *string* de busca, descrita de maneira geral a seguir. Porém, foram necessárias pequenas alterações para se adequar às restrições de cada base de dados indexada, não perdendo a lógica estrutural.

String de busca genérica:((*{system of systems}* OR *{systems of systems}* OR *{System-of-systems}* OR *{systems-of-systems}*) AND (*integration* OR *collaboration* OR *composition* OR *interoperability*))

Nessa RS, foram consideradas bases de busca relevantes para a área de ciência da computação e engenharia de software, que indexam as principais conferências e periódicos dessas áreas, são elas: ACM Digital library, IEEXplore, Scopus, Springer, Science Direct e Web of Science.

3.3.2 Resultados - Revisão Sistemática

Incluíram-se 53 estudos primários de um total de 1398 retornados pelas bases de dados. Na Figura 13 é mostrada a quantidade de artigos obtidos quando executou-se a *string* de busca nas seis máquinas detalhadas anteriormente.

Encontraram-se 552 artigos repetidos, ou seja, artigos que foram indexados por mais de uma base de dados. Ao avaliar os 846 artigos restantes após a exclusão dos repetidos, selecionaram-se 53 artigos para a etapa de leitura completa. Desta quantidade, a versão completa de cinco artigos não estavam disponíveis para download. Entrou-se em contato com os autores dos artigos, porém sem sucesso. Em seguida, realizou-se uma revisão em par com a orientadora desta proposta de mestrado e excluíram-se outros quatro artigos, por terem sido classificados como fora do escopo da revisão. Sendo assim, restaram 44 artigos para leitura completa.

Na etapa final do processo de seleção dos estudos relevantes, selecionaram-se 29 artigos para extração de dados. Na Figura 14, é apresentada a relação da quantidade de artigos publicados por ano e o tipo de publicação (conferência ou *journal*) dos 29 artigos finais. Na próxima seção, são apresentadas as questões que compuseram o questionário da RS e seus resultados.

3.3.3 Resultados - Revisão SoSI

Como base para extração de dados e análise dos resultados consideraram-se os conceitos de SoS definidos no Capítulo 3, tais como definição, características e tipologia do SoS. Sendo assim, respondeu-se o formulário de extração de dados identificando, quando possível, os pontos em comum de cada artigo com os conceitos anteriormente definidos.

Com o intuito de ilustrar a frequência das palavras chaves dos 29 artigos selecionados para extração de dados, criou-se uma nuvem de palavras (do inglês *wordcloud*) com base nas palavras chaves dos artigos. Na Figura 15, é apresentado o panorama final de ocorrência.

Figura 13 – Processo de Seleção de Estudos Relevantes

Fonte: Dados da pesquisa.

3.3.4 Resposta obtida para a subquestão FE1 - Qual é a proposta do artigo em relação à integração de SoS?

Nessa questão objetivou-se descobrir qual o tipo de contribuição feita sobre integração de sistemas no contexto de SoS. Na Tabela 5, são apresentados os tipos identificados e os respectivos estudos, classificados em cinco categorias: modelagem, *framework*, ontologia, abordagem e padrão arquitetural.

De um total de 29 artigos, 13 retrataram algum tipo de modelagem. Embora não tivessem lidado diretamente com o problema da integração dos sistemas constituintes do SoS, eles descreveram uma modelagem de integração como sendo uma parte importante do processo de

Tabela 5 – Respostas Obtidas para FE1

	Proposta do Estudo				
	Modelagem	Framework	Ontologia	Abordagem	Padrão Arquitetural
Total	13	3	1	10	2
ID Artigo	S1, S2, S6, S7, S8, S11, S12, S14, S15, S18, S21, S22, S29	S20, S23, S26	S24	S4, S5, S9, S10, S13, S17, S19, S25, S27, S28	S3, S16

Fonte: Dados da pesquisa.

fim, dois artigos retrataram algum tipo de padrão arquitetural para integração.

3.3.5 Resposta obtida para a subquestão FE 1.1 - Qual contribuição de integração de SoS foi abordada pelo estudo?

Nessa questão objetivou-se sinalizar quais são as principais contribuições de integração de SoS e evidenciar como e quais aspectos referentes à engenharia de software tem sido utilizados no processo de integração. Observou-se que 62.5% dos artigos fizeram alguma contribuição, 15 artigos responderam a esta questão, são eles: **S2, S3, S4, S6, S7, S8, S10, S12, S14, S15, S16, S20, S21, S23 e S24**. Na Seção B.4, do Apêndice B, são detalhados os estudos e suas contribuições a respeito de integração entre membros de SoS.

3.3.6 Resposta Obtida para a Subquestão FE 2 - Qual o Domínio de Aplicação que Tem Sido Alvo no Contexto de Integração de SoS?

Nesta questão objetivou-se descobrir em quais domínios de aplicação a integração de SoS têm sido empregada, 11 artigos responderam completamente esta questão, o que corresponde a 37.93% do artigos. Na Tabela 6 são apresentados os principais domínios de aplicação identificados nesta RS. Pôde-se observar que 54.13% dos estudos não definiram o domínio de aplicação no qual o estudo está inserido.

No decorrer de cada artigo, observou-se exemplos de SoS em domínios de aplicação diferentes daqueles citados na Tabela 6. Contudo, estes exemplos não foram contabilizados porque não estavam diretamente relacionados ao problema de integração. Os domínios especificados na Tabela 6 foram exclusivamente identificados quanto à Integração de Sistemas-de-Sistemas.

3.3.7 Resposta obtida para a subquestão FE 2.1 - Qual o tipo de SoS foi utilizado?

Dos 29 artigos selecionados para extração de dados, apenas um artigo definiu o tipo de SoS a ser utilizado para integração de sistemas. De acordo com [Madni e Sievers \(2014a\)](#),

Tabela 6 – Domínio de Aplicação

Domínio de Aplicação	Número de Estudos	ID Artigo
Nenhum domínio de Aplicação Específico	16	S6, S8, S10, S11, S13, S15, S16, S17, S18, S19, S21, S22, S23, S26, S27, S29
Defesa e Segurança Nacional	5	S3, S4, S14, S20, S28
Militar	1	S2
Automação Industrial	1	S5
Aeroespacial	1	S9
Gerenciamento de Qualidade	2	S1, S25
Sistema de Observação da Terra	1	S24
Simulação	1	S7
Auto Adaptação	1	S12
Total	29	

Fonte: Dados da pesquisa.

SoS Direcionados ou Reconhecidos são pré-especificados, que faz deles serem previsíveis e compatíveis com métodos tradicionais de validação e verificação. Por outro lado, SoS Virtuais ou Colaborativos não são pré-especificados. Isto significa, que não se pode ter prévia expectativa que caminhos e/ou conteúdos de informação são consistentes no SoS.

3.3.8 Resposta Obtida para a Subquestão FE 2.1.1 - Quais as Características do SoS Foram Detalhadas?

Nesta questão objetivou-se identificar as principais características utilizadas na integração dos sistemas constituintes, ou seja, foi avaliado se o processo de integração garante as características essenciais de um SoS, tais como: independência operacional e gerencial, desenvolvimento evolucionário, comportamento emergente e distribuição geográfica. Porém, ao analisar sistematicamente os 29 artigos selecionados para extração de dados, observou-se que apenas quatro deles sinalizaram este tipo avaliação, o que corresponde a somente 13.79% dos artigos encontrados.

Especificamente, os artigos **S5, S6, S19, S24** demonstraram uma precaução maior, em relação às características, ao propor alguma solução de integração. Porém, estes estudos também não evidenciaram de maneira clara tais características no SoS final. Contudo, esses artigos foram classificados como “Sim” (responderam totalmente) porque além de detalhar algumas características, retrataram alguma outra característica que não tenha sido citada pelos demais, como por exemplo: desenvolvimento adaptativo, conectividade, autonomia, diversidade, reconfiguração e princípios de modularidade.

3.3.9 Resposta Obtida para a Subquestão FE 2.2 - Quais Problemas Referentes à Integração de SoS Foram Abordados?

Nesta questão objetivou-se sinalizar quais são as principais dificuldades quando se realiza integração de sistemas no contexto de SoS. Pôde-se observar que 58.33% dos artigos citaram algum problema/dificuldade de integração. Aqueles que detalharam mais de um problema, foram considerados como artigos que responderam totalmente a esta questão. Sete artigos responderam completamente, o que corresponde a 24.13% dos artigos. A seguir são apresentados os problemas/dificuldades apontados pelos estudos:

1. **Gerencial:** O gerente de projeto deve saber claramente o conceito de sistemas e gerenciamento de sistemas para conseguir integrar sistemas individuais.
2. **Modelagem única:** tentar desenvolver um modelo único que represente o SoS como um todo é uma tarefa complicada, pois dificilmente um único modelo conseguiria representar todas as complexidades do SoS.
3. **Complexidade de interações:** Em SoSI um dos problemas mais desafiadores é lidar com a complexidade de interações entre as entidades do SoS resultante.
4. **Complexidade dos sistemas constituintes:** os sistemas constituintes em um SoS são frequentemente complexos, problemas de integração podem surgir por causa da incapacidade de compreender completamente quais são as funcionalidades do sistema constituinte.
5. **Colaboração:** A heterogeneidade dos CS's tem contribuído para uma baixa colaboração entre os sistemas e o desalinhamento dos objetivos entre os sistemas.
6. **Incompatibilidade de Interfaces:** durante a evolução dos CS's os protocolos e interfaces que os sistemas definem são frequentemente não significativos para comunicação, colaboração e evolução.
7. **Evolução:** é difícil evoluir os sistemas com um todo, de forma a colaborar para o SoS. Geralmente, não existe um processo de integração bem definido e a primeira responsabilidade é de otimizar o próprio desempenho do sistema individualmente, para benefício próprio e não para o objetivo geral do SoS.
8. **Constante atualização:** os sistemas constituintes estão em constante mudanças de software ou hardware, tais como atualizações ou adição de nova funcionalidade. Logo, aumenta-se a complexidade.
9. **Documentação:** a documentação dos sistemas legados não está sempre disponível, detalhada, coerente ou mesmo completa.
10. **Roteiro:** falta de roteiros ou tutoriais que auxiliem engenheiros de software e/ou gerentes de projeto a usar/colocar em prática a integração de sistemas no contexto de SoS.

3.3.10 Resposta obtida para a subquestão FE 2.3 - Quais as vantagens obtidas quando se utiliza o conceito de SoS para integrar sistemas?

Nesta questão objetivou-se evidenciar quais são as principais vantagens que surgem ao utilizar SoS para integração de sistemas. Observou-se que 16.66% dos artigos citaram alguma vantagem ao utilizar SoS para integrar. Aqueles que detalharam mais de uma vantagem, foram considerados como artigos que responderam totalmente a esta questão. Dois artigos responderam completamente, o que corresponde a 8.33% do artigos.

A seguir, na Tabela 7 são apresentadas as vantagens apontadas pelos estudos.

Tabela 7 – Vantagens Apontadas Pelos Estudos da RS

ID	Vantagem
1	Envolvimento mais abrangente de todas as partes interessadas.
2	Uso reduzido de múltiplos recursos.
3	Desenvolvimento de sistemas coerentes.
4	Forma harmonizada e unificada de resolver problemas.
5	Melhor custo-benefício.
6	Aumenta a flexibilidade e possibilidade para incluir outros sistemas.
7	Melhor desempenho operacional em conjunto.
8	Maior motivação da equipe.
9	Redução de custos e reengenharia mais eficiente.
10	Manutenção das características originais do sistema constituinte.
11	Independência operacional e gerencial.
12	Melhor escalabilidade.
13	Maior flexibilidade.
14	Melhor interoperabilidade.
15	Facilidade de manutenção.

Fonte: Dados da pesquisa.

3.3.11 Resposta obtida para a subquestão FE 3 - O estudo apresentou alguma ferramenta de integração?

Nesta questão objetivou-se especificar se o estudo aborda, alguma ferramenta que tenha sido utilizada para alcançar a integração. Foi observado que 29.16% dos artigos citaram alguma ferramenta para integrar membros de SoS. Sete artigos responderam completamente, que corresponde a 24.13%. A seguir são apresentadas algumas ferramentas:

- **FireScrum:** Para guiar o esforço de integração adotaram-se processos de desenvolvimento baseados em Scrum (S3).
- **Mind mapping:** para criar uma representação intermediária do processo de negócio, realizaram por meio do processo de análise da arquitetura, refinamentos iterativos de *mind maps*. Encontrando inconsistências nos documentos de processos de negócio (S3).

- **RDL - Requirements Description Language:** linguagem que pode ser utilizada como base para análise de dependências ou para especificar restrições que resolvem inconsistências. Para isto, utilizou-se as ferramentas de suporte para RDL, VMatriz NLP, possibilitando extrair informações a partir de um documento texto (S4).
- **Tool chain:** propuseram estágios que abrangem todo o ciclo de vida de desenvolvimento: Elicitação de Requisitos, Especificação de Serviços e Projeto de Arquitetura, Implementação e Verificação. A Ferramenta SODA possibilita rastrear requisitos de sistemas orientados a serviços e gera documentos de arquitetura e requisitos. A ferramenta M2Code facilita a modelagem gráfica do serviço. A ferramenta Glue mapeia as máquinas de estado dos componentes para arquiteturas-alvo (S8).
- **SENSE:** fornece suporte arquitetural para evolução do SoS, utiliza XML como base (S10).
- **UPPAAL:** ferramenta que facilita modelagem formal, simulação e modelo de verificação das propriedades comportamentais (S12).
- **DEVS:** *Discrete Event System Specification* é uma ferramenta que possibilita especificar os componentes de um sistema em um evento de simulação discreta. São especificados modelos básicos e como esse modelos são interconectados (S15).
- **M-Model:** é a ferramenta (a nível de modelagem) utilizada pelos Engenheiros de Sistemas-de-Sistemas para integrar sistemas complexos com outros sistemas (S21).

3.3.12 Conclusão Sobre Questões de Pesquisa

- **RQ1. Como tem sido investigada a integração entre os membros de um SoS?**

Existem diversos campos de pesquisa em relação à integração de sistemas que ainda são incipientes e necessitam de pesquisas. De acordo com a análise sistemática dos 29 artigos selecionados para extração de dados, pôde-se observar que SoSI possui alta demanda, e diversos estudos evidenciaram esse desafio em suas propostas de trabalho. Nos últimos anos, houve contribuições significativas que agregaram informações relevantes ao estado da arte de SoS. Na Revisão Sistemática da Literatura de SoS com perspectivas Arquiteturais realizada por Nakagawa *et al.* (2013), é possível observar que a partir do ano de 2003 as pesquisas aumentaram significativamente. Paralelamente, na Revisão Sistemática da Literatura de Arquiteturas de SoS proposta por Klein e Vliet (2013), é possível notar que as pesquisas também aumentaram a partir do ano de 2006. *Systems-of-Systems Integration - SoSI* é um desafio para Engenharia de Software e pode-se notar uma crescente preocupação dos pesquisadores nesse contexto. Porém, esta área ainda carece de pesquisas e uma maior disseminação dos conceitos entre os pesquisadores.

Também é interessante notar que a integração entre membros de um SoS usando SOA (*Service-Oriented Architecture*) tem sido investigada. Um número considerável de publicações, aproximadamente 54.16%, têm explorado o uso de SOA no contexto de SoS. Por exemplo, no artigo **S24 Madni e Sievers (2014a)**, os autores apresentaram uma discussão detalhada sobre o uso de SOA em SoS, com as respectivas vantagens associadas ao reúso de software/sistemas.

- **RQ2. Nesse tipo de estudo, tem sido utilizada alguma ferramenta que auxilia na integração dos sistemas constituintes?**

Nesta pergunta objetivou-se caracterizar, quando possível, alguma ferramenta que auxilia na integração dos sistemas constituintes. Sendo assim, por meio da análise sistemática dos 29 artigos selecionados para extração de dados, pôde-se evidenciar que não foram encontradas ferramentas de integração de sistemas no contexto de SoS. As ferramentas encontradas, detalhadas na Seção 3.3.11, apoiam alguma fase do desenvolvimento, como por exemplo: modelagem da arquitetura, ferramenta de análise de inconsistências nos documentos de requisitos, entre outras. Com isso percebe-se que faltam ferramentas que auxiliem na integração de sistemas no contexto de SoS.

3.4 Trabalhos Relacionados

Madni e Sievers (2014a), identificaram importantes desafios de integração, no contexto de SoS, e afirmaram que ela tem se tornado uma das principais preocupações. Atualmente, a integração tem sido muito utilizada, principalmente em sistemas de defesa aeroespacial, nos quais a interoperabilidade é essencial para satisfazer uma determinada missão. A dificuldade de SoSI está diretamente relacionada ao tipo de SoS. Em um tipo virtual, é mais difícil alcançar a integração, pois parte-se do conceito de que não existe uma autoridade central e objetivos predefinidos. Por outro lado, o SoS dirigido e o reconhecido são pré-especificados, tornando-os previsíveis e compatíveis com métodos de verificação e validação tradicionais, consequentemente, a integração pode ser alcançada com menos dificuldade.

De acordo com **Hause (2014)**, a modelagem de SoS está se tornando cada vez mais importante, tanto em sistemas civis quanto em militares. Geralmente, a modelagem do SoS é feita em sua complexidade total e muitas vezes em único nível de abstração. O ponto chave está na RAS (**PARK; PARK; SUGUMARAN, 2007**), pois esta especificação é utilizada para definir os ativos reusáveis, suas interfaces, características e elementos de suporte. Com o intuito de alcançar um reúso efetivo de projeto, juntamente com o princípio de reutilização e boas disciplinas, uma organização precisa de uma biblioteca de ativos inteligentes, que incentiva os *stakeholders* a encontrar e reusar. A RAS, juntamente com a biblioteca de ativos, fornecem as características necessárias para alcançar esses objetivos.

Em um trabalho de doutorado desenvolvido por aluno da mesma orientadora deste trabalho de mestrado [Ramos \(2014\)](#), [Ramos et al. \(2012a\)](#), [Ramos et al. \(2013\)](#), foi concebida uma abordagem para explorar o desenvolvimento de SoS sob diferentes perspectivas de Engenharia de Software, incluindo requisitos, análise, projeto e reengenharia. Na Engenharia de Requisitos, foi proposta uma abordagem baseada em cenários para descrever o SoS progressivamente como um arranjo de comportamentos elementares, porém significativos. O objetivo é facilitar a descrição e entender melhor o funcionamento dinâmico do SoS.

Para a fase de análise, foram propostas extensões aos *statecharts* ([HAREL, 1987](#)) para melhorar a representação visual das interações entre os sistemas que são membros do SoS. As extensões consistem em notações simbólicas para melhor expressar a concorrência e o paralelismo dos sistemas ([RAMOS et al., 2015](#)). Para a fase de projeto, foi proposta uma extensão ao processo convencional de desenvolvimento de linha de produtos de software (LPS), de forma que a LPS possa se tornar uma fonte natural de membros para o SoS. Para isso, a engenharia de domínio é estendida para entregar componentes capazes de compartilhar funcionalidades em um ambiente de SoS. Desta forma, os engenheiros de aplicação podem projetar famílias de produtos compatíveis com diferentes requisitos de SoS. Para a reengenharia, foi proposta uma extensão que permite a evolução de sistemas legados para participar de um SoS, de forma que os sistemas legados que passaram por reengenharia possam compartilhar habilidades úteis no contexto do SoS. Na Figura 16, ilustra-se a abordagem proposta por [Ramos \(2014\)](#).

Figura 16 – Processo para Engenharia de SoS

Fonte: Adaptada de [Ramos \(2014\)](#).

Com esses estudos pôde-se observar que existem poucos trabalhos na literatura, tanto no que se refere a terminologias quanto definições claras sobre integração de sistemas no contexto de SoS, o que também motivou a realização deste trabalho de mestrado.

3.5 Considerações Finais

Neste capítulo foi apresentada uma sinalização dos estudos de revisão da literatura realizados durante o desenvolvimento deste trabalho de mestrado, mais detalhes de cada um deles são apresentados nos apêndices. Foram contextualizados as principais abordagens de integração em SoSI, assim como, abordagens de integração em ambientes de reúso de software. Por meio desses estudos, pode-se evidenciar que não foram encontrados trabalhos semelhantes ao desenvolvido nesse projeto de mestrado. Utilizou-se da RS como base desenvolver a abordagem de SoS direcionado para integrar ambientes de reúso de software que é apresentada, em detalhes, no próximo capítulo.

UMA ABORDAGEM PARA DESENVOLVIMENTO DE SOS DIRECIONADO - APLICAÇÃO NO SOS-REÚSO

4.1 Considerações Iniciais

Neste capítulo, é apresentado o projeto desenvolvido nesta dissertação de mestrado, cujo principal resultado foi o SoS-Reúso, um SoS que visa facilitar a implementação de comportamentos emergentes relacionados ao reúso de software por meio da colaboração entre ambientes de reúso existentes. Com base na experiência de desenvolvimento do SoS-Reúso, observou-se que poderia ser descrita uma abordagem mais genérica que pode ser útil para apoiar o desenvolvimento de SoS do tipo direcionado para outros domínios de aplicação. Por exemplo, outros tipos de ambientes de desenvolvimento de software também podem se juntar rumo a um SoS.

Assim, na Seção 4.2 essa abordagem é apresentada, consistindo em conceitos importantes a serem levados em consideração ao se desenvolver SoS desse tipo, bem como alguns modelos e visões da arquitetura desses SoS. Na Seção 4.3, é apresentada uma concepção de um SoS-Direcionado para reúso de software, o SoS-Reúso, que é uma instanciação da abordagem apresentada na seção anterior. Na Seção 4.4, é apresentada uma visão geral e os principais componentes de um simulador do SoS-Reúso, o SoS-Reuse-Sim. Na Seção 4.5, apresenta-se de forma mais detalhada a modelagem do SoS-Reuse-Sim e de como a abordagem apresentada na Seção 4.2 foi customizada para o caso específico do SoS-Reuse-Sim. Na Seção 4.6, são apresentados alguns detalhes de implementação do SoS-Reuse-Sim, bem como as ferramentas utilizadas durante seu desenvolvimento. Na Seção 4.7, são mostradas as principais funcionalidades implementadas no simulador. Por fim, na Seção 4.8, são apresentadas as considerações finais do capítulo.

4.2 Uma Abordagem para Desenvolvimento de SoS Direcionado

Nesta seção apresenta-se uma abordagem para construção de SoS do tipo direcionado, que pode servir como base para obter SoS em outros domínios. Conforme mencionado na seção anterior, essa abordagem foi obtida a partir da experiência no desenvolvimento do SoS-Reúso, portanto sua validação será necessária para garantir sua aplicabilidade em outros SoS, o que é sugerido como trabalho futuro.

4.2.1 *Conceitos Básicos da Proposta de SoS Direcionado*

Um SoS do tipo direcionado é composto por sistemas integrados que são construídos e gerenciados para um propósito específico (conforme descrito na Seção 2.3.3). Os sistemas constituintes (CS's) são gerenciados centralmente e mantêm a capacidade de operar de forma independente para cumprir um objetivo comum, mas seu modo de operação quando integrado a outros sistemas é subordinado ao propósito central (MAIER, 1998; PÉREZ *et al.*, 2013). Portanto, ao integrar esses sistemas, facilita-se a implementação de comportamentos emergentes por meio da participação e colaboração de cada um dos CS's.

Na abordagem aqui proposta, respeitando as características intrínsecas de um SoS, os CS's são independentes operacional e gerencialmente e não são obrigados a participar de nenhum SoS. Portanto, eles podem ser desativados ou ativados de acordo com sua disponibilidade. O comportamento emergente é obtido invocando-se serviços dos CS's que estão ativos. Assim, quanto mais CS's disponíveis houver no momento da execução do comportamento emergente, mais fácil será para alcançar os comportamentos emergentes. Visto que os CS's não são obrigados a participar do SoS, caso queiram contribuir para algum SoS em específico, basta oferecer os serviços de acordo com as interfaces requeridas pelo SoS.

Como o SoS tratado nesta proposta é do tipo direcionado, a presença de um coordenador central visa garantir, ou pelo menos maximizar, as chances de que o comportamento emergente seja alcançado. O coordenador tem, portanto, esta responsabilidade de assegurar as condições mínimas para que o SoS continue operando. Se o comportamento emergente não puder ser alcançado, ele deve executar ações para tratar as possíveis consequências.

Outra característica de SoS que é preservada na abordagem aqui proposta é o desenvolvimento evolucionário pois, independentemente de quem os gereencie e de onde eles estejam (distribuição geográfica), tanto o SoS pode evoluir como um todo, invocando novos serviços, quanto os CS's podem evoluir individualmente, desde que continuem provendo os serviços necessários para a composição do SoS no qual ele participa. A utilização de serviços permite criar mecanismos de interação independentemente da plataforma utilizada, evitando a criação de dependências entre os CS's e facilitando a integração entre eles.

Com o intuito de garantir que os CS's estão autorizados a participar de um SoS (no caso do SoS-Reúso, pertencem ao domínio de reúso de software), criou-se o Catálogo de Sistemas Confiáveis. Por meio dele é possível cadastrar CS's confiáveis sob o ponto de vista de um determinado SoS, evitando-se obter resultados inesperados ou comportamentos emergentes que não atendem a um propósito específico. O conceito de Catálogo de Sistemas Confiáveis foi explorado em outro trabalho da equipe de projeto no qual este mestrado faz parte, mais especificamente em uma arquitetura baseada em serviços para SoS com um tipo misto entre virtual e colaborativo (BRAGA; VARGAS; GOTTARDI, 2016).

4.2.2 Modelagem Conceitual e Visão Geral do SoS Direcionado

Na Figura 17, é apresentado o modelo conceitual que ilustra os principais conceitos que fazem parte da solução proposta neste trabalho. O SoS é composto por CS's que formam um Catálogo de Sistemas Confiáveis e estão subordinados ao SoSCoordinator, que coordena o SoS como um todo e estabelece o comportamento emergente.

Figura 17 – Modelo Conceitual Ilustrando a Solução Proposta para SoS Direcionado

Fonte: Elaborada pelo autor.

As entidades presentes no modelo conceitual são detalhadas a seguir:

- **SoS**: entidade que representa o SoS de forma geral, especificando um nome e um objetivo. Esses atributos podem variar de acordo com a necessidade e objetivos a serem alcançados;
- **ConstituentSystem**: entidade que representa os sistemas que compõem o SoS. Inicialmente, é necessário informar cinco atributos básicos, dentre os quais destaca-se o atributo *booleano* *active*, que representa se o CS está disponível ou não para compor um SoS e participar de um comportamento emergente. O administrador do SoS pode fazer o cadastro de um CS em um Catálogo de Sistemas Confiáveis e deixá-lo inicialmente inativo. Posteriormente, após autorizado pelo responsável do CS, o administrador poderá torná-lo ativo e disponível a participar de comportamentos emergentes;
- **CatalogofTrustedSystem**: entidade que representa o Catálogo de Sistemas Confiáveis. Esse catálogo é definido pelo administrador do SoS e garante que apenas sistemas confiáveis são permitidos a participar de um SoS, lembrando que é possível desativar um CS se for necessário;
- **SoSCoordinator**: como são tratados apenas SoS do tipo Direcionado, o SoSCoordinator é necessário para representar o Coordenador Central do SoS; todos os CS's de um SoS trabalham subordinados a ele, e é responsável por estabelecer o comportamento emergente; e
- **EmergentBehavior**: entidade que representa o comportamento emergente desenvolvido pelo SoS-Developer. Esta entidade contém dois atributos, um para definir o nome do comportamento emergente e outro para estabelecer um tempo limite de espera pela resposta. Esses atributos podem ser diferentes para cada comportamento emergente que fizer parte de um SoS.

Vale ressaltar que o SoSCoordinator não é um usuário e sim um módulo de implementação. Na solução aqui proposta, as principais características do coordenador são:

- Utiliza como base o Catálogo de Sistemas Confiáveis;
- Compõe o SoS somente com CS's ativos;
- Coleta os parâmetros de entrada do comportamento emergente, a fim de cumprir com a lógica necessária para alcançar tal comportamento; e
- Verifica se a execução do comportamento emergente atingiu um *timeout*, que é um tempo limite de espera preestabelecido para a obtenção do resultado do comportamento emergente. Em caso positivo, o coordenador pode buscar outras alternativas disponíveis para completar a execução ou avisar o usuário sobre a interrupção da execução do comportamento emergente.

Na Figura 18, ilustram-se os principais elementos da solução proposta e seus relacionamentos. Na parte superior da figura está estrutura envolvendo o administrador e o usuário do SoS, enquanto na parte inferior da figura encontram-se os CS's, que são utilizados por seus proprietários. Mais detalhes sobre os elementos serão fornecidos na Seção 4.4, que instancia essa solução para o SoS-Reuse-Sim.

Figura 18 – Visão Geral da Solução Proposta

Fonte: Elaborada pelo autor.

4.2.3 Casos de Uso para o SoS Direcionado

Com o intuito de disponibilizar uma modelagem de referência para possíveis implementações da ideia deste trabalho, foram criados diagramas de casos de uso UML (do inglês, *Unified Modeling Language*). Cada diagrama descreve o comportamento esperado dos atores definidos na abordagem e suas funções. Na Tabela 8, apresentam-se os atores identificados. A identificação tomou como ponto de partida os atores presentes em trabalhos da literatura, como SoS-User e SoS-Developer (RAMOS *et al.*, 2015; LEWIS *et al.*, 2011). A abordagem aqui proposta tem algumas semelhanças com esses trabalhos, em ambos o processo de desenvolvimento está em

conformidade com SOA, são orientados a serviços, e os serviços são reunidos para atingir o objetivo do SoS. Nesse caso, os CS's podem não obter benefícios quando pertencem a um SoS, pois podem simplesmente fornecer serviços. O SoS-Developer deve fazer ajustes no SoS para que os serviços possam ser integrados e utilizados, no SoS-Reúso, ele também é responsável por desenvolver o comportamento emergente. O SoS-User representa o usuário final que executa e/ou solicita novos comportamentos emergentes ao SoS-Developer, que implanta o novo recurso para os usuários finais do SoS.

Tabela 8 – Atores Identificados na Abordagem para SoS Direcionado

Nome do Ator	Descrição
SoS-Admin	Usuário com altos privilégios, com interesse em fazer log in e log out no sistema e possui permissões para incluir CS's, definir Catálogo de Sistemas Confiáveis, compor SoS e gerenciá-lo como um todo.
SoS-User	Usuário comum do sistema. Principais interesses em criar conta, fazer log in e log out no sistema, pesquisar por CS's, ativos e artefatos, invoca comportamento emergente e solicita novo comportamento emergente.
CS-Owner	Usuário com altos privilégios no CS, ele é responsável por gerenciar e mantê-lo de forma independente.
SoS-Developer	Usuário com <i>expertise</i> no desenvolvimento de software e responsável por criar o SoS, desenvolver o coordenador central, implementar o comportamento emergente por meio dos serviços oferecidos pelos CS's.

Fonte: Elaborada pelo autor.

O SoS-Admin pode ser um engenheiro de software ou qualquer profissional da área que fará a gerência do SoS. Suas principais funções são listadas a seguir:

- Concebe/disponibiliza os CS's, podendo adicionar, remover ou editar configurações básicas dos CS's;
- Define o Catálogo de Sistemas Confiáveis;
- Permite a entrada de um CS no SoS e/ou a saída caso ele classifique como necessário;
- Autoriza o acesso dos usuários ao SoS.

O SoS-User pode ser um usuário que esteja interessado em executar um comportamento emergente, esse usuário deve ser previamente autorizado pelo SoS-Admin para então, ter a permissão de executar um comportamento emergente em um determinado SoS. Essa restrição evita-se que qualquer usuário tenha acesso ao ambiente e associa o usuário logado a um ou mais SoS em que o SoS-User pode executar comportamentos emergentes. Suas principais funções são listadas a seguir:

- Visualiza Catálogo de Sistemas Confiáveis definido pelo SoS-Admin.
- Responsável por inserir os valores dos parâmetros básicos e invoca o comportamento emergente que foi previamente implementado pelo SoS-Developer;

O CS-owner é dono do CS que participa do SoS, sendo responsável por gerenciar e operar o CS de forma independente. Também é sua função garantir/implementar os requisitos necessários para a evolução do sistema individual. A evolução do CS de forma individual pode não ter influência no SoS. Se o SoS precisar evoluir de alguma forma, seja modificando algum comportamento emergente já implementado, ou criando um novo comportamento emergente, pode ser necessário encontrar CS's que ofereçam algum serviço adicional e que atendam à evolução que o SoS necessita. Então, o SoS pode evoluir de acordo com a evolução de cada um dos CS's ou até mesmo pela procura de novos serviços e funcionalidades em novos CS's.

O CS-Owner é quem pode, proativamente, solicitar ao SoS-Admin a participação em um SoS. O CS pode ser retirado de um SoS a qualquer momento, seja por decisão do SoS-Admin ou por meio de uma solicitação feita pelo CS-Owner ao SoS-Admin. Pode-se desabilitar um CS caso julguem que o serviço dele não está servindo bem aos propósitos do SoS, ou se o oferecimento de serviços está acarretando redução no desempenho individual do CS. Na abordagem proposta, apenas o SoS-Admin tem o poder de retirar um CS de um SoS. Portanto, se esta decisão for tomada por parte do CS-Owner, ele deverá solicitar ao SoS-Admin a saída do SoS no qual ele participa.

O SoS-Developer pode ser qualquer Engenheiro de Software com necessidade de um comportamento emergente e que pretende consegui-lo por meio da integração de vários CS's. Suas principais funções são listadas a seguir:

- Implementa o coordenador central;
- Realiza a integração dos CS's estabelecendo interfaces entre eles;
- Responsável pela análise dos requisitos e implementação de novos comportamentos emergentes. Ele pode encontrar uma solução alternativa quando é solicitado um novo comportamento emergente, adaptar os existentes ou até mesmo desenvolver um do zero;
- Responsável por testar e validar se a nova implementação afeta o SoS para então fazer a entrega ao SoS-User;

Para alcançar a integração entre os CS's, o SoS-Developer deve manter o foco na descrição das regras de colaboração e mecanismos de integração do SoS. A integração entre os CS's é ilustrada na Figura 19, em que o SoS-Developer é o responsável pela tarefa de integração.

Na Figura 20, é apresentado o diagrama de casos de uso para cada um dos atores explicados anteriormente, representando quais são os principais casos de uso relacionados à criação e composição do SoS.

Figura 19 – Definição da Interface de Integração

Fonte: Elaborada pelo autor.

4.3 SoS-Reúso: Concepção de um SoS Direcionado para Reúso de Software

Conforme descrito nas seções anteriores, pode-se destacar que um SoS do tipo direcionado é desenvolvido para cumprir um propósito específico e os CS's são gerenciados centralmente. Seguindo as diretrizes apresentadas na Seção 4.2, foi proposto o SoS-Reúso, que na verdade é uma instanciação da abordagem apresentada anteriormente, em que os CS's são ambientes de reúso de software diversos. Portanto, o SoS-Reúso visa facilitar a implementação de comportamentos emergentes relacionados ao reúso de software por meio da integração de ambientes de reúso já existentes. Com isso, pode-se potencializar a atividade de reúso, facilitando a busca por ativos reusáveis em diferentes sistemas constituintes.

O SoS-Reúso respeita as características intrínsecas de um SoS definidas anteriormente. Por utilizar SOA como estilo arquitetural, o SoS-Reúso pode aceitar como constituintes quaisquer ambientes de reúso, independente da plataforma, desde que provejam serviços e atendam às interfaces requeridas pelo SoS. O comportamento emergente pode ser obtido invocando-se serviços dos CS's que estão disponíveis. A adição de novos serviços ou a modificação dos serviços já existentes podem resultar em novos comportamentos emergentes.

É preciso que cada CS que queira participar do SoS-Reúso apresente a seguinte estrutura: (1) ser relacionado a reúso de software; (2) disponibilizar ativos para serem reusados por meio de serviços; e (3) no caso específico do comportamento emergente de busca do ativo melhor ativo, o

Figura 20 – Casos de Uso das Funções dos Atores

Fonte: Elaborada pelo autor.

CS deve fornecer um serviço para cálculo da porcentagem de reúso e das horas de retrabalho de cada ativo. Para implementar outros comportamentos emergentes, poderão ser solicitados outros serviços específicos.

Para colocar em prática a ideia do SoS-Reúso, implementou-se um simulador, denominado SoS-Reuse-Sim, que é apresentado nas seções seguintes. Posteriormente, realizou-se um estudo qualitativo para avaliação desse simulador (ver Capítulo 5).

4.4 SoS-Reuse-Sim: um Simulador do SoS-Reúso

O SoS-Reuse-Sim é um simulador do SoS-Reúso em que os ambientes de reúso são instâncias da AIREs, descrita na Seção 2.2.4. Algumas decisões de projeto foram tomadas para viabilizar sua implementação neste projeto de mestrado. As principais decisões foram: 1) como a AIREs ainda está em fase de projeto, não estando disponíveis as implementações das interfaces de serviço, o SoS-Reuse-Sim não utilizou serviços para acesso aos CS's; 2) o SoS-Reuse-Sim implementou apenas os módulos necessários para invocar um comportamento

emergente utilizado como exemplo; 3) nesta primeira versão do SoS-Reuse-Sim, o SoS-User pode ser qualquer usuário devidamente autenticado no sistema, não houve restrição de acesso aos comportamentos emergentes como seria em outros SoS; e 4) o SoS-Reuse-Sim utilizou uma base de dados local, específica do cenário de aplicação do SoS-Reuse-Sim, que representa as informações dos CS's e seus ativos reusáveis.

Na Figura 21, é mostrada a visão geral do SoS-Reuse-Sim e seus principais módulos. No cenário de aplicação abordado neste projeto, cada CS é uma simulação de uma instância independente da AIRES, em que existem disponíveis ativos e artefatos reusáveis. Em futuras versões, os CS's podem ser substituídos por outros ambientes de reúso, desde que eles forneçam os serviços necessários para obter os comportamentos emergentes desejados.

Figura 21 – Figura Geral do SoS-Reuse-Sim

Fonte: Elaborada pelo autor.

Pode-se observar que o SoS-Reuse-Sim é dividido em quatro módulos principais. Dentre eles, dois representam papéis distintos de usuário e que possuem acesso direto ao SoS-Reuse-Sim

(SoS-Admin e SoS-User). Na parte superior da figura encontra-se o módulo GUI SoS-Admin. Por meio dele, o SoS-Admin pode realizar quatro operações básicas (CRUD - *Create, Read, Update e Delete*) na entidade: *ConstituentSystem*. Ou seja, é possível fazer essas operações básicas no simulador a fim de prepará-lo para compor o SoS-Reuse-Sim. Então, deve-se inserir os CS's, definir o Catálogo de Sistemas Confiáveis e disponibilizar os comportamentos emergentes. Além disso, pode-se também gerenciar a autorização dos usuários ao simulador e às funcionalidades disponíveis por meio da autenticação de usuários.

O SoS-User acessa o simulador utilizando o módulo GUI SoS-User, o qual realiza autenticação do usuário e em seguida pode-se executar um comportamento emergente. O módulo do SoS-User possui relação direta com o módulo do SoS-Admin e SoSCoordinator visto que, para um comportamento emergente ser executado, é necessário que o SoS-Admin adicione os CS's confiáveis a um SoS. Os módulos GUI SoS-Admin e GUI SoS-User serão sempre os mesmos independentemente do SoS a ser utilizado, os únicos campos que podem mudar na parte gráfica são os valores dos parâmetros de entrada do comportamento emergente a ser executado.

Na parte central da figura, representa-se o módulo SoSCoordinator, que possui relação direta com todos os outros módulos, pois eles são gerenciados centralmente e trabalham juntos para cumprir um propósito específico. O Coordenador Central compõe o SoS-Reuse-Sim somente com CS's ativos; realiza a comunicação entre os CS's para alcançar o comportamento emergente que retorna o melhor ativo a ser reusado, aqui denominado "*Which is the best asset*"; e verifica se a execução do comportamento emergente atingiu o *timeout* preestabelecido.

Na parte inferior da figura encontram-se os módulos que representam os CS's, e ilustra-se a estrutura interna e quais são os CS's do SoS-Reuse-Sim. Para cada um dos CS's existe um CS-Owner responsável por gerenciar e operar o CS de forma independente. Cada CS é representado por uma instância da AIREs que é composta por um ou mais repositórios de ativos e artefatos. O ativo pode ser visto como algo abstrato e os artefatos como algo concreto, por exemplo: o ativo *Car Financing* contém artefatos como código fonte e casos de teste relativos a um sistema de financiamento de carros. Os CS's oferecem métodos que são invocados pelo SoSCoordinator para execução do comportamento emergente.

Dentre os quatro atores citados na Seção 4.2, pode-se enfatizar que: O SoS-User e SoS-Admin são aqueles que possuem acesso direto ao simulador, por meio de telas de acesso para o papel de User e Admin, respectivamente. Para os atores SoS-Developer e CS-Owner, pode-se dizer que não se trata de acessos diretos ao simulador e sim papéis extras que colaboram para a composição e formação do SoS-Reuse-Sim.

Nesta versão do SoS-Reuse-Sim, foi implementado um comportamento emergente, "*Which is the best Asset?*", em que se coletam alguns parâmetros do SoS-User e faz-se uma busca no repositório de ativos para sugerir o melhor ativo a ser reusado. Esta funcionalidade permite alcançar uma maior efetividade no reúso dos ativos disponíveis no repositório.

O SoS-Reuse-Sim pode ser composto por diversos CS's existentes em diversas organizações ou instituições, por exemplo, uma instância poderia estar no ICMC e conter ativos reusáveis resultantes dos diversos projetos acadêmicos ou institucionais; uma outra instância poderia estar em uma empresa de desenvolvimento de software que disponibiliza ativos na área de telecomunicações; ainda uma outra poderia estar em uma outra universidade que disponibiliza ativos reusáveis mais específicos de computação gráfica. Cada instância da AIRES atua como um repositório de diversos tipos de ativos (por exemplo, classes, modelos, componentes, padrões, serviços, casos de teste), de forma que quando o comportamento emergente for invocado, o SoS-User poderá visualizar quais itens estão disponíveis para reúso naquele momento e adicionalmente contar com a recomendação do melhor ativo a ser reusado.

4.5 Modelagem do SoS-Reuse-Sim

4.5.1 Modelo Conceitual

Nesta seção apresenta-se o modelo conceitual utilizado como referência para desenvolvimento do SoS-Reuse-Sim. Esse modelo é uma extensão do modelo genérico apresentado anteriormente, sendo que foram adicionadas entidades específicas do SoS-Reuse-Sim. Na Figura 4.5.3, são apresentados os detalhes desse modelo, em que o *Package 1* refere-se aos conceitos gerais da abordagem descrita na Seção 4.2 (Figura 17) e o *Package 2* refere-se aos conceitos dos CS's que formam o SoS-Reuse-Sim. Os conceitos do *Package 1* são instanciados para o SoS-Reuse-Sim. Já os conceitos do *Package 2* referem-se a um subconjunto mínimo de classes da RAS (ver Seção 2.2.2.2) que se espera que cada CS implemente para poder compor o SoS-Reuse-Sim. Conforme já mencionado no início da Seção 4.4, a implementação dessas classes deveu-se a uma decisão de projeto, por não estar ainda disponível a camada de serviços da AIRES.

A seguir são detalhadas cada uma das entidades presentes no **Package 1**.

- **SoS**: O SoS-Reuse-Sim é uma instância do SoS-Reúso, ou seja, o atributo “name” recebe “SoS-Reuse-Sim” e o atributo “goal” recebe “integrar ambientes de reúso de software”.
- **ConstituentSystem**: essa entidade representa o CS que compõe o SoS-Reuse-Sim e é formada por cinco atributos. Como exemplo de instância dessa entidade, no caso tem-se uma instância da AIRES com os seguintes atributos: “id” recebe o valor “1”, “name” recebe “CSSãoCarlos”, “serverAddress” recebe “192.168.0.1”, “authentication” recebe “senha12345rp” e “active” recebe valor “true”.
- **CatalogofTrustedSystem**: essa entidade representa uma instância do Catálogo de Sistemas Confiáveis, em que o atributo “name” recebe “Catálogo SoS-Reuse-Sim” e o atributo “description” recebe “Lista dos Sistemas Confiáveis ao SoS-Reuse-Sim”.

Figura 22 – Modelagem Conceitual do SoS-Reuse-Sim

Fonte: Elaborada pelo autor.

- **SoSCoordinator**: essa entidade representa uma instância do coordenador central do SoS, no exemplo do SoS-Reuse-Sim, contém a implementação do comportamento emergente para escolher o melhor ativo.
- **EmergentBehavior**: essa entidade representa o comportamento emergente presente no SoS-Reuse-Sim, por exemplo, o atributo “name” recebe “*Which is the best asset*” e o atributo “timeout” recebe o valor “2” minutos.

No cenário de aplicação abordado neste trabalho, todos os CS’s que compõem o SoS-Reuse-Sim são instâncias da AIRES. Logo, eles possuem no mínimo as entidades apresentadas no **Package 2** da Figura 4.5.3. Mas, se não fossem instâncias da AIRES, bastaria oferecerem serviços que retornassem as informações correspondentes aos métodos `averageNumberOfHoursOfRework` e `averageReusePercentage` (que são parte da interface). A seguir são detalhadas cada uma das entidades presentes no **Package 2**.

- **Asset**: entidade que define a identidade de um ativo de software e representa o conjunto de artefatos a ser reusado pelo SoS-User. De forma abstrata, o ativo contém sete atributos que ajudam a melhor descrevê-lo, são eles: nome do ativo, número de downloads,

identificador, direitos de acesso, breve descrição, estado e versão do ativo. Os atributos `name` e `shortDescription` são tipicamente as principais peças de informação que os usuários vêem quando procuram por ativos no repositório. O nome do ativo deve refletir uma estratégia geral de solução e opcionalmente o problema que ele resolve. O `shortDescription` deve ser adequado para uso de um item no qual vários nomes e breves descrições são mostradas ao SoS-User. Essa entidade fornece dois métodos essenciais para execução do comportamento emergente: os métodos `averageNumberOfHoursofRework` e `averageReusePercentage` (ALMEIDA *et al.*, 2007; POULIN; CARUSO, 1993).

- **ApplicationDomain:** entidade que representa os diversos domínios aos quais o ativo pode pertencer.
- **Usage:** entidade que representa as atividades de uso do ativo. Há várias formas de conduzir atividades de uso em um ativo, algumas delas são mais genéricas e outras mais específicas, pois em diferentes contextos algumas atividades são mais importantes do que outras.
- **UserComments:** entidade que compõe **Usage** e representa uma especificação do uso realizado pelo SoS-User em cada ativo. Com o intuito de obter um *feedback* do usuário definiram-se três atributos, são eles: (1) `numberOfHoursofRework` é responsável por registrar as horas de retrabalho que o SoS-User gastou depois de baixar o ativo e seus artefatos para reusar. Ou seja, quantas horas foram gastas adaptando o ativo para alcançar o objetivo que se deseja. (2) `reusePercentage` é responsável por registrar a porcentagem reusada do ativo, representando o quão útil e reusável foi o ativo para alcançar o objetivo. (3) `comments` é responsável por registrar algum comentário sobre a utilização do ativo, seja um elogio ou crítica/sugestão. Trata-se de um campo de texto livre em que o SoS-User fica apto a deixar seu *feedback* sobre o ativo.
- **Artifact:** entidade que representa um produto de trabalho que pode ser reusado. Um artefato é um arquivo dentro de um ativo, um ou mais artefatos podem compor um ativo, os ativos e seus artefatos são baixados por meio de um link que direciona para repositórios externos onde eles estão armazenados.
- **ArtifactType:** entidade que representa os diversos tipos que classificam um artefato. Eles podem ser de qualquer tipo (binário, texto, .java, .php, web.xml, etc.), na prática o tipo do artefato serve para mapear as extensões do arquivo.
- **Review:** entidade que representa a ação de revisar um ativo. A inclusão dessa entidade se deu pelo fato de exemplificar uma chamada a diversos CS's que possuam revisores de ativos. Inicialmente, este foi considerado um fator opcional, já que torná-lo obrigatório poderia trazer dificuldades e restrições devido a possibilidade de faltar revisores em determinados momentos.

- **User:** entidade que representa a realização de revisões do ativos para entrada no SoS-Reuse-Sim e/ou a pessoa responsável por popular os três atributos da entidade UserComments.

4.5.2 Diagrama de Sequência de Composição do SoS-Reuse-Sim

Na Figura 23, é apresentada a ideia de como acontece a composição do SoS-Reuse-Sim. Inicialmente, o SoS-User invoca o comportamento emergente e o Coordenador Central recebe essa notificação. Em seguida, é feita uma busca no Catálogo de Sistemas Confiáveis para selecionar os CS's que estão ativos e assim obter uma lista dos sistemas disponíveis, naquele momento, que irão participar do SoS-Reuse-Sim. Então, para todos os CS's que estiverem com *status* ativo, o Coordenador Central irá realizar a integração por meio do estabelecimento de interfaces e estabelecer o comportamento emergente, levando-se em conta essa lista de sistemas disponíveis e aptos a participar do SoS.

Figura 23 – Diagrama de Sequência de Composição do SoS-Reuse-Sim

Fonte: Elaborada pelo autor.

Parte-se do princípio que o comportamento emergente é previamente implementado e que toda a lógica e implementação foram desenvolvidas antes de serem disponibilizadas no SoS-Reuse-Sim. Por ser do tipo Direcionado, os objetivos e recursos de integração são predefinidos e desenvolvidos para um propósito específico. Ou seja, pode haver inúmeros comportamentos emergentes no SoS-Reuse-Sim, desde que seja previamente implementado pelo SoS-Developer.

4.5.3 Diagramas do Comportamento Emergente - Which is the best asset?

Na Figura 24, apresenta-se a ideia do comportamento emergente escolhido para o cenário de aplicação do SoS-Reuse-Sim: “Which is the best asset?”.

Figura 24 – Diagrama de Sequência para Sugerir o Melhor Ativo Reusável do SoS-Reuse-Sim

Fonte: Elaborada pelo autor.

Pode-se observar que o SoS-User é responsável por inserir dois parâmetros básicos para a busca por ativos, são eles: o domínio e o tipo do artefato. Os valores desses atributos podem variar de acordo com a necessidade do SoS-User. Os ativos reusáveis variam, bem como os CS's ativos, então poderá ocorrer resultados diferentes na exibição do comportamento emergente em diferentes momentos.

Um exemplo de invocação do comportamento emergente é quando o SoS-User buscar por um artefato de código fonte no domínio de Hotel, os detalhes do melhor ativo serão exibidos, tais como: ativos relacionados, domínio de aplicação e horas de retrabalho. Após a inserção dos parâmetros, o Coordenador Central recebe uma notificação de chamada do comportamento emergente e obtém os parâmetros passados anteriormente.

O Coordenador Central executa o comportamento emergente e o melhor ativo é escolhido com base em dois principais atributos, são eles: porcentagem de reúso e horas de retrabalho do ativo. Cada CS é responsável por calcular a média de cada um dos seus ativos reusáveis referente a esses dois parâmetros. Na entidade ConstituentSystem, apresentada na Figura ,

existem dois métodos para realizar essa média, são eles: `averageNumberOfHoursofRework()` e `averageReusePercentage()` (ALMEIDA *et al.*, 2007; POULIN; CARUSO, 1993).

Considera-se que esses atributos são essenciais para incentivar e potencializar a atividade de reuso em diversos níveis de abstração. Então, obtém-se sucesso ou falha na execução do comportamento emergente e consecutivamente é mostrado para o SoS-User.

Na Figura 25, é possível notar que existem três partes necessárias para a execução de um comportamento emergente, são elas: (1) invocar o comportamento emergente, que possui um nome associado e tempo aproximado de espera pela resposta; (2) inserir os parâmetros de entrada requeridos para execução e (3) o resultado esperado é mostrado para o SoS-User. Utilizou-se o comportamento emergente 1 no cenário de aplicação do SoS-Reuse-Sim, que possui dois parâmetros de entrada obrigatórios, no entanto, os demais comportamentos emergentes 2, n... que forem participar do SoS-Reuse-Sim podem apresentar uma variedade de parâmetros de entrada e consecutivamente terão saídas diferentes.

Figura 25 – Passos Realizados Durante a Execução de um Comportamento Emergente

Fonte: Elaborada pelo autor.

4.6 Implementação

A primeira versão do simulador do SoS-Reuse-Sim foi desenvolvida em Plataforma Java 8 Desktop. Entretanto, não foram utilizados serviços no desenvolvimento do simulador, pois as instâncias da AIRES estão em fase de desenvolvimento e, portanto, não estão prontas para serem utilizadas. Assim, desenvolveu-se todo o simulador utilizando a linguagem de programação Java

Desktop, de forma a representar as instâncias da AIRES, promover a integração entre elas e alcançar o comportamento emergente desejado. Posteriormente, poderão ser feitas alterações no simulador para que ele possa funcionar por meio da utilização de serviços e em ambientes reais.

Para desenvolvimento do simulador utilizou-se banco de dados Mysql, em conjunto com o gerenciador MysqlWorkbench. Conforme mostrado na Figura 26, foram criadas as seguintes tabelas no banco: (1) artifact - para armazenar os diversos artefatos do repositório, inicialmente foram disponibilizados 40 artefatos, (2) asset - para armazenar os diversos ativos dos CS's que farão parte do SoS-Reuse-Sim, inicialmente foram disponibilizados 20 ativos, (3) assetinformation - para armazenar informações de uso dos ativos reusados, (4) constituentsystem - para armazenar os sistemas constituintes que compõem o simulador, foram disponibilizados 10 CS's para o SoS-Reuse-Sim e (5) user - para armazenar os usuários que terão acesso ao simulador, inicialmente foram disponibilizados dois usuários, um com permissões de SoS-User e outro com permissões de SoS-Admin.

Figura 26 – Informações do Banco de Dados

Fonte: Elaborada pelo autor.

Utilizou-se Netbeans IDE (versão 8.2)¹ para desenvolvimento do simulador. Na Figura 27, são mostradas as configurações do arquivo .xml que contém a implementação da conexão com o banco de dados, cujo nome é bancoSoS. Contém também o mapeamento de alguns objetos, como por exemplo: Asset, Artifact e User.

Figura 27 – Código em .xml

Fonte: Elaborada pelo autor.

Na Figura 28, pode-se destacar três principais pacotes de desenvolvimento no simulador, (br.sos, br.sos.admin.views e br.sos.user.views). Eles representam as telas do SoS-Admin, SoS-User e a principal tela de acesso, que são explicados a seguir:

Figura 28 – Principais Pacotes Desenvolvidos no Simulador

Fonte: Elaborada pelo autor.

¹ <https://netbeans.org/downloads/>

- `br.sos`: Esse pacote possui duas classes, a *view* da classe `Main.java` contém a implementação da tela principal, na qual é possível escolher o tipo de usuário a ser usado: SoS-User ou SoS-Admin. A *view* da classe `ViewAbout.java` contém a implementação das informações básicas sobre plataforma de desenvolvimento, autores e versão do simulador.
- `br.sos.admin.views`: Esse pacote possui quatro classes referentes as telas de navegação do SoS-Admin. A *view* da classe `ViewAsset.java` contém a implementação da tela de visualização dos ativos e artefatos de um CS. A *view* da classe `ViewCadastroCS.java` contém a implementação e campos necessários para a tarefa de adicionar, editar ou remover um CS do simulador. A *view* da classe `ViewCatalog.java` contém a implementação da tela que permite ao SoS-Admin visualizar o Catálogo de Sistemas Confiáveis existente naquele momento e por fim a *view* da classe `ViewLoginAdmin.java` contém a implementação da tela de log in necessária para ter acesso ao papel do SoS-Admin. O usuário deve ter sido cadastrado com permissão de acesso a esse papel.
- `br.sos.user.views`: Esse pacote possui quatro classes referentes as telas de navegação do SoS-User. A *view* da classe `ExecutionRequestEB.java` contém a implementação da tela de requisição de um novo comportamento emergente. Ela será utilizada quando o SoS-User necessita de algum comportamento emergente que não esteja disponível no simulador. A *view* da classe `ViewInfoBestAsset.java` é a principal tela do comportamento emergente. Nela será mostrado o melhor ativo e suas informações principais, assim como os ativos relacionados e os comentários de usabilidade. A *view* da classe `ViewLoginUser.java` contém a implementação da tela de log in necessária para ter acesso ao papel de SoS-User. O usuário deve ter sido cadastrado com permissão de acesso a esse papel. A *view* da classe `ViewSimulationSoS.java` contém a implementação da tela de interação com o SoS-User, é por meio dela que será escolhido o SoS a ser utilizado, o comportamento emergente e seus parâmetros de entrada.

4.7 Visão Geral das Funcionalidades do Simulador

4.7.1 Funcionalidades do SoS-User

Na Figura 29, é apresentada a tela principal do simulador desenvolvido. Por meio dela é possível fazer log in utilizando dois diferentes usuários. O primeiro deles é o SoS-User, o qual possui privilégios para desempenhar funções básicas no simulador, como por exemplo: executar um determinado comportamento emergente provido por um SoS. No SoS-Reuse-Sim, está disponível o comportamento emergente de escolher o melhor ativo, o qual requer alguns parâmetros de entrada que serão explicados posteriormente. O segundo deles é o SoS-Admin, o qual possui privilégios e funções gerenciais no simulador, como por exemplo: alterar, criar, editar e cadastrar CS's.

Figura 29 – Principal Tela do Simulador do SoS

Fonte: Elaborada pelo autor.

Na Figura 30, é apresentada a tela referente ao SoS-User, em que se destacam oito campos. Ao iniciar essa tela apenas o primeiro campo é exibido para o SoS-User, os campos seguintes são dinâmicos e dependentes entre eles, e são mostrados na tela de forma sequencial. Como primeiro passo, é necessário selecionar o SoS a ser utilizado, já que pode existir mais de um SoS desenvolvido previamente pelo SoS-Developer. Em nosso cenário de aplicação foi desenvolvido o SoS-Reuse-Sim. Como segundo passo, é necessário selecionar o comportamento emergente, no qual as opções de comportamento emergente dependem do SoS selecionado no campo anterior, ou seja, cada SoS pode oferecer um ou mais comportamentos emergentes. O comportamento emergente é obtido apenas após a integração dos CS's que compõem o SoS, e não está presente nos CS's de forma isolada. No SoS-Reuse-Sim, foi desenvolvido o comportamento emergente *“Which is the best asset?”* que escolherá o melhor ativo dentre os disponíveis no repositório.

Cada comportamento emergente é desenvolvido para um propósito específico e requer diferentes parâmetros de entrada, ou seja, o SoS-User pode inserir diferentes parâmetros ao executar comportamentos emergentes. Os campos para inserção dos parâmetros de entrada são dinâmicos e são mostrados na tela de acordo com o que foi definido e implementado no

Figura 30 – Primeira Tela do SoS-User

The screenshot shows the SoS-User interface. At the top, there is a logo for ICMG-USP and LabES. Below the logo, the title "Simulator of System-of-System (SoS)" is displayed. The interface includes several dropdown menus for configuration: "SoS" set to "SoS Reuse", "Emergent Behavior" set to "Which is the best asset?", "Domain" set to "Hotel", and "Type" set to "Source Code". There are two buttons: "View Catalog" and "Execute". Below the configuration section, there is a "List of available assets:" section with a table of assets. The table has columns for "id", "Name", "Short Description", and "Application Domain". The second asset, "XYZ Delivery", is highlighted. Below the table, there is a "Comments about Assets:" section with a text area containing a comment about the assets.

id	Name	Short Description	Application Domain
1	ABC Fast Food	All reusable artifacts concerning management of customer orders and delivery, human resources ...	Restaurant
2	XYZ Delivery	Source code and use cases for delivery system including customer registration/authentication and...	Restaurant
3	Casual Dining Restaurant	This asset has several includes that can be reused in many other softwares.	Restaurant
4	Barbecue Restaurant	This asset contains all the artifacts that were developed to the development package and this ass...	Restaurant
5	Business Hotel	Source code, conceptual model and system architecture for customer registration, reservation of r...	Hotel
6	Convention Hotel EUA	Source code, conceptual model and system architecture for customer registration, reservation of r...	Hotel

Comments about Assets:

Asset Comment
This asset contains all the artifacts that were developed to the development package and this asset contains a set of all the information that can be reus...

Fonte: Elaborada pelo autor.

comportamento emergente. Para o *Which is the best asset?* foram definidos dois parâmetros de entrada, domínio e tipo do artefato. Então, como terceiro passo, é necessário selecionar o domínio do ativo para que possa ser feita a busca. No cenário de aplicação deste projeto, foi selecionado *Hotel*, porém, foram disponibilizados outros três domínios, são eles: *Health*, *Restaurant* and *Financial*. Como quarto passo, é necessário selecionar o tipo de artefato a ser buscado no repositório de ativos. No cenário de aplicação foi selecionado *Source Code*, porém foram disponibilizados outros dois tipos de artefatos, são eles: *Conceptual Model* e *Test Cases*. Antes de executar o comportamento emergente é possível visualizar o Catálogo de Sistemas Confiáveis, isso pode ser útil para o SoS-User ter uma visão geral de quais CS's estão participando naquele momento do SoS-Reuse-Sim. Ao clicar no botão *View Catalog*, o sistema exibe todos os sistemas confiáveis que participam naquele momento do SoS selecionado.

Para fins informativos, é possível visualizar os nomes, uma breve descrição e o domínio de todos os ativos pertencentes ao SoS-Reuse-Sim. Essa informação pode mudar de acordo com cada SoS. Ao clicar sobre um ativo é possível visualizar na tabela inferior alguns comentários sobre a utilização do ativo selecionado. Essa funcionalidade permite que o SoS-User faça uma breve visualização dos ativos disponíveis e tenha um conhecimento prévio sobre eles.

Após preencher os quatro campos citados anteriormente, o próximo passo é executar o comportamento emergente. Ao clicar no botão **Execute**, é mostrado o melhor ativo com suas respectivas informações. Na Figura 31, são mostradas as informações sobre o melhor ativo.

Figura 31 – Informações Sobre o Melhor Ativo

Fonte: Elaborada pelo autor.

Na parte superior da tela exibe-se as informações básicas de um ativo. No lado direito superior é possível visualizar três botões: ao clicar em **Steps Executed** é mostrado um fluxo de todos os passos executados pelo comportamento emergente para a obtenção do melhor ativo. Esse fluxo começa pela ação do SoS-User de invocar o comportamento emergente e vai até a obtenção da resposta de sucesso ou falha. Ao clicar no botão **Related Assets** identifica-se os ativos relacionados àquele escolhido como o melhor. Essa identificação é feita por meio de uma busca no repositório de ativos por todos aqueles que possuem o mesmo domínio de aplicação que foi inserido como parâmetro de entrada. Ou seja, na Figura 30 foi inserido o domínio de *Hotel*, então, os ativos relacionados serão todos aqueles que pertencem a esse mesmo domínio. Na Figura 32, são mostrados todos os ativos do domínio de *Hotel* e suas respectivas informações. O último botão **Download** possibilita ao SoS-User baixar o ativo, ao clicar nesse botão o SoS-User

é direcionado para um link externo onde se encontram os ativos reusáveis. Na parte inferior da tela, são exibidos inicialmente os comentários sobre o melhor ativo, mas ao clicar em algum ativo relacionado são exibidos os comentários do que estiver selecionado na tabela central da tela.

Figura 32 – Ativos Relacionados ao Escolhido como o Melhor

The screenshot displays the 'The best Asset is:' section of the SoS-Admin interface. It includes fields for Name (Hotel Accessories), Version (3.1), Number of Download (99), Access Right (Public), Rework (2), and Reuse Percentage (90). The State is 'Completed'. A Short Description is provided: 'products stock control and user authentication/authorization. The artifacts are independent and can be reused isolated or together.' There are buttons for 'Related Assets', 'Download', and 'Steps Executed'.

Below this section is a table titled 'Assets' with the following data:

id	Name	Number of ...	Access Right	State	Application ...	Version	Reuse Per...	Hours of ...	Short Description	Url
5	Business Hotel	14	Public	Completed	Hotel	3.1	60	1	Source code, conceptual ... w...	...
6	Convention Hotel EUA	87	Public	Completed	Hotel	2.2	10	1	Source code, conceptual ... w...	...
8	Resort Hotel Brasil	34	Public	Completed	Hotel	7	60	0.4	Source code, conceptual ... w...	...
11	Business Hotel	23	Public	Completed	Hotel	2.4	10	1.2	Source code, conceptual ... w...	...
12	Ariport Hotel	45	Public	Completed	Hotel	4.3	22	1.3	The asset contains a set ... w...	...
13	Resort Hotel	21	Public	Completed	Hotel	2	60	0.9	All reusable artifacts con... w...	...
16	Hotel Accessories	99	Public	Completed	Hotel	3.1	90	2	All reusable artifacts con... w...	...
18	Casino Hotel	11	Public	Completed	Hotel	2.8	50	1	Source code, test cases a... w...	...

At the bottom, there is an 'Asset Informations' section with an 'Asset Comment' field containing the text: 'This asset contains all the artifacts that were created during the development cycle, for example: source code, conceptual models, system architecture.'

Fonte: Elaborada pelo autor.

4.7.2 Funcionalidades do SoS-Admin

Na Figura 33, apresenta-se a tela referente ao SoS-Admin, em que se destacam sete tópicos. O primeiro tópico é apenas uma descrição sobre o Catálogo de Sistemas Confiáveis e ações do SoS-Admin. O segundo tópico é responsável por exibir todos os CS's já cadastrados como confiáveis no SoS-Reuse-Sim, assim como algumas informações básicas de cada um deles. O botão Add CS pode ser utilizado para adicionar CS ao Catálogo de Sistemas confiáveis. Ao clicar nele é mostrado um formulário com quatro campos, na parte inferior da tela, para inserção das informações básicas de cada CS. O botão Remove CS pode ser utilizado para remover CS's do Catálogo de Sistemas Confiáveis, basta que o CS a ser removido esteja selecionado na tabela

superior da tela. O botão **Edit CS** pode ser utilizado para edição de informações em cada um dos CS's presentes no Catálogo de Sistemas Confiáveis, bastando que o CS a ser editado esteja selecionado na tabela superior da tela.

Figura 33 – Tela Principal do SoS-Admin

Catalog of Trusted Systems

Description
Below are listed all the registered trusted systems that can participate of a SoS and the SoS Admin can add, remove or edit CS's.

Table of registered Constituents Systems

id	Name	Server Address	Authentication	Active
1	CSRibeiraopreto1	192.168.1.2	senha:*****	true
2	CSGoias	155.133.0.1	senha:***** user: goias	true
3	CSSaopaulo	187.123.0.1	senha:***** user: goiano	true
4	CSMorrinhos	122.124.1.1	senha:***** user: mhs	true
5	CSSaoCarlos	186.168.1.23	senha:***** user: sanca	true
6	CSAraraquara	166.143.87.44	senha:***** user: arara	true
7	CSUberlandia	122.156.44.1	senha:***** user: uber	true
8	CSUberaba	144.111.0.1	senha:***** user: beraba	false
9	CSSantoAndre	133.123.11.13	senha:***** user: sa	true

Constituent System

Name:

Server Address:

Authentication:

Active:

Fonte: Elaborada pelo autor.

Há quatro campos que compõem o formulário para adicionar, remover ou editar um ativo. São eles: **name**, em que é possível inserir/alterar o nome do CS; **server address**, em que é necessário adicionar informações do IP do servidor; **authentication**, em que são inseridas informações de autenticação, como por exemplo: usuário e senha; por fim o atributo **Active**, que permite configurar um CS como ativo ou inativo. O SoS-Admin pode adicionar um CS ao Catálogo de Sistemas Confiáveis mas deixá-lo inicialmente como inativo ou vice-versa.

O botão *View Asset* pode ser utilizado para visualizar os ativos de cada CS. Na Figura 34, é mostrada a tela de visualização dos ativos e seus artefatos, em que foram destacadas três partes. A primeira parte representa o CS selecionado para visualização dos seus ativos. A segunda parte consiste em clicar sobre o botão *View Asset*, que é responsável por exibir os ativos do CS

selecionado anteriormente. Um CS pode ter mais de um ativo disponível, então, a terceira parte consiste em clicar sobre um dos ativos e visualizar a lista de artefatos que ele oferece.

Figura 34 – Tela de Visualização dos Ativos e Artefatos de um CS

Fonte: Elaborada pelo autor.

4.8 Considerações Finais

Neste capítulo foram apresentados a abordagem para desenvolvimento de SoS Direcionado e o simulador da implementação dessa ideia, que fazem uso dos conceitos e materiais apresentados nos capítulos anteriores. O SoS-Reuse-Sim disponibiliza dois papéis de usuário, SoS-User e SoS-Admin, com o intuito de promover um suporte computacional que facilita a atividade de reúso de software. O desenvolvimento desse simulador pode auxiliar o engenheiro de software durante o processo de desenvolvimento, por facilitar o reúso de ativos de software que podem estar em diversas localizações. No próximo capítulo, é apresentado um estudo experimental realizado para avaliar o simulador desenvolvido, com o intuito de obter dos usuários um *feedback* sobre sua usabilidade e adequação funcional.

ESTUDO QUALITATIVO DA USABILIDADE E ADEQUAÇÃO FUNCIONAL DO P-SOS-REÚSO

5.1 Considerações Iniciais

Neste capítulo, avalia-se o SoS-Reuse-Sim por meio da realização de um estudo qualitativo e relatos de usabilidade/funcionalidade coletados dos usuários. Documentou-se o estudo qualitativo usando-se a estrutura proposta por (WOHLIN *et al.*, 2012): definição, planejamento, operação do estudo qualitativo e análise e interpretação dos resultados. Os diversos usos dos processos e produtos deste trabalho são relatados de maneira informal, com enfoque maior nos resultados obtidos e lições aprendidas.

Deve-se ressaltar que não foi feita uma avaliação geral do simulador, tendo sido a avaliação executada em uma única etapa. A realização de uma avaliação geral demandaria muito tempo e ultrapassaria os limites de tempo estabelecidos para a pesquisa. No entanto, acredita-se que essa avaliação geral poderá ser feita, em trabalhos futuros, de modo a aperfeiçoar o simulador. Foi realizada uma análise qualitativa com a participação de oito alunos de pós graduação.

Na Seção 5.2 são apresentados a definição e o planejamento do estudo. Na Seção 5.3 são apresentados os procedimentos e estrutura do estudo qualitativo. Na Seção 5.4 são apresentados os dados obtidos para cada formulário. Na Seção 5.5 é apresentada uma breve discussão dos resultados. Na Seção 5.6 são apresentadas ameaças à validade dos resultados e, por fim, na Seção 5.7 são apresentadas as considerações finais do capítulo.

5.2 Definição e Planejamento do Estudo Qualitativo

A proposta deste estudo é investigar os benefícios e as limitações do simulador desenvolvido em relação a dois fatores: adequação funcional e usabilidade, que são duas características de um conjunto de normas para avaliação da qualidade dos produtos de software (ISO/IEC, 2011). Para isso, realizou-se uma análise qualitativa do simulador com o intuito de obter evidências do quão bom ele é para apoiar a atividade de reúso, assim como obter evidências de restrições e melhorias. Os grupos possivelmente interessados são: engenheiros de software, arquitetos de software, analistas de sistemas e diversos outros *stakeholders* que estejam envolvidos no processo de desenvolvimento de software. A análise do estudo foi orientada a descoberta, com base em questionários subjetivos e objetivos. Deseja-se investigar a adequação funcional e usabilidade do ponto de vista do SoS-User e do SoS-Admin. Considerando esses aspectos, definiu-se neste estudo abordar duas questões de pesquisa:

- **RQ1. Quão importante o SoS-Reuse-Sim pode ser considerado para apoiar a atividade de reúso de software?** Pretende-se obter o *feedback* do usuário ao utilizar um ambiente integrado de reúso e se ele o considera útil para o desenvolvimento de software, sem precisar fazer busca manual por diversos ativos.
- **RQ2. No que se refere à usabilidade/adequação funcional do simulador o que pode ser melhorado em termos de suas duas diferentes funções de usuário?** Pretende-se descobrir quais as limitações do simulador e o que pode ser melhorado no que se refere a usabilidade e adequação funcional. Essas duas características serão explicadas a seguir. Devem ser destacados pontos positivos e negativos de cada uma das funções para serem analisados e implementados na próxima versão do simulador.

A norma ISO/IEC 25010 (ISO/IEC, 2011) consiste em um conjunto de normas para avaliação da qualidade dos produtos de software. A SQuaRE - *Software Product Quality Requirements and Evaluation* propõe um modelo de qualidade composto por oito características e trinta e um atributos de qualidade, relacionados à qualidade interna e externa. Dentre as oito características, selecionou-se duas delas para serem aplicadas no estudo qualitativo: adequação funcional e usabilidade. A seguir são descritas essas duas características e seus atributos de qualidade.

- **Adequação Funcional:** grau em que um produto ou sistema fornece instruções que correspondam às necessidades explícitas e implícitas quando utilizado sob as condições especificadas.
 - **Compleitude Funcional:** grau no qual um conjunto de funções abrange todas as tarefas especificadas e os objetivos do usuário.

- **Corretude Funcional:** grau no qual um produto ou sistema fornece resultados corretos, com um determinado grau de precisão.
- **Funcionalidade apropriada:** grau em que as funções facilitam a realização das tarefas e dos objetivos para os quais o sistema foi especificado.
- **Usabilidade:** grau em que um produto ou sistema pode ser utilizado por usuários específicos para atingir metas especificadas com eficácia, eficiência e satisfação, em determinado contexto.
 - **Conhecimento adequado:** grau em que os usuários podem reconhecer se um produto ou sistema é apropriado para suas necessidades.
 - **Apreensibilidade:** grau em que o produto ou sistema pode ser usado para alcançar objetivos específicos de aprendizagem para utilização do produto ou sistema com eficácia, eficiência, inexistência de risco e satisfação, em um contexto de uso especificado.
 - **Operabilidade:** grau de facilidade com a qual um produto ou sistema é operado ou controlado.
 - **Acessibilidade:** grau em que um produto ou sistema pode ser usado com a mais ampla gama de características e capacidades, a fim de alcançar um objetivos especificado em um contexto de uso determinado.
 - **Proteção de erro do usuário:** grau em que um sistema protege os usuários de cometer erros.
 - **Estética da interface de usuário:** grau em que uma interface de usuário permite interação agradável e satisfatória.

Dentre os atributos de qualidade das características adequação funcional e usabilidade, selecionou-se sete atributos de qualidade, são eles: completude funcional, corretude funcional, funcionalidade apropriada, conhecimento adequado, operabilidade e acessibilidade. Desconsiderou-se os demais, por serem classificados como fora do escopo desse estudo qualitativo. Sendo assim, foram elaboradas questões para medir o atendimento de cada um dos atributos em relação ao simulador desenvolvido. Nas seções a seguir são apresentadas as questões e demais etapas do estudo qualitativo.

5.2.1 Definição de Metas

Utilizou-se a organização proposta por Solingen *et al.* (2002) *Goal/Question/Metric* (GQM), descrevendo o estudo qualitativo em 5 partes: objeto de estudo, propósito, perspectiva, foco de qualidade e contexto.

- **Objeto do estudo:** simulador desenvolvido para representar o SoS-Reuse-Sim.

- **Propósito:** examinar o *feedback* do usuário sob dois diferentes papéis de uso do simulador: (SoS-User e SoS-Admin), a fim de obter evidências sobre a potencialização da atividade de reúso de software e ao mesmo caracterizar algumas deficiências no uso do simulador. Especialmente, investiga-se se os conceitos de SoS, reúso de software e RAS estão sendo bem aplicados e representados na ferramenta.
- **Foco de Qualidade:** consiste em avaliar o quão bom e útil o simulador pode ser para apoiar o reúso de software, por integrar ambientes que estavam espalhados.
- **Perspectiva:** do ponto de vista de um usuário comum, representado pelo SoS-User e do ponto de vista de um usuário com habilidades gerenciais no simulador, o SoS-Admin.
- **Contexto:** realizou-se o estudo qualitativo com oito estudantes de pós graduação em Ciência da Computação da Universidade de São Paulo - USP e Universidade Federal de São Carlos - UFSCar.

5.3 Operação do Estudo Qualitativo

Três questionários foram aplicados neste estudo qualitativo, são eles: (1) Questionário pessoal para caracterizar o perfil¹ e o nível de experiência do participante em relação aos conceitos abordados. (2) Questionário sobre o papel de SoS-User² para coletar informações do participante sob o ponto de vista de um usuário comum que esteja interessado em fazer reúso de software. (3) Questionário sobre o papel de SoS-Admin³ para coletar informações do participante sob o ponto de vista de um usuário com funções gerenciais e responsabilidades maiores para composição de um SoS. Cada participante do estudo qualitativo utilizou os dois diferentes papéis de usuário do simulador e respondeu aos três questionários. Esses questionários visaram coletar dados e evidências dos usuários a fim de responder as questões de pesquisa definidas anteriormente.

O projeto do estudo utilizado para avaliar o simulador inclui seis passos, os quais são descritos na Tabela 9:

Para as questões objetivas dos questionários definiram-se alguns critérios para as respostas, como a seguir:

- **Muito baixo:** você ainda não ouviu falar sobre isso antes.
- **Baixo:** você já ouviu falar sobre isso antes.

¹ <https://docs.google.com/forms/d/e/1FAIpQLScL1Q_6y1lCFFMVgXlHuoQ5Zoq11veiW2YbEbY2scmybQSlmw/viewform>

² <https://docs.google.com/forms/d/e/1FAIpQLSdTfLIFkJug-92rVpy_n3J6rK2cuB-BYEvCiKyBLaRmq8mcrQ/viewform>

³ <https://docs.google.com/forms/d/e/1FAIpQLSc2RV1_YKb4JD8FEIxBfSNpHUHqbt4GzX8faUsQb0L0InfIA/viewform>

Tabela 9 – Detalhamento das Etapas do Estudo Qualitativo

N.	Tarefa:	Realizado por:	Tempo Estimado:
1	Questionário Pessoal (profile)	Todos os Participantes	5 minutos
2	Treinamento Teórico (SoS, Reúso de Software e RAS)	Proponente deste Trabalho	25 minutos
3	Demonstração de Utilização do Simulador	Proponente deste trabalho	10 minutos
4	Realização do estudo qualitativo	Todos os Participantes	25 minutos
5	Questionário sobre módulo SoS-User	Todos os Participantes	10 minutos
6	Questionário sobre módulo SoS-Admin	Todos os Participantes	10 minutos
Total:			1 h 25 min

Fonte: Dados da pesquisa.

- **Normal:** você conhece os conceitos básicos.
- **Alto:** você conhece e aplicou esses conceitos, poucas vezes, na academia, indústria, ou ambos.
- **Muito alto:** você conhece e aplicou esses conceitos, muitas vezes, na academia, indústria ou ambos.

5.4 Análise e Interpretação dos Resultados

5.4.1 Questionário Pessoal

Cada pergunta do formulário foi avaliada com peso um, com exceção da questão: **Como você considera o ambiente integrado para apoiar o reúso de software?**, presente tanto no formulário do SoS-User quanto no SoS-Admin, que foi avaliada com peso dois. Por se tratar da ideia geral deste trabalho e abordar os conceitos mencionados anteriormente, decidiu-se atribuir um peso maior a essa pergunta. A seguir são apresentadas as respostas para o questionário pessoal. Na Figura 35 é apresentada a quantidade de pessoas em relação à idade de cada uma delas. Pode-se observar que não há uma diferença muito grande entre as idades dos participantes, dois participantes estão no limite inferior de 23 anos e apenas um participante com 28 anos.

Na Figura 36 é apresentado o nível de escolaridade dos participantes do estudo qualitativo. A maioria deles é estudante de mestrado da USP-São Carlos e/ou UFSCar e dois participantes são estudantes de doutorado.

Na primeira pergunta objetivou-se reportar o nível de conhecimento de SoS dos participantes. **1. Qual o seu nível de conhecimento sobre SoS (Systems-of-Systems)?**. Conforme mostrado na Figura 37, metade deles possui baixo nível de conhecimento sobre SoS e apenas um deles possui alto conhecimento sobre o assunto.

Figura 35 – Relação da Quantidade de Participantes X Idade

Fonte: Dados da pesquisa.

Figura 36 – Nível de Escolaridade dos Participantes

Fonte: Dados da pesquisa.

Na segunda pergunta objetivou-se reportar o nível de conhecimento de Reúso de Software dos participantes. **2. Qual o seu nível de conhecimento sobre Reúso de Software?**. Conforme mostrado na Figura 38, pode-se perceber que metade deles possui conhecimento regular em reúso de software, na maioria dos casos foram disciplinas estudadas durante a pós graduação que abordaram o conceito de reúso de software. Em contrapartida, um participante possui conhecimento muito alto sobre reúso de software.

Conforme mostrado na Figura 39, objetivou-se reportar o nível de conhecimento de RAS dos participantes, **3. Qual seu conhecimento sobre RAS (*Reusable Asset Specification*)?**. Pode-se perceber que metade deles possui baixo conhecimento sobre RAS. Em contrapartida, um participante possui conhecimento muito alto sobre o assunto. Por não ser um conceito amplamente utilizado, esperava-se que a maioria dos participantes não tivesse alto nível de conhecimento sobre esse tópico e portanto, foi abordado no treinamento teórico de forma mais detalhada.

Figura 37 – Nível de Conhecimento sobre SoS

Fonte: Dados da pesquisa.

Figura 38 – Nível de Conhecimento sobre Reúso de Software

Fonte: Dados da pesquisa.

Com o intuito de reportar se os participantes do estudo qualitativo já utilizaram algum tipo de repositório de ativos e em caso afirmativo mencionar qual foi o tipo utilizado, na quarta pergunta **4. Você já usou algum repositório de ativos de reúso de software? Qual tipo?**, pode-se perceber que muitos deles não utilizaram um repositório de ativos para desenvolvimento de software e apenas um deles já utilizou esse tipo de recurso. Um dos participantes que nunca utilizaram repositórios que facilitassem o reúso declarou já ter utilizado imagens, sons e códigos para desenvolvimento de um jogo educacional. Esses itens ficavam armazenados em pastas do próprio computador e foram buscados manualmente na internet. Um participante declarou ter feito uso de diversos repositórios de Gerenciamento de Configuração.

Na quinta pergunta objetivou-se reportar a experiência dos participantes em relação

Figura 39 – Nível de Conhecimento sobre RAS

Fonte: Dados da pesquisa.

ao desenvolvimento de software. **5. Caso já tenha atuado no desenvolvimento de software, indique quantos anos.** Conforme mostrado na Figura 40, pode-se notar que dois participantes possuem três anos de experiência como desenvolvedor e em contrapartida três pessoas não tiveram experiência em desenvolvimento.

Figura 40 – Experiência com Desenvolvimento de Software

Fonte: Dados da pesquisa.

5.4.2 Questionário Sobre o Papel do SoS-User

A seguir são apresentadas as respostas para o questionário sobre o papel do SoS-User. Na primeira pergunta o intuito era de reportar o nível de satisfação dos participantes em relação

a busca por ativos. **1. Você ficou satisfeito com a forma que foi feita a pesquisa/busca por ativo?**. Conforme mostrado na Figura 41, pode-se observar que três participantes se sentiram muito satisfeitos com a forma de buscar/pesquisar por ativos no repositório, o que corresponde a 37.5%. Três participantes se sentiram satisfeitos, que corresponde aos mesmos 37.5% e dois participantes se sentiram insatisfeitos com a forma de buscar por ativos, que corresponde a 25%.

Figura 41 – Nível de Satisfação com a Busca por Ativos

Fonte: Dados da pesquisa.

Com o objetivo de coletar sugestões dos participantes, deixou-se a seguinte sub pergunta aberta: **1.1 Você teria alguma sugestão de melhoria para a pesquisa/busca de ativos?**. Pode-se perceber algumas críticas e sugestões, alguns consideraram a busca um pouco restrita, por ser baseada em apenas em dois atributos e por ser obrigatória a inserção do tipo do artefato. Sugeriu-se que o usuário pudesse escolher apenas o domínio e obter e os melhores ativos. Em contrapartida, sugeriu-se que a busca pudesse ter mais opções de critérios, tais como: o mais novo ou o mais velho, busca por nome do ativo e/ou comentário.

O nível de satisfação dos usuários, em relação aos dois atributos requeridos para a execução do comportamento emergente no SoS-Reuse-Sim (domínio do ativo e tipo de artefato), é investigado na terceira pergunta. **Os atributos de busca selecionados são suficientes?** Havia três opções de resposta: Suficiente, Indiferente ou Não Suficiente. Na Figura 42, pode-se perceber que 87.5% dos participantes consideraram suficientes os atributos colocados para a busca por ativos no SoS-Reuse-Sim.

Sob o ponto de vista do SoS-User, na terceira pergunta objetivou-se coletar dos participantes a importância da visualização dos atributos de ativo. **1.3 Disponibilizar ao SoS-User a visualização dos atributos (horas de retrabalho, percentual de reúso e comentários em geral) de um ativo, colabora para ele ser encorajado a fazer reúso. Dessa afirmação, qual sua opinião?**. Na Figura 43, pode-se observar que cinco participantes consideraram que essa visualização colabora muito para encorajar o reúso de software, que corresponde a 62.5%. Três participantes consideraram que essa visualização colabora para encorajar o reúso de software, que corresponde a 37.5%.

Figura 42 – Pergunta em Relação aos Atributos de Busca

Fonte: Dados da pesquisa.

Figura 43 – Visualização dos Atributos do Ativo

Fonte: Dados da pesquisa.

Com o objetivo de obter um *feedback* do participante em relação aos atributos do ativo, deixou-se a seguinte sub pergunta: **1.3.1 Você considera esses atributos do ativo suficientes?**. Na Figura 44, pode-se observar que seis participantes consideraram suficientes os atributos dos ativos presentes no SoS-Reuse-Sim, que corresponde à 75%. Um participante considerou indiferente os atributos apresentados e um participante considerou não suficiente, que correspondem a 12.5% cada um deles.

Também deixou-se uma sub questão aberta com o intuito de coletar sugestões/melhorias dos participantes: **Você teria alguma sugestão de atributos para serem adicionados ao ativo?**. Relataram-se sugestões para inserir a data em que o ativo foi adicionado e tempo em que ele está disponível, pois o ativo pode ser recente e melhor, entretanto, não foi classificado como *Best*

Figura 44 – Avaliação dos Atributos de ativo

Fonte: Dados da pesquisa.

Asset, por falta de avaliações de sua utilização (por exemplo: horas de retrabalho e porcentagem de reúso). Outra sugestão foi adicionar os detalhes de plataforma, pois mesmo que se encontre o melhor ativo ele pode ser incompatível com o projeto que se está desenvolvendo no momento.

Na quarta pergunta objetivou-se relatar o quão útil pode ser considerado o simulador desenvolvido. **1.4 O simulador desenvolvido é útil para apoiar o reúso de software. Dessa afirmação, qual sua opinião?** Na Figura 45, pode-se observar que três participantes consideraram o ambiente integrado como Muito Útil para apoiar o reúso de software, que corresponde a 37%. Cinco participantes consideraram como Útil para apoiar o reúso de software, que corresponde a 63%. Em linhas gerais, pode-se considerar que o simulador teve uma avaliação positiva em relação ao intuito de apoiar a atividade de reúso de software. Os participantes se sentiram motivados a utilizar o simulador como ferramenta de apoio para desenvolvimento de software.

Figura 45 – Avaliação de Quão Útil é o Simulador Desenvolvido

Fonte: Dados da pesquisa.

Na quinta pergunta objetivou-se relatar o quão fácil é utilizar a busca automática por

ativos, proporcionada pelo simulador. **O quão bom (fácil) é utilizar o ambiente integrado em relação ao fato de não precisar fazer uma busca manual por ativos?.** Na Figura 46, pode-se observar que seis participantes consideraram como Muito Bom o recurso de busca automática, que corresponde a 75% dos participantes. Dois participantes consideraram como Bom o recurso de busca automática, que corresponde a 25% dos participantes.

Figura 46 – Avaliação de Quão Fácil é Buscar Automaticamente os Ativos

Fonte: Dados da pesquisa.

Na sexta pergunta objetivou-se relatar o *feedback* do usuário em relação ao comportamento emergente presente no SoS-Reuse-Sim. Então, implementou-se um comportamento emergente que sugere o melhor ativo a ser reusado. A partir dessa ideia, essa pergunta teve o intuito de relatar se essa funcionalidade colabora para eficiência do reúso. **1.6 Sugerir o melhor ativo, colabora para uma melhor eficiência no reúso de ativos. Dessa afirmação, qual sua opinião?.** Na Figura 47, pode-se observar que cinco participantes consideraram que sugerir o melhor ativo Colabora para uma melhor eficiência no reúso de ativos, que corresponde a 62.5%. Três participantes consideraram que sugerir o melhor ativo Colabora Muito para obter melhor eficiência no reúso de ativos, que corresponde a 37.5%.

Na sétima pergunta objetivou-se avaliar a fácil visualização dos ativos de um CS no simulador. **A ferramenta proporciona fácil visualização dos ativos de um CS. Dessa afirmação, qual sua opinião?.** Na Figura 48 pode-se observar que sete participantes consideraram como Fácil a forma de visualização de ativos dentro do simulador, que corresponde a 87.5%. Um participante considerou como indiferente essa visualização, que corresponde a 12.5%.

Na oitava pergunta objetivou-se avaliar a hierarquia dos CS's presentes no simulador. **1.8 A hierarquia de CS-> Ativos-> Artefatos está bem representada no simulador?.** Na Figura 49, pode-se observar que cinco participantes consideraram como Bem Representada a hierarquia utilizada, e corresponde a 62.5%. Três participantes consideraram como Muito Bem representada a hierarquia dos CS's, que corresponde a 37.5%.

Figura 47 – Avaliação da Importância de Sugerir o Melhor Ativo

Fonte: Dados da pesquisa.

Figura 48 – Avaliação da Facilidade de Visualização dos Ativos

Fonte: Dados da pesquisa.

Na nona pergunta objetivou-se coletar um *feedback* dos participantes do estudo qualitativo sobre o treinamento teórico/prático realizado pelo proponente de trabalho de mestrado. **1.9 O treinamento foi bom/adequado?** Na Figura 50, são apresentadas as avaliações dos usuários. Pode-se observar que cinco participantes consideraram como Muito Bom o treinamento realizado, que corresponde a 62.5%. Três participantes consideraram como Bom o treinamento, que corresponde a 37.5%.

Por fim, deixou-se duas questões abertas com o intuito de coletar dos usuários sugestões e críticas. A primeira delas, é referente a ideia de integração dos CS's em um único ambiente: **Integrar os CS em um único ambiente, que forma o SoS-Reuse-Sim, faz sentido? Qual seu *feedback* em relação a isso?** Na Tabela 10, são apresentados os comentários deixados pelos usuários. Essa avaliação foi realizada pelo papel do SoS-User, ou seja, pode ser qualquer pessoa

Figura 49 – Avaliação da Hierarquia dos CS's

Fonte: Dados da pesquisa.

Figura 50 – Avaliação do Treinamento Realizado

Fonte: Dados da pesquisa.

que esteja interessada em reusar ativos durante o ciclo de desenvolvimento de software. Pode-se perceber que os comentários foram bastante positivos e apoiadores à ideia apresentada neste trabalho, ou seja, os usuários ficaram encorajados a utilizar o simulador para potencializar a atividade de reúso. Em alguns comentários, pode-se perceber algumas sugestões de melhorias que serão implementadas na próxima versão do simulador, como por exemplo: informar em qual base está sendo feita a busca por ativos. Dentre os oito participantes do estudo qualitativo, seis deles optaram por fazer algum tipo de comentário, pois essa questão era opcional, deixando o usuário livre para fazer críticas, sugestões ou até mesmo elogios.

A segunda questão aberta é: **2.0 Você teria alguma sugestão de melhoria?. Se teve dificuldades, por favor descreva-as.** A única dificuldade relatada foi de um participante no

Tabela 10 – *Feedback* dos Usuários em Relação a Ideia do Simulador sob o Ponto de Vista do SoS-User

	Comentário
1	Sim, faz sentido. Os usuários poderão encontrar em um único local tudo (ou pelo menos muito) do que procuram.
2	Sim, pois facilita a comunicação e o entendimento do trabalho. Faz muito sentido sim, pois facilita o reuso dos ativos.
3	Sim, o usuário possui uma grande autonomia para reusar artefatos e com fácil adaptação ao simulador.
4	Sim faz muito sentido, deixa muito transparente os CS's selecionados. Mas, não dá para saber se o resultado vai ser o mesmo se eu procurar depois de algum tempo, porque não mostrou em quais bases a pesquisa foi realizada.
5	Sim, pois permite a verificação facilitada de CS's mais confiáveis para reuso de software.
6	Sim, pois é possível visualizar uma lista dos CS disponíveis com seus respectivos ativos. Além disso, é possível obter o melhor ativo, o que contribui muito para o reuso.

Fonte: Dados da pesquisa.

começo da avaliação para o entendimento do simulador, porém, depois de algumas dúvidas sanadas pelo proponente deste trabalho o participante conseguiu entender o funcionamento do simulador e pôde realizar o estudo qualitativo de forma mais tranquila.

5.4.3 Questionário Sobre o papel SoS-Admin

A seguir são apresentadas as respostas para o questionário sob o ponto de vista do SoS-Admin. Na primeira pergunta objetivou-se reportar o nível de dificuldade em publicar um CS no simulador. **1. O quão trabalhoso é publicar/disponibilizar um CS no simulador?** Na Figura 51, é possível visualizar a porcentagem das respostas dos usuários. Pode-se observar que seis usuários classificaram que não é trabalhoso publicar um CS no simulador. Em linhas gerais, pode-se dizer que ficaram satisfeitos com a forma proposta, que representa 75%. Dois usuários consideraram como pouco trabalhoso, que corresponde a 25%, ou seja, também não sentiram dificuldades e não acharam que a tarefa demandasse muito trabalho.

Na segunda pergunta objetivou-se coletar dos participantes uma avaliação sobre o ponto de vista gerencial do simulador. **2. As funções de adicionar, remover e editar CS colaboram para melhor gerência do SoS. Dessa afirmação, qual sua opinião?.** Na Figura 52, pode-se perceber que cinco deles consideraram que essas funções Colaboram para a gerência do SoS, que corresponde a 62.5%. Dois participantes consideraram como Colabora Muito, que corresponde a 25%, e um deles classificou como indiferente para gerência do SoS.

Na terceira pergunta objetivou-se reportar o quão fácil é a visualização dos ativos. **3. A ferramenta proporciona fácil visualização dos ativos de um CS. Dessa afirmação, qual sua opinião?.** Pode-se perceber que cinco deles consideraram como Fácil a visualização dos ativos, que corresponde a 62.5%. Dois deles consideraram como Muito fácil, que corresponde a 25%.

Figura 51 – Avaliação do Quão Trabalhoso é Publicar um CS

Fonte: Dados da pesquisa.

Figura 52 – Avaliação das Funções de Adicionar/Remover/Editar

Fonte: Dados da pesquisa.

Um deles considerou como difícil a forma de visualizar os ativos presente em cada um dos CS's.

Na quarta pergunta objetivou-se relatar a importância do Catálogo de Sistemas Confiáveis.

4. O quão importante você considera manter um Catálogo de CS's confiáveis? Na Figura 53, são apresentadas as avaliações dos usuários. Pode-se perceber que quatro deles classificaram como Muito Importante manter essa lista de CS's confiáveis, que corresponde a 50%. Três participantes classificaram como Importante manter esse Catálogo, que corresponde a 37.5%. Um participante classificou como Indiferente, que corresponde a 12.5%.

Na quinta pergunta objetivou-se relatar a importância do atributo active de cada CS.

5. Fazer uma monitoração do atributo active de cada CS's colabora para manter o comportamento emergente disponível. Dessa afirmação, qual sua opinião? Pode-se observar que seis participantes consideraram esse atributo como Colabora para manter o comportamento

Figura 53 – Avaliação da Importância de Manter o Catálogo de Sistemas Confiáveis

Fonte: Dados da pesquisa.

emergente disponível, que corresponde a 75%. Dois participantes consideraram como colabora muito, que corresponde a 25%.

Sob o ponto de vista do SoS-Admin, analogamente ao que foi avaliado para o módulo SoS-User, na sexta pergunta objetivou-se relatar o quão útil pode ser considerado o simulador desenvolvido, **6. O ambiente integrado é útil para apoiar o reúso de software. Dessa afirmação, qual sua opinião?**. Pode-se observar que quatro participantes consideraram o ambiente integrado como Muito Útil para apoiar o reúso de software, que corresponde a 50%. Quatro participantes consideraram como Útil para apoiar o reúso de software, que corresponde a 50%. Assim como no SoS-User, pode-se considerar que o simulador teve uma avaliação positiva em relação ao intuito de apoiar a atividade de reúso de software, os participantes se sentiram motivados a utilizar o simulador como ferramenta de apoio para desenvolvimento de software.

Por fim, deixou-se duas questões abertas com o intuito de coletar dos usuários sugestões e críticas. A primeira delas é referente a ideia de integração dos CS's em um único ambiente, porém, utilizando do papel do SoS-Admin: **Integrar os CS em um único ambiente, que forma o SoS-Reuse-Sim, faz sentido? Qual seu feedback em relação a isso?**. Na Tabela 11, são apresentados os comentários deixados pelos usuários. Essa avaliação foi realizada pelo módulo SoS-Admin, ou seja, pode ser qualquer profissional de Engenharia de software que esteja interessada gerenciar o simulador como um todo. Pode-se perceber que os comentários foram bastante positivos e apoiadores à ideia apresentada neste trabalho. É possível destacar que os usuários ficaram encorajados a utilizar o simulador para potencializar a atividade de reúso. Em alguns comentários, pode-se perceber algumas sugestões de melhorias que serão implementadas na próxima versão do simulador, como por exemplo: informar em qual base está sendo feita a busca por ativos.

Tabela 11 – *Feedback* dos Usuários em Relação a Ideia do Simulador sob o Ponto de Vista do SoS-Admin

	Comentário
1	Creio que criar um SoS direcionado e centralizado cria dependências que os fornecedores de artefatos podem não gostar.
2	Sim faz todo o sentido, torna a atividade de reúso mais prática e fácil.
3	Faz sentido, o meu <i>feedback</i> é que potencializa o reúso por ser um ambiente integrado.
4	Sim, pois facilita a verificação de CS's como um todo, se ele é reutilizável para um SoS local ou para outros SoS.
5	Sim, faz sentido. Achei válida essa integração, pois ajuda os usuários a encontrar o melhor artefato, seguindo um critério confiável.
6	Sim. Conforme o aumento de sistemas constituintes no SoS-Reuse-Sim a praticidade para disponibilizar e utilizar sistemas vai ser crucial para a relevância da ferramenta, contudo a integração faz todo sentido para acontecer o impacto no estado da arte.
7	Sim, pois é possível obter uma lista de CS's com seus respectivos ativos.
8	Essa integração sem dúvida pode colaborar efetivamente para o reúso, entretanto, acredito que deverão ser definidas algumas características que facilitem o reúso.

Fonte: Dados da pesquisa.

A segunda delas: **2.0 Você teria alguma sugestão de melhoria?. Se teve dificuldades, por favor descreva-as.** Na Tabela 12, são apresentadas sugestões de melhorias de cinco participantes. Pode-se perceber algumas importantes sugestões de melhorias, que serão implementadas na próxima versão do simulador, como por exemplo: agrupar os ativos por domínio e ordem alfabética.

Tabela 12 – Sugestões de Melhorias dos Usuários no papel de SoS-Admin

	Sugestões de Melhorias
1	Usar uma janela separada para ver ativos no papel de SoS-Admin atrapalha, deveria ser um painel com todas as informações. <i>Usage</i> é mais importante do que atributo <i>downloads</i> , principalmente se o ativo for novo pode ser mais interessante considerar o uso do que o número de <i>downloads</i> . A descrição de domínio é sucinta demais para ajudar na decisão de escolher ativo. Poderia ter uma forma do CS-owner solicitar a inclusão de um CS e o SoS-Admin apenas aprovaria a requisição.
2	Melhorar o design e correção de <i>bugs</i> da ferramenta.
3	Adicionar um campo “saúde” do CS para que tanto o SoS-Admin quanto o SoS-User possa verificar o desempenho do CS.
4	Colocar os CS's em ordem alfabética ou agrupar por domínio para facilitar a identificação dos CS's na lista disponível.
5	Descentralizar o trabalho do papel do SoS-Admin. Inserir um botão voltar e atualizar.

Fonte: Dados da pesquisa.

Na Tabela 13, são apresentadas as dificuldades de quatro usuários. Dentre elas pode-se

destacar algumas dificuldades intermitentes do simulador, no momento de execução, em paralelo por todos os oito participantes. Disponibilizou-se um servidor local do banco de dados do simulador, e todos os participantes utilizam ao mesmo tempo esse servidor. Isso fez com que o simulador apresentasse alguns erros pontuais, mas que não atrapalharam a realização do estudo qualitativo.

Tabela 13 – Dificuldades Encontradas pelos Usuários no papel de SoS-Admin

	Dificuldades
1	Botões duplicados e atraso de resposta confundem.
2	Não tive dificuldades no funcionamento do simulador, mas poderia disponibilizar um pequeno tutorial ou caminho mais claro das atividades básicas do simulador.
3	Dificuldades nas operações gráficas e de banco, visto que por duas vezes os dados desapareceram.
4	A única dificuldade foi visualizar os dados de forma atualizada.

Fonte: Dados da pesquisa.

5.5 Discussão dos Resultados

A partir dos resultados obtidos na execução do estudo qualitativo, pode-se indicar possíveis respostas para as questões de pesquisa, propostas no planejamento do estudo, apresentadas na Seção 5.2. A RQ1 está relacionada à ideia geral deste trabalho, abordando a motivação, experiência do usuário nos conceitos abordados e à ideia de integração de vários CS's para compor o SoS-Reuse-Sim e prover um meio de potencializar a atividade de reúso. De forma geral, pode-se concluir que o SoS-Reuse-Sim foi considerado como uma ferramenta que potencializa a atividade de reúso de software. Os participantes fizeram comentários positivos quanto a ideia principal do simulador e consideraram-o útil para o desenvolvimento de software, visto que ao utilizar o simulador não precisa-se fazer busca manual por ativos de software reusáveis.

A RQ2 está relacionada a descobrir limitações de usabilidade e adequação funcional do simulador. De forma geral, pode-se concluir que o SoS-Reuse-Sim apresenta boa usabilidade, pois atingiu os objetivos específicos com eficiência e boa satisfação dos participantes. Em relação à adequação funcional, o SoS-Reuse-Sim ofereceu resultados corretos e se comportou de forma apropriada na realização das tarefas para os quais ele foi desenvolvido e projetado.

Durante a execução da avaliação do simulador e por meio dos questionários de *feedback* preenchidos pelos participantes, foram coletadas algumas sugestões de melhorias. Foi evidenciada a necessidade de alguns ajustes de design no simulador, como inserção de botões voltar e atualizar, reduzir o número de janelas e o ajuste no tamanho de algumas delas. Além disso, relatou-se a necessidade de um manual que explicasse as operações básicas dos dois papéis do simulador.

Conforme explicado anteriormente, dividiu-se o simulador em dois papéis de uso: SoS-Admin e SoS-User. Para cada um dos papéis, os usuários responderam formulários com diferentes perguntas. No entanto, a seguinte pergunta é comum em ambos os formulários: **O ambiente integrado é útil para apoiar o reúso de software. Dessa afirmação, qual sua opinião?**. Pode-se observar que houve diferentes porcentagens e em ambas o simulador foi bem avaliado. Dentre as cinco opções de resposta, apenas as duas melhores foram selecionadas: **Muito Útil ou Útil**. Na avaliação pelo módulo SoS-User, pode-se observar que 63% dos usuários consideraram que o simulador é Útil para apoiar o reúso de software e 37% consideraram como Muito Útil. Na avaliação pelo módulo SoS-Admin, pode-se observar que 50% dos usuários consideraram que o simulador é Útil para apoiar o reúso de software e 50% consideraram como Muito Útil. Na visão de administrador do simulador obteve-se uma porcentagem maior dos que classificaram como Muito Útil e pode-se dizer que eles tiveram uma satisfação maior ao utilizar o módulo SoS-Admin. Possivelmente, esse resultado se dá devido a esse módulo possuir maior responsabilidade no simulador. Por meio dele, é possível realizar muitas tarefas que vão compor o SoS-Reuse-Sim e qualquer outro SoS que venha a ser criado no simulador, portanto nesse caso o usuário desempenhou funções mais importantes e gerenciais e isso pode ter levado a uma melhor avaliação.

5.6 Ameaças à Validade

Uma questão fundamental em um estudo qualitativo é com relação à validade de seus resultados. Sendo assim, a validade dos resultados deve ser considerada na fase de planejamento para que os resultados sejam adequados ao grupo de interesse (WOHLIN *et al.*, 2012). Há quatro tipos de validade de resultados de estudos, conforme apresentado a seguir:

- **Validade de Conclusão:** está relacionada com a capacidade de inferir conclusões corretas a respeito do tratamento e o resultado do estudo. Sendo assim, é necessário considerar a confiabilidade das medidas. Os formulários podem ser uma ameaça desse tipo, pois eles podem ter sido mal elaborados e podem não ter permitido a coleta de informações relevantes. Para resolver esse problema, foram utilizados como base formulários de estudo qualitativos já executados com sucesso. Outro fator pode ser a descrição das perguntas dos três formulários. Procurou-se desenvolver as perguntas de forma clara e objetiva, porém, pode ter havido algum mal entendimento dos participantes e, consecutivamente, eles podem não ter respondido o que de fato foi requisitado na pergunta.
- **Validade Interna:** está relacionada à verdade aproximada sobre relações da causa e efeito, ou seja, com as influências que podem afetar a casualidade, das quais o pesquisador não possui conhecimento ou controle. Para isso, é necessário que haja uma maior atenção quanto aos participantes, com relação à seleção da população, divisão de classes, aspectos sociais, entre outros. Um ponto que pode ter influenciado os resultados foi a utilização de

alunos de pós-graduação do ICMC/USP, do mesmo laboratório de pesquisa do proponente deste estudo, como participantes do estudo. No entanto, os estudantes participaram do estudo de forma anônima. Além disso, eles não receberam nenhum tipo de recompensa ou favorecimento, para que não criassem expectativas e se comportassem de forma anormal durante o estudo.

- **Validade de Construção:** considera os relacionamentos entre a teoria e a observação, ou seja, está relacionada com influência dos tratamentos nos resultados observados. Um fator é que os participantes podem basear seu comportamento em suposições sobre a hipótese. A fim de evitar suposições, no treinamento aplicado buscou-se deixar claro o real propósito do estudo qualitativo e esclarecer as dúvidas sobre os conceitos abordados. Para não influenciar diretamente os resultados do estudo, o proponente do estudo fez apenas o papel de tutor. Esse é um procedimento que pode garantir a independência e imparcialidade da pesquisa realizada.
- **Validade Externa:** está relacionada com a generalização dos resultados obtidos no estudo para outros contextos. Desta forma deve-se observar a interação entre a seleção do sujeito e a interação do ambiente de execução. A experiência dos participantes pode ser considerada como uma ameaça à validade externa. No entanto, para amenizar esse problema foram realizados treinamentos para que os participantes tivessem um conhecimento mínimo para a execução do estudo. Além disso, o estudo foi realizado em um ambiente acadêmico e os resultados podem não ser semelhantes se esse estudo tivesse sido realizado com participantes da indústria. É importante ressaltar que o estudo foi realizado com alunos do meio acadêmico e não é possível generalizar os resultados, pois a amostra foi pequena e pode não representar a população alvo do simulador. Outro problema potencial é o fato dos mesmos participantes terem desempenhado o papel de SoS-User e SoS-Admin. Isso pode ter levado a opiniões similares sobre o sistema, por tratar-se da mesma pessoa. Em um ambiente real, esse comportamento poderia ser diferente.

5.7 Considerações Finais

Neste capítulo, apresentou-se a avaliação do simulador SoS-Reuse-Sim analisando sua adequação funcional e usabilidade. A avaliação do simulador foi realizada por meio de um estudo orientada a descoberta, com base em questionários subjetivos e objetivos. Os resultados obtidos no estudo indicam que o simulador apresenta uma boa qualidade, com relação à usabilidade e aplicação dos conceitos base abordados neste trabalho de mestrado, uma vez que a maioria dos participantes avaliaram o simulador de forma positiva. Com relação à usabilidade foram identificados apenas problemas pontuais, que não afetaram a utilização do simulador, nem mesmo os benefícios que ele pode oferecer.

CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

6.1 Visão Geral da Pesquisa

Entre os desafios da Engenharia de Software está a busca pela melhoria na produtividade e qualidade dos processos no desenvolvimento de software. Reúso é uma técnica essencial para superar esses desafios, por facilitar a reutilização de artefatos produzidos previamente. Então, é fundamental que haja apoio computacional ao reúso desde as fases de desenvolvimento. Essa não é uma tarefa fácil, visto a grande variedade de ferramentas e artefatos espalhados por toda parte, além do número elevado de plataformas de desenvolvimento, muitas vezes proprietárias ou de difícil acesso, o que dificulta sua integração. Para tentar superar esse desafio, este projeto de mestrado teve como objetivo propor uma abordagem para desenvolvimento de SoS Direcionado, com aplicação no SoS-Reúso, que representa a integração de ambientes de reúso de software, e pode facilitar/potencializar o reúso de diversos tipos de artefatos.

O SoS-Reúso aborda os conceitos de Reúso de Software, RAS, SoS e SOA, em especial os dois últimos têm potencial para solucionar vários dos problemas mencionados. Para avaliar a efetividade do SoS-Reúso, um simulador foi desenvolvido e avaliado em relação à usabilidade e adequação funcional, por meio de um estudo qualitativo. Os resultados obtidos indicaram que o simulador é adequado e apresenta uma boa usabilidade sob o ponto de vista dois diferentes papéis: SoS-Admin e SoS-User. Sendo assim, acredita-se que esse simulador poderá ser utilizado como uma ferramenta de apoio ao reúso de software e, consequentemente, potencializar a atividade de reúso.

6.2 Contribuições

A principal contribuição obtida com a realização deste projeto de mestrado foi o desenvolvimento de uma abordagem para desenvolvimento de SoS Direcionado - com aplicação no SoS-Reuse-Sim. Essa abordagem é o ponto de partida para o desenvolvimento de uma arquitetura que facilitará a integração de sistemas existentes rumo a SoS, no contexto do projeto MBA4SoSi (BRAGA, 2016). :

Contribuições secundárias também foram alcançadas, dentre as quais destacam-se:

- Identificação do estado da arte referente à integração de ambientes de reúso de software. O mapeamento sistemático realizado está disponível e pode ser consultada por interessados em realizar integrações semelhantes.
- Identificação do estado da arte referente à abordagens para integração em Sistemas-de-Sistemas. A revisão sistemática foi publicada (VARGAS; GOTTARDI; BRAGA, 2016) e pode ser utilizada para auxiliar pesquisadores interessados em SoSi.
- Implementação de um SoS do tipo direcionado, o SoS-Reuse-Sim, que concretiza os conceitos propostos na abordagem aqui apresentada. Pesquisadores na temática de SoS podem contar com esse exemplo concreto de implementação, que está disponível ¹ para utilização e aprimoramento.

6.3 Trabalhos Futuros

A partir do estudo experimental realizado no Capítulo 5, e por meio de *feedbacks* do proponente deste projeto de mestrado, foram identificados alguns aspectos do SoS-Reuse-Sim que podem ser investigados como trabalhos futuros. Dentre esses aspectos, destacam-se:

- Desenvolvimento de um manual de ajuda com os principais comandos do simulador.
- Desenvolvimento de uma funcionalidade que unifique em uma única janela todas as informações disponíveis dos ativos, ao invés de abrir seguidas janelas e sub janelas.
- Melhorar o mecanismo de busca pelo melhor ativo reusável, inserindo mais atributos e tornando a busca mais flexível.
- Desenvolvimento de uma funcionalidade que permita as usuários voltarem de janelas.
- Resolver problemas de usabilidade identificados no simulador.
- Desenvolvimento de um estudo qualitativo mais completo do simulador para avaliar diversos outros fatores, tais como: compatibilidade, confiabilidade e eficiência de desempenho.

¹ <https://www.dropbox.com/sh/lnibe0rkrrx6ots/AADC25rnqqluUUn_QRPWwg_Pa?dl=0>

Em relação a concepção do SoS-Reúso apresentada na Seção 4.3, podem-se destacar os seguintes trabalhos futuros:

- Desenvolvimento da versão completa do SoS-Reúso, removendo as limitações consideradas na implementação do SoS-Reuse-Sim.
- Validar a aplicabilidade da abordagem para desenvolver SoS em outros domínios de aplicação, de forma que utilize as mesmas ideias e conceitos. Um exemplo é o trabalho de mestrado em andamento (REIS, 2016), em que está sendo investigado um processo de desenvolvimento de aplicações da internet das coisas com base em SoS.

No escopo do projeto MBA4SoSi (BRAGA, 2016), um trabalho de mestrado encontra-se em desenvolvimento com o objetivo de facilitar o gerenciamento de SoS e seus constituintes (NASCIMENTO, 2017). Outros trabalhos estenderão e generalizarão conceitos estabelecidos no presente documento rumo a um *framework* baseado em modelos para apoio ao desenvolvimento de SoS, que inclui a definição de um processo que auxilie o SoS-Developer na tarefa de criação do SoS e seus comportamentos emergentes.

6.4 Produção Científica

A seguir, são apresentadas as publicações já elaboradas e as publicações em andamento.

6.4.1 Publicações Elaboradas

Como parte dos resultados obtidos na realização deste projeto de mestrado, destacam-se as seguintes publicações:

- Vargas, I. G.; Gottardi, Thiago; Braga, R. T. V. **Approaches for Integration in System of Systems: A Systematic Review**. In: 4th International Workshop on Software Engineering for Systems-of-Systems, 2016, Austin, Texas, USA. SESoS Proceedings. Austin, TX: ACM, 2016. v. 1. p. 32-38.
- Braga, R. T. V.; Vargas, I. G.; Gottardi, Thiago. **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), 2016, Maringá - PR, Brazil. CBSOFT Proceedings. São Paulo - SP, Brazil: Sociedade Brasileira de Computação, 2016. v. 1. p. 51-60.

6.4.2 Publicações em Andamento

Existem também alguns trabalhos que estão em andamento e serão submetidos para publicações, são eles:

- Vargas, I. G.; Braga, R. T. V. Integração de Ambientes de Reúso de Software - Um Mapeamento Sistemático.
- Vargas, I. G.; Braga, R. T. V. *SoS-Reuse: Leveraging reuse through the integration of reuse environments into a System-of-Systems*.
- Vargas, I. G.; Braga, R. T. V. Avaliação da Usabilidade e Adequação Funcional do SoS-Reuse-Sim.

REFERÊNCIAS

ALMEIDA, E. S.; ALVARO, A.; GARCIA, V. C.; JORGE; BURÉGIO, V. A.; NASCIMENTO, L. M.; LUCRÉDIO, D.; SILVIO. **C.R.U.I.S.E: Component Reuse in Software Engineering**. 1st. ed. Recife: C.E.S.A.R e-book, 2007. Citado nas páginas [90](#) e [93](#).

ALVARO, A.; LUCREDIO, D.; GARCIA, V.; PRADO, A. Francisco do; TREVELIN, L. Orion-re: a component-based software reengineering environment. In: **Proceedings 10th Working Conference on Reverse Engineering. WCRE 2003**. Victoria, B.C., Canada: IEEE Computer Society, 2003. p. 248–257. ISSN 1095-1350. Citado na página [152](#).

ANDREWS, Z.; PAYNE, R.; ROMANOVSKY, A.; DIDIER, A.; MOTA, A. Model-based development of fault tolerant systems of systems. In: **IEEE. Systems Conference (SysCon), 2013 IEEE International**. [S.l.], 2013. p. 356–363. Citado na página [48](#).

ASMIAÅ, EK, M.; KALNINS, A.; KALNINA, E.; AMBROZIEWICZ, A.; STRASZAK, T.; WOLTER, K. Comprehensive system for systematic case-driven software reuse. In: LEEUWEN, J. van; MUSCHOLL, A.; PELEG, D.; JAROSLAV, P.; RUMPE, B. (Ed.). **SOFSEM: Theory and Practice of Computer Science**. [S.l.]: Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, v. 5901). p. 697–708. ISBN 978-3-642-11265-2. Citado na página [44](#).

ASSMANN, U.; BARTHO, A.; BÜRGER, C.; CECH, S.; DEMUTH, B.; HEIDENREICH, F.; JOHANNES, J.; KAROL, S.; POLOWINSKI, J.; REIMANN, J. *et al.* Dropbox: the dresden open software toolbox. In: . [S.l.]: Springer Berlin Heidelberg, 2014. v. 13, n. 1, p. 133–169. Citado na página [152](#).

AZARNOUSH, H.; HORAN, B.; SRIDHAR, P.; MADNI, A. M.; JAMSHIDI, M. Towards optimization of a real-world robotic-sensor system of systems. In: **IEEE. Automation Congress, 2006. WAC'06. World**. [S.l.], 2006. p. 1–8. Citado na página [46](#).

BALDASSARRE, M. T.; BIANCHI, A.; CAIVANO, D.; VISAGGIO, G. An industrial case study on reuse oriented development. In: **Proceedings of the 21st IEEE International Conference on Software Maintenance**. Washington, DC, USA: IEEE Computer Society, 2005. (ICSM '05), p. 283–292. ISBN 0-7695-2368-4. Citado nas páginas [34](#) e [35](#).

BALDWIN, W. C.; SAUSER, B. Modeling the characteristics of system of systems. In: **2009 IEEE International Conference on System of Systems Engineering (SoSE)**. [S.l.: s.n.], 2009. p. 1–6. Citado na página [48](#).

BASSO, F.; WERNER, C.; OLIVEIRA, T. Towards facilities to introduce solutions for mde in development environments with reusable assets. In: **Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on**. Redwood City, California, US: [s.n.], 2014. p. 195–202. Citado nas páginas [61](#), [152](#) e [153](#).

BATISTA, T. Challenges for sos architecture description. In: **Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems**. New York, NY, USA: ACM, 2013. (SESoS '13), p. 35–37. ISBN 978-1-4503-2048-1. Citado na página [49](#).

- BOARDMAN, J.; SAUSER, B. System of systems - the meaning of of. In: **IEEE/SMC International Conference on System of Systems Engineering**. [S.l.: s.n.], 2006. p. 6 pp.–. Citado nas páginas [28](#) e [47](#).
- BOURQUE, P.; FAIRLEY, R. E. *et al.* **Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0**. [S.l.]: IEEE Computer Society Press, 2014. Citado na página [47](#).
- BRAGA, R.; CHAN, A. Peony: A web environment to support pattern-based development. In: **Eighth International Conference on Web Engineering. ICWE '08. New York, USA**. [S.l.: s.n.], 2008. p. 358–361. Citado na página [44](#).
- BRAGA, R. M. M.; WERNER, C. M. L.; MATTOSO, M. Odyssey-search: A multi-agent system for component information search and retrieval. **J. Syst. Softw.**, Elsevier Science Inc., New York, NY, USA, v. 79, n. 2, p. 204–215, fev. 2006. ISSN 0164-1212. Citado nas páginas [62](#), [152](#) e [154](#).
- BRAGA, R. T.; FELONI, D.; PACINI, K.; FILHO, D. S.; GOTTARDI, T. Aires: An architecture to improve software reuse. In: **Proceedings of the 15th International Conference on Software Reuse: Bridging with Social-Awareness - Volume 9679**. New York, NY, USA: Springer-Verlag New York, Inc., 2016. (ICSR 2016), p. 231–246. ISBN 978-3-319-35121-6. Citado nas páginas [27](#), [45](#) e [46](#).
- 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 [29](#), [126](#) e [127](#).
- 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 [79](#).
- BRAMBILLA, M.; CABOT, J.; WIMMER, M. **Model-Driven Software Engineering in Practice**. 1. ed. ,USA: Morgan & Claypool, 2012. Citado na página [29](#).
- BRYANS, J.; PAYNE, R.; HOLT, J.; PERRY, S. Semi-formal and formal interface specification for system of systems architecture. In: IEEE. **Systems Conference (SysCon), 2013 IEEE International**. [S.l.], 2013. p. 612–619. Citado nas páginas [49](#) e [50](#).
- CHAN, A. 2008. Peônia: um ambiente web para apoiar processos de desenvolvimento com utilização de padrões de software e requisitos de teste no projeto de aplicações. Dissertação de Mestrado sob orientação da Profa. Dra. Rosana T. Vaccare Braga. Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos-SP. Citado na página [44](#).
- CHAN, A.; CAGNIN, M. I.; MALDONADO, J. C.; BRAGA, R. T. V. An environment to support the use of software patterns and test requirements during application development (in portuguese). In: . In: 6th Latin American Conference on Pattern Languages of Programming. Porto de Galinhas - PE: [s.n.], 2007. p. 245–260. Citado na página [44](#).
- CHEN, W.; XIE, C.; SHI, J. A component-based model integrated framework for embedded software. In: WU, Z.; CHEN, C.; GUO, M.; BU, J. (Ed.). **Embedded Software and Systems**. [S.l.]: Springer Berlin Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3605). p. 563–569. ISBN 978-3-540-28128-3. Citado nas páginas [61](#), [152](#) e [154](#).

CHEN, Y.; CHENG, B. Facilitating an automated approach to architecture-based software reuse. In: **Proceedings 12th IEEE International Conference on Automated Software Engineering, Incline Village, NV**. [S.l.: s.n.], 1997. p. 238–245. Citado nas páginas [25](#), [26](#), [45](#) e [152](#).

CHU, W. C.; YANG, H. A formal method to software integration in reuse. In: **Proceedings of the 20th Conference on Computer Software and Applications**. Washington, DC, USA: IEEE Computer Society, 1996. (COMPSAC '96), p. 343–348. Citado na página [152](#).

COLOMBO, A.; KARNOUSKOS, S.; BANGEMANN, T. A system of systems view on collaborative industrial automation. In: **IEEE International Conference on Industrial Technology (ICIT)**. Cape Town, South Africa: [s.n.], 2013. p. 1968–1975. Citado na página [167](#).

COOPER, H.; HEDGES, L. V.; VALENTINE, J. C. **The handbook of research synthesis and meta-analysis**. [S.l.]: Russell Sage Foundation, 2009. Citado na página [168](#).

CORRADINI, F.; MARIANI, L.; MERELLI, E. An agent-based approach to tool integration. In: . Berlin, Heidelberg: Springer-Verlag, 2004. v. 6, n. 3, p. 231–244. ISSN 1433-2779. Citado na página [152](#).

CRUZES, D. S.; DYBÅ, T. Research synthesis in software engineering: A tertiary study. **Information and Software Technology**, Elsevier, v. 53, n. 5, p. 440–455, 2011. Citado na página [168](#).

DAHMAN, J. S.; JR, G. R.; LANE, J. A. **Systems engineering for capabilities**. [S.l.], 2008. Citado nas páginas [49](#) e [50](#).

DEMCHAK, B.; ERMAGAN, V.; FARCAS, E.; HUANG, T.-j.; KRÜGER, I. H.; MENARINI, M. A rich services approach to cocome. In: **The Common Component Modeling Example**. Berlin, Heidelberg: Springer-Verlag, 2008. p. 85–115. Citado na página [167](#).

DIMARIO, M.; BOARDMAN, J.; SAUSER, B. System of systems collaborative formation. **Systems Journal, IEEE**, v. 3, n. 3, p. 360–368, Sept 2009. ISSN 1932-8184. Citado na página [167](#).

DOD, D. o. D. 2008. System engineering guide for system-of-systems engineering. Technical Report, Department of Defense of United States, 2008. Citado na página [53](#).

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. **Inf. Softw. Technol.**, Butterworth-Heinemann, Newton, MA, USA, v. 50, n. 9-10, p. 833–859, ago. 2008. ISSN 0950-5849. Citado na página [165](#).

ERL, T. **Service-Oriented Architecture: Concepts, Technology, and Design**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005. ISBN 0131858580. Citado nas páginas [38](#) e [39](#).

ERMAGAN, V.; KRÜGER, I. H.; MENARINI, M. Aspect-oriented modeling approach to define routing in enterprise service bus architectures. In: **Proceedings of the 2008 International Workshop on Models in Software Engineering**. New York, NY, USA: ACM, 2008. (MiSE '08), p. 15–20. ISBN 978-1-60558-025-8. Citado na página [167](#).

EZRAN, M.; MORISIO, M.; TULLY, C. **Practical software reuse**. [S.l.]: Springer Science & Business Media, 2002. Citado na página [40](#).

FEILER, P.; SULLIVAN, K.; WALLNAU, K.; GABRIEL, R.; GOODENOUGH, J.; LINGER, R.; LONGSTAFF, T.; KAZMAN, R.; KLEIN, M.; NORTHROP, L.; SCHMIDT, D. **Ultra-Large-Scale Systems: The Software Challenge of the Future**. [S.l.]: Software Engineering Institute, Carnegie Mellon University, 2006. Citado na página 47.

FICHMAN, R. G.; KEMERER, C. F. Incentive compatibility and systematic software reuse. **Journal of Systems and Software**, Elsevier, v. 57, n. 1, p. 45–60, 2001. Citado na página 27.

FIRESMITH, D. **Profiling Systems Using the Defining Characteristics of Systems of Systems (SoS)**. Pittsburgh, PA, 2010. Disponível em: <<http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9269>>. Citado nas páginas 47 e 49.

FITZGERALD, J.; BRYANS, J.; PAYNE, R. A formal model-based approach to engineering systems-of-systems. In: _____. **Collaborative Networks in the Internet of Services: 13th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2012, Bournemouth, UK, October 1-3, 2012. Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 53–62. ISBN 978-3-642-32775-9. Citado nas páginas 47 e 48.

FRAKES, W.; TERRY, C. Software reuse: Metrics and models. In: . New York, NY, USA: ACM, 1996. v. 28, n. 2, p. 415–435. ISSN 0360-0300. Citado na página 25.

GHOSE, S.; DELAURENTIS, D. Development of an exploratory computer model for acquisition management of system-of-systems. **INCOSE International Symposium**, v. 19, n. 1, p. 36–51, 2009. ISSN 2334-5837. Citado na página 48.

GUTIERREZ-GARCIA, J.; RAMOS-CORCHADO, F.; KONING, J.-L. Obligations as constraints, descriptors, and linkers of open system of systems. In: **IEEE International Conference on System of Systems Engineering SoSE 2009**. Albuquerque, New Mexico, U.S.: [s.n.], 2009. p. 1–6. Citado na página 167.

HADDAD, H. Integrated collections: Approach to software component reuse. In: **Third International Conference on Information Technology: New Generations. ITNG 2006**. Las Vegas, Nevada, USA: [s.n.], 2006. p. 28–33. Citado nas páginas 62 e 155.

HALEY, C.; NUSEIBEH, B. Bridging requirements and architecture for systems of systems. In: **International Symposium on Information Technology. ITSIM 2008**. Kuala Lumpur, Malaysia: [s.n.], 2008. v. 4, p. 1–8. Citado na página 167.

HALFAWY, M.; FROESE, T. A component-based framework for integrated aec/fm project systems. In: **proceedings of the CSCE 2002 Conference of the Canadian Society for Civil Engineers, Montreal, Canada, June**. [S.l.: s.n.], 2002. p. 5–8. Citado na página 152.

HAN, J.; CHEN, P. Architecture support for system-of-systems evolution. In: HAN, Y.; TAI, S.; WIKARSKI, D. (Ed.). **Engineering and Deployment of Cooperative Information Systems**. [S.l.]: Springer Berlin Heidelberg, 2002, (Lecture Notes in Computer Science, v. 2480). p. 332–346. ISBN 978-3-540-44222-6. Citado nas páginas 51, 167 e 177.

HAREL, D. Statecharts: A visual formalism for complex systems. **Science of computer programming**, Elsevier, v. 8, n. 3, p. 231–274, 1987. Citado na página 75.

HASKINS, B.; STRIEGEL, J. Integration challenges of complex systems. **INCOSE International Symposium**, v. 16, n. 1, p. 1531–1545, 2006. ISSN 2334-5837. Citado nas páginas 167 e 170.

HAUSE, M. Sos for sos: A new paradigm for system of systems modeling. In: IEEE. **Aerospace Conference**. Big Sky, Montana, US, 2014. p. 1–12. Citado na página 74.

HOCHMÜLLER, E. **Process support for software reuse with AUGUSTA**. [S.l.: s.n.], 1995. Institute of University Klagenfurt, Austria. Citado nas páginas 62, 152 e 154.

HSIEH, M.; TEMPERO, E. Supporting software reuse by the individual programmer. In: **Proceedings of the 29th Australian Computer Science Conference - Vol. 48**. Darlinghurst, Australia: Australian Computer Society, 2006. (ACSC '06), p. 25–33. ISBN 1-920682-30-9. Citado nas páginas 61, 152 e 154.

HU, J.; HUANG, L.; CHANG, X.; CAO, B. A model driven service engineering approach to system of systems. In: **Systems Conference (SysCon), 2014 8th Annual IEEE**. [S.l.: s.n.], 2014. p. 136–145. Citado na página 167.

IRELAND, V.; CAVALLO, A.; OMAROVA, A.; OOI-SANCHES, Y.; RAPAPORT, B. Approaches in addressing system of systems. In: **System of Systems Engineering (SoSE), 2012 7th International Conference on**. [S.l.: s.n.], 2012. p. 380–385. Citado na página 167.

ISO/IEC. Measurement of system and software product quality. In: **Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE)**. Geneva: ISO, 2011. p. 19,50,51,53,54,55,58,67,85. Citado na página 104.

JAIN, R.; CHANDRASEKARAN, A.; ELIAS, G.; CLOUTIER, R. Exploring the impact of systems architecture and systems requirements on systems integration complexity. **Systems Journal, IEEE**, IEEE, v. 2, n. 2, p. 209–223, 2008. Citado na página 54.

JAMSHIDI, M. System of systems - innovations for 21st century. In: **Industrial and Information Systems, 2008. ICHIS 2008. IEEE Region 10 and the Third international Conference on**. [S.l.: s.n.], 2008. p. 6–7. Citado nas páginas 47 e 50.

JOVEL, J.; JAIN, R. Impact of identified causal factors to “system of systems” integration complexity from a defense industry perspective. **Global Journal of Flexible Systems Management GIFT**, Global Institute of Flexible Systems Management (GIFT), Delhi, India, v. 10, n. 4, p. 45, 2009. Citado nas páginas 54, 167, 180 e 181.

KARAPETROVIC, S.; WILLBORN, W. Integration of quality and environmental management systems. In: . [S.l.]: University of Manitoba, Winnipeg, Canada. MCB UP Ltd, 1998. v. 10, n. 3, p. 204–213. Citado na página 167.

KAZHAMIKIN, R.; PISTORE, M.; SANTUARI, L. Analysis of communication models in web service compositions. In: **Proceedings of the 15th International Conference on World Wide Web**. New York, NY, USA: ACM, 2006. (WWW '06), p. 267–276. ISBN 1-59593-323-9. Citado na página 39.

KEWLEY, R.; COOK, J.; GOERGER, N.; HENDERSON, D.; TEAGUE, E. Federated simulations for systems of systems integration. In: **Simulation Conference, 2008. WSC 2008. Winter**. [S.l.: s.n.], 2008. p. 1121–1129. Citado nas páginas 167 e 175.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004. Citado na página 157.

KITCHENHAM, B.; PRETORIUS, R.; BUDGEN, D.; BRERETON, O. P.; TURNER, M.; NIAZI, M.; LINKMAN, S. Systematic literature reviews in software engineering - a tertiary study. **Inf. Softw. Technol.**, Butterworth-Heinemann, Newton, MA, USA, v. 52, n. 8, p. 792–805, ago. 2010. ISSN 0950-5849. Citado na página [164](#).

KITCHENHAM, B. A. Systematic review in software engineering: Where we are and where we should be going. In: **Proceedings of the 2Nd International Workshop on Evidential Assessment of Software Technologies**. New York, NY, USA: ACM, 2012. (EAST '12), p. 1–2. ISBN 978-1-4503-1509-8. Citado na página [162](#).

KLEIN, J.; VLIET, H. van. A systematic review of system-of-systems architecture research. In: **Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures**. New York, NY, USA: ACM, 2013. (QoSA '13), p. 13–22. ISBN 978-1-4503-2126-6. Citado nas páginas [64](#), [65](#), [73](#), [158](#) e [183](#).

KRUEGER, C. W. Software reuse. In: . New York, NY, USA: ACM Comput. Surv., 1992. v. 24, n. 2, p. 131–183. ISSN 0360-0300. Citado nas páginas [34](#), [141](#) e [151](#).

KRUEGER, I.; 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 and Integration**. [S.l.: s.n.], 2006. p. 51–56. Citado nas páginas [167](#) e [176](#).

LANE, J. A. **What is a System of Systems and Why Should I Care?** 2013. Technical Report, University of Southern California. Citado na página [50](#).

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. ISSN 0740-7459. Citado na página [81](#).

LINTHICUM, D. S. **Next Generation Application Integration: From Simple Information to Web Services**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0201844567. Citado nas páginas [26](#), [28](#) e [55](#).

LOMBARDO, L.; BIAJIZ, M.; PRADO, A. D. A component's framework based on the mvc pattern for the integration of data and tools on the molecular biology domain. In: **2006 IEEE International Conference on Information Reuse and Integration**. [S.l.: s.n.], 2006. p. 389–394. Citado na página [152](#).

LOPEZ, D. Lessons learned from the front lines of the aerospace industry - balancing complexity and risk. In: **Conference on System of Systems Engineering, IEEE/SMC International**. [S.l.: s.n.], 2006. p. 5–14. Citado na página [47](#).

LÓPEZ, J.; BELLAS, F.; PAN, A.; MONTOTO, P. A component-based approach for engineering enterprise mashups. In: **9th International Conference on Web Engineering**. Berlin, Heidelberg: Springer-Verlag, 2009. (ICWE '9), p. 30–44. ISBN 978-3-642-02817-5. Citado nas páginas [61](#), [62](#), [152](#), [153](#) e [154](#).

LUNA, S.; LOPES, A. J.; TAO, H. Y. S.; ZAPATA, F.; PINEDA, R. Integration, verification, validation, test, and evaluation (ivvt&e) framework for system of systems (sos). In: DAGLI, C. H. (Ed.). **Complex Adaptive Systems**. Baltimore, Maryland, USA: Elsevier, 2013. (Procedia Computer Science, v. 20), p. 298–305. Citado nas páginas [167](#) e [181](#).

MADNI, A. M.; SIEVERS, M. System of systems integration: Key considerations and challenges. **Systems Engineering**, v. 17, n. 3, p. 330–347, 2014. ISSN 1520-6858. Citado nas páginas [28](#), [48](#), [50](#), [51](#), [52](#), [53](#), [55](#), [69](#), [74](#), [159](#), [167](#), [171](#), [182](#) e [183](#).

_____. Systems integration: Key perspectives, experiences, and challenges. **Systems Engineering**, Wiley Online Library, v. 17, n. 1, p. 37–51, 2014. ISSN 1520-6858. Citado na página [51](#).

MAHMOOD, S.; AHMED, M.; ALSHAYEB, M. Reuse environments for software artifacts: Analysis framework. In: IEEE. **12th International Conference on Computer and Information Science (ICIS)**, IEEE/ACIS. Niigata, Japan, 2013. p. 35–40. Citado nas páginas [44](#), [45](#) e [145](#).

MAIER, M. W. Architecting principles for systems-of-systems. In: **INCOSE International Symposium**. [S.l.]: John Wiley & Sons, Inc., 1998. v. 1, n. 4, p. 565–573. ISSN 1520-6858. Citado nas páginas [47](#), [48](#), [49](#), [50](#), [78](#), [141](#) e [172](#).

MALAKUTI, S. Detecting emergent interference in integration of multiple self-adaptive systems. In: **Proceedings of the 2014 European Conference on Software Architecture Workshops**. New York, USA: ACM, 2014. (ECSAW '14), p. 24:1–24:7. ISBN 978-1-4503-2778-7. Citado nas páginas [167](#) e [178](#).

MANE, M.; DELAURENTIS, D. Impact of programmatic system interdependencies on system-of-systems development. In: **IEEE International Conference on System of Systems Engineering. SoSE 2009**. Albuquerque, New Mexico, U.S.: [s.n.], 2009. p. 1–6. Citado nas páginas [167](#) e [178](#).

MASCENA, J. C. C. P.; MEIRA, S. R. d. L.; ALMEIDA, E. S. de; GARCIA, V. C. Towards an effective integrated reuse environment. In: **Proceedings of the 5th International Conference on Generative Programming and Component Engineering**. New York, NY, USA: ACM, 2006. (GPCE '06), p. 95–100. ISBN 1-59593-237-2. Citado nas páginas [43](#), [45](#), [60](#), [141](#), [152](#) e [153](#).

MAYK, I.; MADNI, A. M. **The role of ontology in system-of-systems acquisition**. 2006. Technical Report, Intelligent Systems Technology, Inc. Santa Monica, CA, USA. Citado na página [47](#).

MAZZUCHI, T.; ALBAKRI, G. System of systems engineering facilitates integration of large scale complex systems. **INCOSE International Symposium**, v. 21, n. 1, p. 811–855, 2011. ISSN 2334-5837. Citado nas páginas [53](#), [167](#) e [180](#).

MÜRK, O.; KABANOV, J. Aranea: Web framework construction and integration kit. In: **Proceedings of the 4th International Symposium on Principles and Practice of Programming in Java**. New York, NY, USA: ACM, 2006. (PPPJ '06), p. 163–172. ISBN 3-939352-05-5. Citado na página [152](#).

NAKAGAWA, E. Y.; GONÇALVES, M.; GUESSI, M.; OLIVEIRA, L. B. R.; OQUENDO, F. The state of the art and future perspectives in systems of systems software architectures. In: **Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems**. New York, USA: ACM, 2013. (SESoS '13), p. 13–20. ISBN 978-1-4503-2048-1. Citado nas páginas [49](#), [64](#), [65](#), [73](#), [158](#) e [183](#).

NAQVI, S. A.; CHITCHYAN, R.; ZSCHALER, S.; RASHID, A.; SÜDHOLT, M. Cross-document dependency analysis for system-of-system integration. In: **Proceedings of the 15th Monterey Conference on Foundations of Computer Software: Future Trends and Techniques for Development**. Berlin, Heidelberg: Springer-Verlag, 2010. (Monterey'08), p. 201–226. ISBN 3-642-12565-4, 978-3-642-12565-2. Citado nas páginas 167 e 176.

NASCIMENTO, D. de L. 2017. Uma abordagem baseada em serviços para gerenciamento de Sistemas-de-Sistemas e seus constituintes. Monografia de Qualificação - Dissertação de mestrado em andamento, Orientadora: Profa. Dra. Rosana T. Vaccare Braga, ICMC-USP. Citado na página 127.

NIELSEN, C. B.; LARSEN, P. G.; FITZGERALD, J.; WOODCOCK, J.; PELESKA, J. Model-based engineering of systems of systems. **Submitted to ACM Computing Surveys**, 2013. Citado na página 49.

NORMAN, D.; KURAS, M. Engineering complex systems. In: BRAHA, D.; MINAI, A. A.; BAR-YAM, Y. (Ed.). **Complex Engineered Systems**. [S.l.]: Springer Berlin Heidelberg, 2006, (Understanding Complex Systems). p. 206–245. ISBN 978-3-540-32831-5. Citado na página 47.

OMG. 2005. RAS - Reusable Asset Specification. Available at <<http://www.omg.org/spec/RAS/2.2/>>. Citado nas páginas 40, 41, 42, 43 e 141.

PALUDO, M.; REINEHR, S.; MALUCELLI, A.; BRUZON, L.; PINHO, P. Applying pattern structures to document and reuse components in component-based software engineering environments. In: **2011 IEEE International Conference on Information Reuse and Integration (IRI)**. Las Vegas, Nevada, USA: [s.n.], 2011. p. 378–383. Citado na página 152.

PAPAZOGLU, M. P.; HEUVEL, W.-J. Service oriented architectures: Approaches, technologies and research issues. **The VLDB Journal**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 16, n. 3, p. 389–415, jul. 2007. ISSN 1066-8888. Citado nas páginas 37, 38 e 141.

PARISI, C.; SAHIN, F.; JAMSHIDI, M. A discrete event xml based system of systems simulation for robust threat detection and integration. In: **IEEE International Conference on System of Systems Engineering, SoSE '08**. Singapore, Malaysia: [s.n.], 2008. p. 1–8. Citado nas páginas 48, 53, 167 e 179.

PARK, S.; PARK, S.; SUGUMARAN, V. Extending reusable asset specification to improve software reuse. In: **Proceedings of the 2007 ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2007. (SAC '07), p. 1473–1478. ISBN 1-59593-480-4. Citado na página 74.

PELTZ, C. Web services orchestration and choreography. In: . Los Alamitos, CA, USA: IEEE Computer Society Press, 2003. v. 36, n. 10, p. 46–52. ISSN 0018-9162. Citado nas páginas 39 e 40.

PÉREZ, J.; DÍAZ, J.; GARBAJOSA, J.; YAGÜE, A.; GONZALEZ, E.; LOPEZ-PEREA, M. Large-scale smart grids as system of systems. In: **Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems**. New York, NY, USA: ACM, 2013. (SESoS '13), p. 38–42. ISBN 978-1-4503-2048-1. Citado nas páginas 48, 49, 50 e 78.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATSSON, M. Systematic mapping studies in software engineering. In: **Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08**. Swinton, UK, UK: British Computer Society, 2008. p. 68–77. Citado na página 143.

PEÑA-MORA, F.; VADHAVKAR, S.; DIRISALA, S. K. Component-based software development for integrated construction management software applications. **AI EDAM**, Cambridge Univ Press, Massachusetts Avenue, Cambridge, USA, v. 15, p. 173–187, 4 2001. ISSN 1469-1760. Citado na página 152.

PONNALAGU, K.; NARENDRA, N.; GANGADHARAN, G. Integrated asset analysis framework for model-driven development of soa based solutions. In: PALLIS, G.; JMAIEL, M.; CHARFI, A.; GRAUPNER, S.; KARABULUT, Y.; GUINEA, S.; ROSENBERG, F.; SHENG, Q.; PAUTASSO, C.; MOKHTAR, S. B. (Ed.). **Service-Oriented Computing - ICSOC 2011 Workshops**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7221). p. 257–269. ISBN 978-3-642-31874-0. Citado nas páginas 61, 152 e 153.

POULIN, J. S.; CARUSO, J. M. A reuse metrics and return on investment model. In: **[1993] Proceedings Advances in Software Reuse**. [S.l.: s.n.], 1993. p. 152–166. Citado nas páginas 90 e 93.

RAMAMOORTHY, C.; CHANDRA, C.; KIM, H.; SHIM, Y.; VIJ, V. Systems integration: problems and approaches. In: **Proceedings of the Second International Conference on Systems Integration. ICSI '92**. Morristown, New Jersey, United States: [s.n.], 1992. p. 522–529. Citado na página 152.

RAMOS, M.; MASIERO, P.; BRAGA, R.; PENTEADO, R. Reengineering legacy systems towards system of systems development. In: **Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on**. [S.l.: s.n.], 2012. p. 624–630. Citado nas páginas 28 e 75.

_____. Reengineering legacy systems towards system of systems development. In: **Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on**. [S.l.: s.n.], 2012. p. 624–630. Citado na página 167.

_____. From software product lines to system of systems: Analysis of an evolution path. In: **Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on**. [S.l.: s.n.], 2013. p. 394–401. Citado nas páginas 28 e 75.

RAMOS, M.; MASIERO, P.; PENTEADO, R.; BRAGA, R. Extending statecharts to model system interactions. **Journal of Software Engineering Research and Development**, Springer Berlin Heidelberg, v. 3, n. 1, 2015. Citado na página 167.

RAMOS, M. A. **Bridging software engineering gaps towards system of systems development**. Tese (Doutorado) — ICMC-USP, 2014. Orientadora: Profa. Dra. Rosana T. V. Braga, 134 pgs. Citado nas páginas 28 e 75.

RAMOS, M. A.; MASIERO, P. C.; PENTEADO, R. A.; BRAGA, R. T. V. Extending statecharts to model systems' interactions. In: . São Carlos–SP–Brazil: Journal of Software Engineering Research and Development, 2015. Citado nas páginas 75 e 81.

- RANA, S.; SUN, C. Design and implementation of decs: distributed environment for cooperative software. In: **Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on**. British Columbia, Vancouver, Canada: [s.n.], 1995. v. 5, p. 4551–4556 vol.5. Citado nas páginas [62](#), [152](#) e [154](#).
- RANSOM, K. J.; MARLIN, C. D. Supporting software reuse within an integrated software development environment (position paper). In: . New York, NY, USA: ACM, 1995. v. 20, n. SI, p. 233–237. ISSN 0163-5948. Citado na página [152](#).
- REIS, D. F. dos. **Apoio ao processo de desenvolvimento de aplicações da internet das coisas com base em sistemas de sistemas**. 2016. Exame de Qualificação de Mestrado. ICMC-USP, Orientadora: Rosana T. Vaccare Braga. Citado na página [127](#).
- ROBINSON, W.; WOO, H. Finding reusable uml sequence diagrams automatically. **Software, IEEE**, v. 21, n. 5, p. 60–67, Sept 2004. ISSN 0740-7459. Citado na página [44](#).
- ROSELLÓ, E. G.; LADO, M. J.; MÉNDEZ, A. J.; DACOSTA, J. G.; COTA, M. P. A component framework for reusing a proprietary computer-aided engineering environment. In: . Oxford, UK: Elsevier Science Ltd., 2007. v. 38, n. 4, p. 256–266. ISSN 0965-9978. Citado nas páginas [61](#), [152](#) e [154](#).
- ROTHENBERG, J. Interoperability as a semantic cross-cutting concern. **Interoperabiliteit: Eerlijk zullen we alles delen**, Forum Standaardisatie Den Haag, 2008. Citado na página [53](#).
- RUBIN, J.; CHECHIK, M. Combining related products into product lines. In: LARA, J. de; ZISMAN, A. (Ed.). **Fundamental Approaches to Software Engineering**. [S.l.]: Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7212). p. 285–300. ISBN 978-3-642-28871-5. Citado na página [44](#).
- SAGE, A. P.; CUPPAN, C. D. On the systems engineering and management of systems of systems and federations of systems. In: . Amsterdam, The Netherlands, The Netherlands: IOS Press, 2001. v. 2, n. 4, p. 325–345. ISSN 1389-1995. Citado na página [33](#).
- SAWADSKY, N.; MURPHY, G. C. Fishtail: From task context to source code examples. In: **Proceedings of the 1st Workshop on Developing Tools As Plug-ins**. New York, NY, USA: ACM, 2011. (TOPI '11), p. 48–51. ISBN 978-1-4503-0599-0. Citado nas páginas [43](#), [61](#), [152](#) e [154](#).
- SCHMIDT, D. D. **Model-Driven Engineering**. [S.l.]: IEEE Computer, 2006. v. 39, n. 2. Citado na página [29](#).
- SHI, Y.; GANG, X.; SHENG-QIONG, Y.; HAO, C. Code component reuse supporting environment (ccrse) and its design strategy. **Wuhan University Journal of Natural Sciences**, Wuhan University, v. 8, n. 2, p. 363–367, 2003. ISSN 1007-1202. Citado nas páginas [62](#), [64](#), [152](#) e [154](#).
- SOLINGEN, R. V.; BASILI, V.; CALDIERA, G.; ROMBACH, H. D. Goal question metric (gqm) approach. **Encyclopedia of software engineering**, Wiley Online Library, 2002. Citado na página [105](#).
- SOMMERVILLE, I. **Engenharia de software**. 9. ed. [S.l.]: Pearson Education, 2011. 544 p. Citado nas páginas [25](#), [34](#), [35](#), [36](#) e [37](#).

SPINELLIS, D. Version control systems. In: . Dept. of Manage. Sci. & Technol., Athens ZUniv. of Econ. & Bus., Greece: IEEE Computer Society, 2005. v. 22, n. 5, p. 108–109. ISSN 0740-7459. Citado na página [44](#).

STOIAN, I.; STANCEL, E.; IGNAT, S.; BALOGH, S.; DANCEA, O. Federative scada consideration. In: **IEEE International Conference on Automation Quality and Testing Robotics (AQTR)**. Cluj-Napoca, Romania: [s.n.], 2010. v. 3, p. 1–6. Citado na página [167](#).

THIMMEGOWDA, N.; ALAM, O.; SCHÖTTLE, M.; ABED, W. A.; DI'MECO, T.; MARTELOTTO, L.; MUSSBACHER, G.; KIENZLE, J. Concern-driven software development with jucmnav and touchram. In: **Proceedings of the Demonstrations Track of the International Conference on Model Driven Engineering Languages and Systems (MODELS 2014)**. Valencia, Spain: [s.n.], 2014. v. 1255, p. 1–6. Citado nas páginas [61](#), [152](#) e [153](#).

TOMFORDE, S.; HAEHNER, J.; SEEBACH, H.; REIF, W.; SICK, B.; WACKER, A.; SCHOLTES, I. Engineering and mastering interwoven systems. In: **27th International Conference on Architecture of Computing Systems (ARCS)**. Luebeck, Deutschland, Germany: [s.n.], 2014. p. 1–8. Citado na página [167](#).

TRAN, H.; ZDUN, U.; DUSTDAR, S. Name-based view integration for enhancing the reusability in process-driven soas. In: MUEHLEN, M. zur; SU, J. (Ed.). **Business Process Management Workshops**. [S.l.]: Springer Berlin Heidelberg, 2011, (Lecture Notes in Business Information Processing, v. 66). p. 338–349. ISBN 978-3-642-20510-1. Citado na página [152](#).

UNDERSTANDING patterns for system of systems integration. In: 8TH International Conference on System of Systems Engineering (SoSE). Maui, Hawaii: [s.n.], 2013. p. 141–146. Citado nas páginas [53](#), [167](#), [179](#) e [180](#).

VANLEER, M. D.; JAIN, R. A framework to address the impact of system of systems integration using commercially off the shelf (cots) technology. In: **International Journal of System of Systems Engineering (IJSSE)**, Vol. 4, No. 1. [S.l.: s.n.], 2013. Citado na página [167](#).

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**. New York, NY, USA: ACM, 2016. (SESoS '16), p. 32–38. ISBN 978-1-4503-4172-1. Citado nas páginas [28](#), [30](#) e [126](#).

WERNER, C.; MANGAN, M.; MURTA, L.; PINHEIRO, R.; MATTOSO, M.; BRAGA, R.; BORGES, M. Odysseyshare: An environment for collaborative component-based development. In: **IEEE International Conference on Information Reuse and Integration, IRI 2003**. New York, NY, USA: Elsevier Science Inc., 2003. p. 61–68. Citado nas páginas [27](#), [62](#), [152](#) e [154](#).

_____. Odysseyshare: an environment for collaborative component-based development. In: **Information Reuse and Integration, 2003. IRI 2003. IEEE International Conference on**. [S.l.: s.n.], 2003. p. 61–68. Citado na página [44](#).

WOHLIN, C.; RUNESON, P.; HST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLN, A. **Experimentation in Software Engineering**. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434, 9783642290435. Citado nas páginas [31](#), [103](#) e [122](#).

WOJCIK, L.; HOFFMAN, K. Systems of systems engineering in the enterprise context: a unifying framework for dynamics. In: **System of Systems Engineering, 2006 IEEE/SMC International Conference on**. [S.l.: s.n.], 2006. p. 8 pp.–. Citado na página [47](#).

WU, Y.; WANG, X.; LIN, Y. An integration process model of enterprise information system families based on system of systems. In: **Advances in Swarm Intelligence**. [S.l.]: Springer, 2011. p. 479–485. Citado nas páginas 167 e 177.

ZE-MIN, L.; JING, Z.; ZHI-QIANG, L.; LI-LI, C.; JI-JUAN, X. Application of service-oriented architecture to equipment sos joint simulation. In: **System of Systems Engineering (SOSE), 2014 9th International Conference on**. [S.l.: s.n.], 2014. p. 119–123. Citado nas páginas 167 e 177.

ZEIGLER, B. P.; MITTAL, S. **Implementation of a Formal Standard for Interoperability in M&S/Systems of Systems Integration with DEVS/SOA**. 2008. Citado na página 51.

ZHANG, H.; KITCHENHAM, B.; PFAHL, D. Reflections on 10 years of software process simulation modeling: A systematic review. In: **International Conference on Making Globally Distributed Software Development a Success Story**. Berlin, Heidelberg: Springer-Verlag, 2008. (ICSP'08), p. 345–356. ISBN 3-540-79587-1, 978-3-540-79587-2. Citado na página 166.

GLOSSÁRIO

Ambientes de Reúso de Software: fornecem aos usuários um conjunto de funcionalidades que apoiam a atividade de reúso ([MASCENA et al., 2006](#))..

RAS: (do inglês, *Reusable Asset Specification*) é um conjunto de diretrizes e recomendações sobre a estrutura, o conteúdo e as descrições de ativos reusáveis de software ([OMG, 2005](#))..

Reúso de Software: é o processo de implementar ou atualizar sistemas de software usando ativos de software existentes ([KRUEGER, 1992](#))..

Sistemas-de-Sistemas: Um SoS é usualmente descrito como uma composição de sistemas, em que os elementos são projetados separadamente, descobertos, selecionados e integrados de modo a formar um sistema maior (mais complexo), emergindo um novo comportamento ou função não disponível em qualquer sistema individual que o compõe ([MAIER, 1998](#))..

SOA: (do inglês, *Service Oriented Architecture*) é um estilo arquitetural que permite desenvolver os sistemas com independência de linguagem de programação e flexibilidade quanto à plataforma, uma vez que funcionalidades oferecidas por serviços são definidas por meio de linguagens de descrição padronizadas ([PAPAZOGLU; HEUVEL, 2007](#))..

INTEGRAÇÃO DE AMBIENTES DE REÚSO DE SOFTWARE - UM MAPEAMENTO SISTEMÁTICO

Com o objetivo de reunir estudos primários que propõem técnicas/abordagens/ferramentas de integração de ambientes de reúso de software, desenvolveu-se um mapeamento sistemático da literatura. O mapeamento sistemático é um método de pesquisa que fornece uma visão geral de uma determinada área de pesquisa e permite identificar os tipos e quantidade de publicações existentes nesta área ([PETERSEN *et al.*, 2008](#)).

Nesta seção são apresentados os resultados do mapeamento sistemático, com o intuito de identificar e categorizar um conjunto de estudos que retratam integração entre ambientes de reúso. Visa-se produzir uma síntese que seja completa e imparcial, seguindo um processo bem definido. A condução do mapeamento sistemático dividiu-se em três etapas principais, que são: 1) Planejamento, 2) Condução do Mapeamento e 3) Sumarização dos Dados. Cada uma dessas etapas é definida a seguir.

1. Planejamento: Nesta etapa é identificada a necessidade de realização do mapeamento, de tal forma que seja definido um protocolo a ser seguido.
2. Condução: Nesta etapa o mapeamento sistemático é conduzido, identificando as fontes de busca relevantes e realizando busca por estudos nas máquinas de busca. Paralelamente, executa-se a extração de informações dos estudos selecionados que são adequadas ao escopo deste mapeamento.
3. Sumarização dos Dados: Síntese das informações coletadas dos estudos selecionados e publicação dos resultados obtidos.

Utilizou-se como ferramenta de apoio para condução deste mapeamento sistemático a ferramenta StArT¹, desenvolvida pelo Laboratório de Pesquisa em Engenharia de Software da Universidade Federal de São Carlos. Com a utilização dessa ferramenta foi possível executar todas as três etapas citadas anteriormente, que são detalhadas a seguir.

A.1 Planejamento

A.1.1 *Background: Justificativa para o Mapeamento Sistemático.*

O intuito deste mapeamento sistemático é investigar como tem sido feita a integração entre ambientes de reúso de software e atentar para os seguintes pontos: definições de ambientes de reúso de software, justificativa para a necessidade de integrar sistemas (em especial de reúso), ponderar as dificuldades e desafios encontrados. Sendo assim, a motivação deste mapeamento sistemático está voltada a evidenciar como ambientes de reúso de software tem sido integrados.

Na etapa de **Planejamento** foram definidas as questões de pesquisas. Portanto, o mapeamento sistemático objetivou responder às seguintes questões:

1. **RQ1. Como tem sido investigada a integração entre ambientes de reúso de software?**

Investigar se ferramentas de Engenharia de Software, especificamente ambientes de reúso, tem sido integradas e evidenciar como essa integração tem sido alcançada.

2. **RQ2. SoS tem sido apontado como uma possível solução para integrar ambientes de reúso?**

Evidenciar se o conceito de SoS tem sido proposto como uma possível solução para alcançar a integração entre ambientes de reúso de software

Com o intuito de melhor responder às questões de pesquisas, utilizou-se o seguinte **Formulário de Extração de Dados**. Todos esses dados foram extraídos de cada artigo selecionado para extração de dados e sua análise e sumarização é detalhada na Etapa 3 do mapeamento sistemático.

FE1- Existem definições de ambientes de reúso de software? Nesta questão objetivou-se evidenciar definições formais sobre ambientes de reúso de software.

FE 2- Como tem sido feita a integração entre ambientes de reúso? Nesta questão objetivou-se caracterizar, como e quais aspectos referentes à Engenharia de Software tem sido utilizados para obter a integração de ambientes de reúso e quando possível exemplificar a necessidade de integração.

¹ http://lapes.dc.ufscar.br/tools/start_tool

FE 3-Utilizou-se alguma ferramenta de apoio à integração? Nesta questão objetivou-se evidenciar se ferramentas de suporte tem sido utilizadas no processo de integração de ambientes de reúso.

FE 4- O artigo citou dificuldades e/ou desafios a serem superados? Nesta questão objetivou-se evidenciar possíveis problemas de integração e quais os desafios para a engenharia de software.

FE 5- SoS tem sido empregado como uma possível solução de integração entre ferramentas de Reúso? Nesta questão objetivou-se evidenciar se os conceitos de SoS estão sendo empregados quando ambientes de reúso de software são integrados.

Os principais fatores que compreendem o escopo desse estudo, utilizados como base para formulação das questões de pesquisa são descritos a seguir.

- **População:** compreende o conjunto de estudos primários ou secundários que tratam da integração de ambientes de reúso, sendo eles: acadêmicos/industriais, livros, artigos, teses e dissertações disponíveis nas bases de dados indexadas.
- **Intervenção:** corresponde ao que será observado no estudo, neste caso reúne estudos que retratam integração de ferramentas de engenharia de software, apresentando técnicas, métodos, ferramentas, taxonomias, *frameworks*, etc.
- **Controle:** corresponde ao conjunto de dados iniciais utilizados como base para o estudo. Neste caso, utilizou-se o estudo de [Mahmood, Ahmed e Alshayeb \(2013\)](#) como controle para este mapeamento.
- **Resultados:** compreende os resultados esperados, eles auxiliarão na resolução das questões de pesquisa. Os resultados observados neste estudo são definições, exemplos e tipos de técnicas/ferramentas utilizadas na integração.
- **Aplicação** corresponde aos papéis, tipos de profissionais ou áreas de aplicação que irão se beneficiar dos resultados do mapeamento sistemático. Neste caso são: desenvolvedores de software, engenheiros de software, arquitetos de software e profissionais acadêmicos que lidam Reúso de Software.

A.1.2 Estratégia para Seleção dos Estudos Primários

Para condução deste mapeamento, definiu-se uma estratégia de busca sistemática. A estratégia especifica onde e como executar as buscas, utilizando uma abordagem sistemática e rigorosa para identificar e selecionar os estudos primários. Neste estudo executou-se pesquisas por publicações em múltiplas fontes de dados, seguindo um protocolo predefinido.

As *strings* de pesquisa utilizadas nessa revisão foram construídas usando as seguintes estratégias:

- Identificar os principais termos baseados nas questões de pesquisa.
- Determinar e adicionar os sinônimos e termos relacionados, incorporando-os com o uso do operador **OR**.
- Indexar os principais termos usando o operador **AND**.
- Executar as *strings* de pesquisa de forma iterativa até uma fase suficiente de extração de dados ser alcançada.

Levou-se em consideração que Reúso de Software é um campo independente de pesquisa e prática, sendo assim focou-se em estudos que abordaram explicitamente este tópico. Os principais termos são “*support*”, “*integration*” e “*reusable component*”.

A.1.3 String de Pesquisa e Máquinas de Busca

As questões de pesquisa são utilizadas para construção da *string* de busca e tem o objetivo de identificar estudos primários relevantes. A partir das questões de pesquisa foram extraídos termos relevantes que correspondem aos três tópicos de interesse deste mapeamento:

1. *Software Reuse*: utilizou-se o termo *reusable component*, suas derivações e sinônimos, tais como: *reusable artifact*, *reusable assets*, *reusable items* e *reusable code*.
2. *Tool support*: utilizou-se o termo: *support*, suas derivações e sinônimos, tais como: *environment* e *tool*.
3. *Integration*: utilizou-se o termo: *integration*, suas derivações e sinônimos, tais como: *integrated* e *composition*.

Operadores *booleanos* **AND** e **OR** foram utilizados para a construção da *string* de busca, que é descrita de maneira geral a seguir: (sofre pequenas alterações para se adequar às restrições de cada base de busca indexada, não perdendo a lógica estrutural).

String de busca:*((integration OR integrated OR composition) AND (support OR environment OR tool) AND ({reusable component} OR {software reuse} OR {reusable artifact} OR {reusable assets} OR {reusable item} OR {reusable code}))*

Neste estudo foram consideradas bases de busca relevantes para a área de ciência da computação e engenharia de software, que indexam as principais conferências e periódicos dessas áreas. Na Tabela 14 são listadas as bases de dados utilizadas e seus respectivos endereços de acesso.

Após realizar a busca nas bases de dados selecionadas, diversos estudos foram retornados e analisados a fim de verificar sua adequação ao objetivo desta revisão sistemática. As próximas seções detalham como aconteceu a seleção de estudos.

Tabela 14 – Bases de Dados Utilizadas no Mapeamento Sistemático

Máquina de busca	Referência (URL)
ACM Digital library	http://dl.acm.org
IEEExplore	http://ieeexplore.ieee.org
Scopus	http://www.scopus.com
Springer	http://www.springer.com
Science Direct	http://www.sciencedirect.com
Web of Science	http://sub3.webofknowledge.com

Fonte: Dados da pesquisa.

A.2 Condução do Mapeamento

A.2.1 Critérios de Inclusão e Exclusão

A primeira seleção dos estudos realizou-se por meio da leitura do título e resumo. Se o estudo está relacionado às questões de pesquisa e satisfaz a pelo menos um critério de inclusão, ele é adicionado à revisão. Caso o estudo satisfaça a alguma das restrições impostas pelos critérios de exclusão, ele é excluído, visto que os critérios de exclusão tem prioridade sobre os de inclusão. Os critérios de inclusão e exclusão definidos para este estudo foram os seguintes:

- **Critério de Inclusão:**

Incluir estudos que:

CI1: abordam integração entre ferramentas de reúso de software.

CI2: abordam integração entre ambientes de desenvolvimento e alguma ferramenta de reúso.

CI3: descrevem ambientes de desenvolvimento que possuem módulo/*plugin* de reúso.

CI4: descrevem ambientes de desenvolvimento com funcionalidades de reúso oferecidas por meio de componente ou serviço.

- **Critérios de Exclusão:**

CE1: Excluir os estudos primários que retratam abordagens de integração de sistemas não relacionadas com Reúso de Software, ou seja, excluir estudos que retratam reúso de software e outras disciplinas fora do escopo de Integração.

CE2: Excluir os estudos primários que retratam apenas abordagens relacionadas aos conceitos de Reúso de Software.

CE3: Excluir estudos repetidos (detectados semi-automaticamente pela ferramenta de apoio Start, ou identificados manualmente posteriormente).

CE4: Excluir estudos com nomes diferentes (mas de mesmo assunto e mesmos autores), nesse caso o mais completo será considerado.

CE5: Excluir estudos que não estão escritos em inglês.

CE6: Excluir estudos incompletos e/ou menores de três páginas.

A seleção dos estudos primários realizou-se em duas etapas. Na primeira etapa, o aluno de mestrado avaliou todos os artigos retornados pelas bases de dados a fim de incluir apenas artigos relevantes à pesquisa, por meio da leitura paralela do Título e Resumo. A segunda etapa consiste de uma revisão em par com a orientadora desta proposta de mestrado, para avaliação e seleção final dos estudos primários obtidos na etapa anterior, de tal forma que o protocolo “test-retest” foi utilizado para validar as decisões de exclusão.

A.2.2 Procedimentos para Seleção dos Estudos

- **Análise por título e Análise por Abstract:** O título e *abstract* de cada artigo foram analisados de forma paralela e classificados como não relevantes ou relevantes. Sendo assim, com base nos critérios de inclusão e exclusão definidos anteriormente, o artigo foi excluído ou aceito.
- **Leitura transversal dos artigos:** Nesta etapa os artigos foram lidos de forma transversal, enfocando-se nas Seções de Introdução e Conclusão, na qual houve a classificação de alguns artigos para exclusão, por não ter respondido à nenhuma das RQ's.

Incluíram-se 47 estudos primários de um total de 1224 retornados pelas bases de dados. Na Figura 54, é ilustrada a quantidade de artigos obtidos quando executou-se a *string* de busca nas seis máquinas detalhadas anteriormente na Tabela 14.

Encontraram-se 330 artigos repetidos, ou seja, artigos que foram indexados por mais de uma máquina de busca. Ao avaliar os 894 artigos restantes após a exclusão dos repetidos, por Título e Resumo selecionaram-se 47 artigos para a etapa de leitura transversal e extração de dados. Em seguida, realizou-se uma revisão em par com a orientadora desta proposta de mestrado e excluíram-se 19 artigos, por terem sido classificados como fora do escopo da pesquisa. Sendo assim, restaram 28 artigos para extração de dados.

Observou-se que a maioria dos artigos foi excluída pelo critério de exclusão CE1. Isso aconteceu devido ao fato do conceito de Reúso de Software ser bastante difundido e conter uma literatura abrangente. De forma análoga, obtiveram-se artigos que envolvem o assunto integração de sistemas mas não necessariamente retrataram uma abordagem que detalha reúso. Optou-se por utilizar a *string* dessa maneira, pois uma restrição de nível maior na *string* deixou muitos trabalhos relevantes de fora da pesquisa, o que foi testado e observado em execuções prévias nas máquinas de busca.

Na Figura 55 é apresentada a quantidade de artigos retornados por cada base de dados e a porcentagem referente a cada uma delas em relação ao total. A maior parte dos artigos foi

Figura 54 – Processo de Seleção de Estudos Relevantes para o Mapeamento Sistemático

Fonte: Dados da pesquisa.

retornada pela máquina de busca Scopus, o que correspondeu à 43% do total de artigos obtidos no mapeamento sistemático. De maneira oposta, a menor parte dos artigos foi retornada pela máquina de busca ACM, o que correspondeu a 2% do total de artigos.

Na etapa final do processo de seleção dos estudos relevantes, realizou-se a leitura transversal de 47 artigos e selecionaram-se 28 artigos para extração de dados. Na Figura 56, é apresentada a relação da quantidade de artigos publicados por ano e o tipo de publicação (conferência ou *journal*) dos 28 artigos finais.

Figura 55 – Relação Quantidade por Máquina de Busca e Porcentagem do Mapeamento Sistemático

Fonte: Dados da pesquisa.

Figura 56 – Relação Quantidade de Artigos Publicados por Ano e Tipo de Publicação do Mapeamento Sistemático

Fonte: Dados da pesquisa.

A.3 Resultados - Mapeamento Sistemático

Como base para extração de dados e análise dos resultados consideraram-se os conceitos de Reúso de Software definidos por Krueger (1992), tais como definição, características, entre outros. Sendo assim, respondeu-se o formulário de extração de dados identificando quando possível os pontos em comum de cada artigo com os conceitos anteriormente definidos. Na sequência, apresentam-se as respostas obtidas para cada subquestão do formulário de extração de dados e ao final a resposta para cada questão de pesquisa.

A fim de ilustrar a frequência das palavras chaves dos 28 artigos selecionados para extração de dados, criou-se uma *wordcloud* com base em todas as *keywords* dos artigos. Na Figura 57, é apresentado o panorama final de ocorrência.

Figura 57 – Word Cloud para o Tema do MS

Fonte: Dados da pesquisa.

Quanto maior for o número de vezes que a palavra aparecer nas *keywords* de cada artigo, maior será a fonte utilizada para exibir essa palavra. Portanto, pôde-se observar que as palavras *software reuse*, *integration*, *reuseability* são algumas das que mais se destacaram e se tornaram intrínsecas quando o tópico deste mapeamento sistemático foi estudado.

Na Tabela 15, são apresentados os 28 artigos selecionados para extração de dados com seus respectivos anos de publicação.

Tabela 15 – Artigos Selecionados para Leitura Transversal com seus Respetivos Anos de Publicação

ID	Nome do Artigo	Ano
S1	<i>A component framework for reusing a proprietary computer-aided engineering environment</i> (ROSELLÓ <i>et al.</i> , 2007)	2007
S2	<i>A component-based approach for engineering enterprise mashups</i> (LÓPEZ <i>et al.</i> , 2009)	2009
S3	<i>A component-based framework for integrated AEC/FM project systems</i> (HALFAWY; FROESE, 2002)	2002
S4	<i>A component-based model integrated framework for embedded software</i> (CHEN; XIE; SHI, 2005)	2005
S5	<i>A component's framework based on the MVC pattern for the integration of data and tools on the molecular biology domain</i> (LOMBARDO; BIAJIZ; PRADO, 2006)	2006
S6	<i>A formal method to software integration in reuse</i> (CHU; YANG, 1996)	1996
S7	<i>An agent-based approach to tool integration</i> (CORRADINI; MARIANI; MERELLI, 2004)	2004
S8	<i>Applying pattern structures to document and reuse components in component-based software engineering environments</i> (PALUDO <i>et al.</i> , 2011)	2011
S9	<i>Aranea - Web framework construction and integration kit</i> (MÜRK; KABANOV, 2006)	2006
S10	<i>Code component reuse supporting environment (CCRSE) and its design strategy</i> (SHI <i>et al.</i> , 2003)	2003
S11	<i>Component-based software development for integrated construction management software applications</i> (PEÑA-MORA; VADHAVKAR; DIRISALA, 2001)	2001
S12	<i>Concern-driven software development with jUCMNav and TouchRAM</i> (THIMMEGOWDA <i>et al.</i> , 2014)	2014
S13	<i>Design and implementation of DECS: distributed environment for cooperative software</i> (RANA; SUN, 1995)	1995
S14	<i>DropsBox: the Dresden Open Software Toolbox</i> (ASSMANN <i>et al.</i> , 2014)	2012
S15	<i>Facilitating an automated approach to architecture-based software reuse</i> (CHEN; CHENG, 1997)	1997
S16	<i>Fishtail: From task context to source code examples</i> (SAWADSKY; MURPHY, 2011)	2001
S17	<i>Integrated asset analysis framework for model-driven development of SOA based solutions</i> (PONNALAGU; NARENDRA; GANGADHARAN, 2012)	2014
S18	<i>Integrated Collections: Approach to Software Component Reuse</i> (TRAN; ZDUN; DUSTDAR, 2011)	2011
S19	<i>Name-based view integration for enhancing the reusability in process-driven SOAs</i> (TRAN; ZDUN; DUSTDAR, 2011)	2011
S20	<i>Odyssey-Search: A multi-agent system for component information search and retrieval</i> (BRAGA; WERNER; MATTOSO, 2006)	2006
S21	<i>OdysseyShare: an environment for collaborative component-based development</i> (WERNER <i>et al.</i> , 2003a)	2003
S22	<i>Orion-RE: A Component-Based Software Reengineering Environment</i> (ALVARO <i>et al.</i> , 2003)	2003
S23	<i>Process support for software reuse with AUGUSTA</i> (HOCHMÜLLER, 1995)	1995
S24	<i>Supporting software reuse by the individual programmer</i> (HSIEH; TEMPERO, 2006)	2006
S25	<i>Supporting software reuse within an integrated software development environment</i> (RANSOM; MARLIN, 1995)	1995
S26	<i>Systems integration: problems and approaches</i> (RAMAMOORTHY <i>et al.</i> , 1992)	1992
S27	<i>Towards an Effective Integrated Reuse Environment</i> (MASCENA <i>et al.</i> , 2006)	2006
S28	<i>Towards facilities to introduce solutions for MDE in development environments with reusable assets</i> (BASSO; WERNER; OLIVEIRA, 2014)	2014

Fonte: Dados da pesquisa.

A.3.1 Resposta obtida para a subquestão FE 1- Existem definições de ambientes de reúso de software?

Nesta questão objetivou-se evidenciar definições sobre ambiente de reúso de software. Dentre os 28 artigos selecionados para extração de dados, não foram encontradas definições que respondem diretamente a esta pergunta. Contudo, conforme citado em [Mascena et al. \(2006\)](#), pôde-se coletar indícios de que são ambientes que fornecem aos usuários um conjunto de funcionalidades que apoiam a atividade de reúso. O conhecimento disponível na maioria dos casos é desestruturado e espalhado, sendo assim, um ambiente de reúso pode potencializar a atividade de reúso.

A.3.2 Resposta obtida para a subquestão FE 2- Como tem sido feita a integração entre ferramentas de reúso?

Nesta questão objetivou-se caracterizar, como e quais aspectos referentes à Engenharia de Software tem sido utilizados na integração de ferramentas de reúso e quando possível exemplificar a necessidade de integração.

Em **S28** ([BASSO; WERNER; OLIVEIRA, 2014](#)), utilizou-se a linguagem RAS (do inglês *Reusable Asset Specification*) para especificar os *assets* reusáveis e permitir a representação de seus elementos como tarefas e funções que instruem os usuários finais a adaptar os *assets* para uma necessidade específica. Após a especificação em RAS é possível descobrir inconsistências de requisitos e alcançar uma integração de *assets* reusáveis para um propósito específico.

Em **S12** ([THIMMEGOWDA et al., 2014](#)), afirma-se que o primeiro passo na integração de duas ferramentas é definir os conceitos de CDD (do inglês *Concern-Driven Development*) em um metamodelo, guiado por quatro passos: (1) *Concern* que representa o domínio de interesse do modelo; (2) Interface do *Concern* a interface de variação, a interface de personalização e a interface de uso; (3) Reúso do *Concern* um conceito que é usado para armazenar os recursos selecionados e (4) Características, com as associações para conectar os modelos que implementam a característica.

Em **S17** ([PONNALAGU; NARENDRA; GANGADHARAN, 2012](#)), foi proposto um processo de busca por potenciais *assets* reusáveis integrados em um repositório, a fim de minimizar o custo e o esforço por construir soluções baseadas em SOA, em vez de se desenvolver a partir do zero. Vale observar que nesse caso não se trata da integração propriamente dita, pois trabalha-se com um repositório integrado de *assets* reusáveis.

Em **S2** ([LÓPEZ et al., 2009](#)), foi proposta a criação de componentes para acessar fontes heterogêneas, um componente para combinar dados de diferentes fontes e outro componente para a construção de uma interface gráfica. O usuário constrói graficamente a aplicação *mashup*, selecionando, personalizando, e interligando componentes.

Em **S1** (ROSELLÓ *et al.*, 2007), foi proposta uma abordagem de integração de componentes com uma interface simples e fácil de usar orientada a objetos, que dá acesso a toda a funcionalidade do Matlab, que podem ser integrados a ferramentas de desenvolvimento, permitindo projetar seus próprios aplicativos de forma simples e intuitiva. Isso significa que o usuário não precisa nem usar Matlab nem aprendê-lo, e os desenvolvedores não necessitam usar diretamente o Matlab para o desenvolvimento de software.

Em **S24 e S16** (HSIEH; TEMPERO, 2006; SAWADSKY; MURPHY, 2011), foi proposto suporte ao reúso de código integrando *plugin* de desenvolvimento ao IDE Eclipse.

Em **S4** (CHEN; XIE; SHI, 2005), foi proposta a integração de componentes baseada em modelos reusáveis, os componentes são projetados e usados como blocos de construção para a integração, cada um dos quais é modelado de forma abstrata, com reações e ações de comunicação. Ele usa composição hierárquica para esconder os detalhes de implementação dos componentes, e mantém os componentes em um mesmo nível de hierarquia interagindo sob um modelo bem definido de integração.

Em **S10** (SHI *et al.*, 2003), foi proposto um ambiente de suporte ao reúso de código de componente, que integra ferramenta de desenvolvimento, ferramenta de simulação de execução e uma biblioteca de código de componente. Então, o ambiente pode fornecer suporte abrangente para muitas atividades de reúso de código de componente, tais como: desenvolvimento, composição, visualização, prototipação, consulta, navegação e transmissão.

Em **S2** (LÓPEZ *et al.*, 2009), foi proposto um *framework* que fornece um conjunto de componentes reusáveis para facilitar a integração de projetos distribuídos. Ferramentas de software legadas podem ser integradas por meio da implementação de adaptadores, representando um conjunto de componentes e interfaces por meio de uma camada de serviços de domínio comum.

Em **S13** (RANA; SUN, 1995), foi proposto um ambiente distribuído para software cooperativo (DECS). O objetivo do DECS é proporcionar uma plataforma flexível para integrar mecanismos variantes e libera os desenvolvedores de aspectos de protocolos de comunicação de baixo nível. Para auxiliar o processo de desenvolvimento, DECS fornece um conjunto de ferramentas de desenvolvimento e *frameworks* reusáveis. Essas ferramentas e estruturas destinam-se a apoiar as diversas atividades do desenvolvimento de software cooperativo.

Em **S23** (HOCHMÜLLER, 1995), foi proposto uma abordagem com foco no aspecto de reúso de componentes que permite a instanciação de programas a partir de componentes genéricos e a composição da aplicação completa baseado em uma estrutura de aplicação genérica. Permite-se então o desenvolvimento de software baseado em componente de integração, prototipagem e reutilização.

Em **S20 e S21** (BRAGA; WERNER; MATTOSO, 2006; WERNER *et al.*, 2003a), foi proposto um ambiente de desenvolvimento baseado em componentes. Os componentes são

desenvolvidos de acordo com um processo específico de Engenharia de Domínio, e são disponibilizados em “pacotes” que incluem diagramas, código fonte, descrições textuais e código binário, entre outros artefatos. Esses artefatos podem ser reusáveis em todas as fases de um desenvolvimento de aplicações, de acordo com um processo de Engenharia de Aplicação.

Em **S18** (HADDAD, 2006), foi proposto uma abordagem que define coleções de componentes reusáveis que estão integrados no ambiente de desenvolvimento. O conceito de interface genérica também é utilizado para definir um mecanismo de interface padrão e controlar a interface das coleções reusáveis no domínio de interesse. A metodologia de coleções reusáveis baseia-se na ideia de integrar componentes relacionados e comumente relacionados, também chamado de famílias de componentes. Com a abordagem de integração os componentes de software comuns são agrupados em conjuntos de modo que cada conjunto está associado a uma interface.

A.3.3 Resposta obtida para a subquestão FE 3- Utiliza-se alguma ferramenta de apoio à integração?

Nesta questão objetivou-se evidenciar se ferramentas suporte tem sido utilizadas para o processo de integração. Por meio da análise dos 28 artigos selecionados para extração de dados, pôde-se perceber que os poucos estudos que retrataram não utilizaram ferramentas de integração. Geralmente, a integração é realizada de maneira *ad-hoc*, sem suporte de ferramenta de integração.

A.3.4 Resposta obtida para a subquestão FE 4- O artigo citou dificuldades e/ou desafios a serem superados?

Nesta questão objetivou-se evidenciar possíveis problemas de integração e desafios para engenharia de software ao integrar ambientes de reúso de software.

- **Ferramentas proprietárias:** Quando a integração envolve ferramentas de reúso que são proprietárias, se o usuário não possuir a licença de utilização, não poderá ter acesso às funcionalidades da interface (**S1**).
- **Falta de ferramentas de integração:** as formas para integrar técnicas de reúso podem ter sido limitadas pela falta de ferramentas e técnicas adequadas que facilitem a tarefa, exceto de uma forma *ad-hoc* (**S25**).
- **Identificação de conflitos:** os conflitos não são fáceis de identificar, pois são normalmente escondidos na aplicação. A adaptação de componentes pode ser bastante trabalhosa, o que pode restringir a forma com que o componente será usado (**S15**).

- **Teste:** alguns problemas podem ocorrer no teste de integração de sistemas como: integração de software aplicativo, testes de integração e integração errada(S26).
- **Incompatibilidade:** As ferramentas envolvidas podem ter formatos incompatíveis de dados, interfaces diferentes, diferentes paradigmas de interação, diferentes níveis de acessibilidade para funcionalidades internas e diferentes direitos de acesso (S7).

A.3.5 Resposta obtida para a subquestão FE 5- SoS tem sido empregado como uma possível solução de integração entre ambientes de Reúso?

Nesta questão objetivou-se evidenciar se os conceitos de SoS estão sendo empregados quando ambientes de reúso de software são integrados. Porém, nenhum dos estudos apontou SoS como possível solução de integração entre ambientes de reúso.

A.4 Conclusão sobre as Questões de Pesquisa

A.4.1 RQ1. Como tem sido investigada a integração entre ambientes de reúso de software?

O intuito desta questão de pesquisa foi investigar se ferramentas de Engenharia de Software, especificamente ambientes de reúso, tem sido integradas e evidenciar como essa integração tem sido alcançada. Ao analisar os 28 artigos selecionados para extração de dados, é possível perceber que ainda são incipientes os estudos em direção à integração de ambientes de reúso. Existem algumas abordagens, apresentadas anteriormente, que estão focadas na maioria dos casos em proporcionar o reúso de código e ainda assim de forma isolada de outras ferramentas.

A.4.2 RQ2. SoS tem sido apontado como uma possível solução para integrar ambientes de reúso?

Esta questão teve por objetivo evidenciar se o conceito de SoS tem sido proposto como uma possível solução para alcançar a integração de ambientes de reúso de software. Contudo, ao realizar a análise dos 28 artigos selecionados para extração de dados, foi possível perceber que os conceitos de SoS não tem sido utilizados para integrar ambientes de reúso, o que fomenta a motivação desta proposta de mestrado por objetivar alcançar um ambiente integrado de reúso de software com características de SoS.

ABORDAGENS PARA INTEGRAÇÃO EM SISTEMAS-DE-SISTEMAS - UMA REVISÃO SISTEMÁTICA

Com objetivo de reunir estudos primários que propõem técnicas/abordagens/ferramentas de integração de sistemas no contexto de SoS, desenvolveu-se uma revisão sistemática da literatura. A revisão sistemática tem como objetivo apresentar uma avaliação criteriosa a respeito de um determinado tópico de pesquisa, fazendo uso de uma metodologia de revisão que seja confiável, rigorosa e que permita auditoria (KITCHENHAM, 2004).

Neste capítulo apresenta-se a revisão sistemática e visa-se produzir uma síntese que seja completa e imparcial, seguindo um processo bem definido. A condução da revisão sistemática é dividida em três etapas principais, que são: (1) Planejamento, (2) Execução e (3) Sumarização dos Dados. Cada uma dessas etapas é definida a seguir.

1. Planejamento: Nesta etapa é identificada a necessidade da realização da revisão, de tal forma que seja definido um protocolo de revisão.
2. Condução: Nesta etapa a revisão sistemática é conduzida, identificando as fontes relevantes e realizando busca por estudos nas máquinas de pesquisa. Paralelamente, executa-se a extração de informações, adequadas ao escopo desta revisão, dos estudos selecionados.
3. Sumarização dos Dados: Nesta etapa é feita a síntese das informações coletadas e publicação dos resultados obtidos.

Utilizou-se como ferramenta de apoio para condução desta revisão sistemática a ferramenta StArT¹, desenvolvida pelo Laboratório de Pesquisa em Engenharia de Software da

¹ http://lapes.dc.ufscar.br/tools/start_tool

Universidade Federal de São Carlos. Com a utilização dessa ferramenta, foi possível executar todas as três etapas citadas anteriormente, que são detalhadas a seguir.

B.1 Planejamento

B.1.1 Background: Justificativa para a Revisão Sistemática.

Os conceitos de SoS contém várias lacunas de pesquisa, entre elas sua aplicação e integração à ambientes de desenvolvimento de software. De acordo com as revisões sistemáticas apresentadas por Nakagawa *et al.* (2013) e Klein e Vliet (2013), não há evidências de que os conceitos de SoS estão sendo utilizados para integração de sistemas no contexto de reúso de software. Nesses estudos, é possível notar alguns domínios de aplicação em que SoS tem sido empregado: Militar, AeroEspacial, Aviônico, Automotivo e Comércio, entre outros. Contudo, os dois trabalhos citados anteriormente, são voltados para Arquitetura de Software no contexto de SoS e não possuem como foco de estudo a integração dos sistemas constituintes. Sendo assim, a revisão sistemática conduzida nesta proposta de mestrado, baseou-se na premissa de descobrir/evidenciar métodos de engenharia de software que auxiliam na integração entre os membros de um SoS.

Na etapa de **Planejamento** foram definidas as questões de pesquisa - (RQ - do inglês *Research Question*). Portanto, a revisão sistemática foi conduzida com o objetivo de responder às seguintes questões:

1. RQ1. Como tem sido investigada a integração entre os membros de um SoS?

Nesta questão objetivou-se evidenciar trabalhos que investigam as formas de integração no contexto de um SoS, identificando por exemplo, os problemas, dificuldades, abordagens, técnicas, soluções e vantagens para integrar membros de um SoS.

2. RQ2. Nesse tipo de estudo, tem sido utilizada alguma ferramenta que auxilia na integração dos sistemas constituintes?

Nesta questão objetivou-se caracterizar, quando possível, alguma ferramenta que auxilia na integração dos sistemas constituintes de um SoS.

Com o intuito de melhor responder às questões de pesquisa, utilizou-se o seguinte **Formulário de Extração de Dados**, contendo oito campos que foram extraídos de cada artigo selecionado para leitura completa. A análise e sumarização é detalhada na Etapa 3 da revisão sistemática.

FE1- Qual é a proposta do artigo em relação à integração de SoS? Nesta questão objetivou-se descobrir qual o tipo de estudo que foi realizado sobre integração de SoS e caracte-

rizar, por exemplo: como uma ontologia de integração, modelagem de integração, abordagem, framework, entre outros.

FE 1.1- Qual contribuição de integração de SoS foi abordada pelo estudo? Nesta questão objetivou-se caracterizar de forma detalhada como e quais aspectos referentes à engenharia de software tem sido utilizados para obter a integração de sistemas no contexto de SoS.

FE 2- Qual o domínio de aplicação que tem sido alvo na integração de SoS? Nesta questão objetivou-se descobrir em quais contextos a integração de SoS tem sido empregada, a fim de evidenciar a falta de trabalhos relacionados à integração de ferramentas no contexto de reuso de software.

FE 2.1- Qual o tipo de SoS foi utilizado? Com base nos diferentes tipos de SoS, citados no Capítulo 3, nesta questão objetivou-se caracterizar o tipo de SoS utilizado para integração dos sistemas constituintes.

FE 2.1.1- Quais características de SoS foram detalhadas? Com base nas diferentes características de SoS, citadas no Capítulo 3, nesta questão objetivou-se identificar as principais características utilizadas na integração dos sistemas constituintes de um SoS.

FE 2.2- Quais os problemas referentes à integração de SoS foram abordados? Nesta questão objetivou-se evidenciar os problemas apontados pelos estudos quando se realiza integração de sistemas no contexto de SoS.

FE 2.3- Quais as vantagens se têm quando utiliza os conceitos de SoS para integrar sistemas? Nesta questão objetivou-se evidenciar as vantagens e possíveis desafios que surgem ao utilizar SoS para integração de sistemas.

FE 3- O estudo apresentou alguma ferramenta de integração? Nesta questão objetivou-se especificar se o estudo aborda, alguma ferramenta utilizada para alcançar a integração.

Os principais fatores que compreendem o escopo deste estudo, usado como base para formulação das questões de pesquisas são descritos a seguir.

- **População:** compreende o conjunto de estudos primários ou secundários que retratam a integração de SoS, sendo eles: acadêmicos/industriais, livros, artigos, teses e dissertações disponíveis em bases de dados indexadas.
- **Intervenção:** corresponde ao que será observado no estudo, neste caso reúne estudos que retratam a integração de sistemas no contexto de SoS, apresentando técnicas, métodos, ferramentas, taxonomias, *frameworks*, entre outros.
- **Controle:** corresponde ao conjunto de dados iniciais utilizados como base para o estudo. Neste caso, utilizou-se o estudo de [Madni e Sievers \(2014a\)](#) como controle para esta revisão.

- **Resultados:** compreende os resultados que são esperados no estudo, e auxiliam na resolução das questões de pesquisa. Os resultados observados neste estudo são a quantidade e tipos de técnicas/ferramentas utilizadas na integração de sistemas no contexto de SoS.
- **Aplicação** corresponde aos papéis, tipos de profissionais ou áreas de aplicação que podem se beneficiar dos resultados da revisão sistemática. Neste caso são: desenvolvedores de software, engenheiros de software, arquitetos de software e profissionais acadêmicos que lidam com SoS.

B.1.2 Estratégia para Seleção dos Estudos Primários

Para condução deste estudo definiu-se uma estratégia de busca sistemática. A estratégia especifica onde e como executar as buscas, utilizando de uma abordagem sistemática e rigorosa para identificar e selecionar os estudos primários. Nesta RS executou-se uma pesquisa sistemática por estudos em múltiplas fontes de dados, seguindo um protocolo predefinido.

A *string* de busca utilizada nesta revisão foram construídas usando as seguintes estratégias:

- Identificar os principais termos baseados nas questões de pesquisa.
- Determinar e adicionar os sinônimos e termos relacionados, incorporando-os com o uso de **OR**.
- Indexar os principais termos usando **AND**.
- Executar a *string* de busca de forma iterativa até uma fase suficiente de extração de dados ser alcançada.

Levou-se em consideração que sistema de sistemas é uma área independente de pesquisa e prática, sendo assim focou-se em estudos que abordam explicitamente esse tópico. Os principais termos são “system of systems” e “integration”.

B.1.3 Conduzindo a busca por Estudos Primários

As questões de pesquisa foram utilizadas para construção da *string* de busca, a partir delas foram extraídos termos relevantes que correspondem aos dois tópicos de interesse desta revisão:

1. *System of Systems*: utilizou-se o termo `system of systems`, suas derivações e sinônimos, tais como: `systems of systems`, `Systems-of-systems`, `systems-of-systems`.
2. *Integration*: utilizou-se o termo: `integration`, suas derivações e sinônimos, tais como: `collaboration` e `composition`.

Operadores booleanos AND e OR foram utilizados para a construção da *string* de busca, descrita de maneira geral a seguir. Porém, necessita de pequenas alterações para se adequar às restrições de cada base de dados indexada, não perdendo a lógica estrutural.

String de busca:((*{system of systems}* OR *{systems of systems}* OR *{System -of-systems}* OR *{systems-of-systems}*) AND (*integration* OR *collaboration* OR *composition* OR *interoperability*)).

Neste estudo foram consideradas bases de dados relevantes para a área de ciência da computação e engenharia de software, que indexam as principais conferências e periódicos dessas áreas. Na Tabela 16, são listadas as bases de dados utilizadas e seus respectivos endereços de acesso.

Tabela 16 – Bases de Dados Utilizadas na Revisão Sistemática

Máquina de busca	Referência (URL)
ACM Digital library	http://dl.acm.org
IEEXplore	http://ieeexplore.ieee.org
Scopus	http://www.scopus.com
Springer	http://www.springer.com
Science Direct	http://www.sciencedirect.com
Web of Science	http://sub3.webofknowledge.com

Fonte: Elaborada pelo autor.

Após realizar a busca nas bases de dados selecionadas, diversos estudos foram retornados e analisados a fim de verificar sua adequação ao objetivo desta revisão sistemática.

B.1.4 Seleção dos Estudos Primários

A primeira seleção dos estudos realizou-se por meio da leitura do título e resumo. Se o estudo está relacionado às questões de pesquisa e satisfaz a pelo menos um critério de inclusão, ele é adicionado à revisão, caso o estudo satisfaça a alguma das restrições impostas pelos critérios de exclusão, o mesmo é eliminado, visto que os critérios de exclusão tem prioridade sobre os de inclusão. Os critérios de inclusão e exclusão definidos foram os seguintes:

- **Critério de Inclusão:**

CI1: O estudo primário retrata paralelamente o conceito de Sistema de Sistemas e o conceito de integração de sistemas.

- **Critérios de Exclusão:**

CE1: Excluir os estudos primários que retratam abordagens de integração de sistemas não relacionadas com SoS.

CE2: Excluir os estudos primários que retratam apenas abordagens relacionadas ao conceito de SoS.

CE3: Excluir estudos repetidos (detectados semi-automaticamente pela ferramenta de apoio Start) ou identificados manualmente.

CE4: Excluir estudos com nomes diferentes (mas de mesmo assunto e mesmos autores), o mais completo será considerado.

CE5: Excluir estudos que não estão escritos em inglês.

CE6: Excluir estudos incompletos e/ou menores de três páginas.

A seleção dos estudos primários realizou-se em duas etapas. Na primeira etapa o aluno de mestrado avaliou todos os artigos retornados pelas bases de dados a fim de incluir apenas artigos relevantes à pesquisa, por meio da leitura paralela do título e resumo. A segunda etapa consiste de uma revisão em par com a orientadora desta proposta de mestrado, para avaliação e seleção final dos estudos primários obtidos na etapa anterior, de tal forma que o protocolo “*test-retest*” foi utilizado para validar as decisões de exclusão, conforme definido em [Kitchenham \(2012\)](#).

B.2 Condução da Revisão

B.2.1 Procedimentos para Seleção dos Estudos

- **Análise por título e resumo:** O título e o resumo de cada artigo foram analisados de forma paralela e classificados como não relevantes ou relevantes. Sendo assim, com base nos critérios de inclusão e exclusão definidos anteriormente, o artigo foi excluído ou aceito.
- **Leitura completa dos artigos:** Neste procedimento os artigos foram lidos em sua totalidade, no qual houve a classificação de alguns artigos para exclusão, por não ter respondido à nenhuma das RQ's.

Incluíram-se 53 estudos primários de um total de 1398 retornados pelas bases de dados. Na Figura [58](#) é mostrada a quantidade de artigos obtidos quando executou-se a *string* de busca nas seis máquinas detalhadas anteriormente, na Tabela [16](#).

Encontraram-se 552 artigos repetidos, ou seja, artigos que foram indexados por mais de uma base de dados. Ao avaliar os 846 artigos restantes após a exclusão dos repetidos, selecionaram-se 53 artigos para a etapa de leitura completa. Desta quantidade, a versão completa de nove artigos não estavam disponíveis para download. Entrou-se em contato com os autores dos artigos, porém sem sucesso. Em seguida, realizou-se uma revisão em par com a orientadora desta proposta de mestrado e excluíram-se outros quatro artigos, por terem sido classificados como fora do escopo da revisão. Sendo assim, restaram 40 artigos para leitura completa.

Figura 58 – Processo de Seleção de Estudos Relevantes

Fonte: Elaborada pelo autor.

Na Figura 59 é apresentada a quantidade de artigos retornados por cada base de dados e a porcentagem referente a cada uma delas em relação ao total.

Observou-se que a maioria dos artigos foi excluída pelo critério de exclusão CE2. Isso aconteceu devido ao fato de que a *string* de busca utilizada permitiu o retorno de artigos que retratavam o conceito de SoS de uma forma geral, não apresentando, necessariamente, o foco na integração dos sistemas constituintes. De forma intrínseca, a obtenção de um SoS envolve integração de sistemas *stand-alone*, mas não necessariamente o estudo retratou uma abordagem que detalhava o processo de integração. Optou-se por utilizar a *string* dessa maneira, pois uma restrição de nível maior deixou muitos trabalhos relevantes de fora da pesquisa, o que foi testado

Figura 59 – Relação da Quantidade por Máquina de Busca e Porcentagem

Fonte: Elaborada pelo autor.

e observado em execuções prévias da *string* de busca. A maior parte dos artigos foi retornada pela máquina de busca Scopus, o que correspondeu à 47% do total de artigos obtidos na revisão sistemática. De maneira oposta, a menor parte dos artigos foi retornada pela máquina de busca Science Direct, o que correspondeu a 2% do total de artigos.

Na etapa final do processo de seleção dos estudos relevantes, realizou-se a leitura completa de 40 artigos e selecionaram-se 29 artigos para extração de dados. Na Figura 60 é apresentada a relação da quantidade de artigos publicados por ano e o tipo de publicação (conferência ou *journal*) dos 29 artigos finais.

B.2.2 Critérios de Avaliação de Qualidade

Definiram-se critérios de avaliação de qualidade com o intuito de avaliar os estudos primários. De acordo com Kitchenham *et al.* (2010), o propósito dos critérios de qualidade é indicar quais são os estudos mais relevantes para o objetivo da revisão sistemática em questão. Desta forma, é possível minimizar as limitações e melhorar a análise dos resultados. Avaliaram-se os estudos através de dois formulários. O primeiro deles, referiu-se à cobertura sobre as questões de pesquisa. Conforme detalhado anteriormente, no início da Seção B.1 definiu-se o formulário de extração de dados contendo oito campos a serem respondidos, sendo assim este primeiro formulário de avaliação de qualidade está relacionado aos respectivos campos. Portanto, os campos do formulário foram traduzidos em critérios quantitativos, para avaliar o quanto cada artigo responde favoravelmente às questões. Na Tabela 17 são apresentados os oito critérios e a

Figura 60 – Relação da Quantidade de Artigos Publicados por Ano e Tipo de Publicação

Fonte: Elaborada pelo autor.

descrição utilizada. O segundo formulário, referiu-se à cobertura sobre a qualidade do estudo. Na Tabela 18 são apresentados dez critérios de avaliação de qualidade que foram utilizados.

Tabela 17 – Formulário de Avaliação de Qualidade - Cobertura das Questões de Pesquisa

Nome	Artigo Exemplo 1	Score
ID	Critérios de Qualidade - Cobertura sobre Questões de Pesquisa	0/0.5/1
CQ1	O estudo descreve claramente o tipo de pesquisa conduzida?	
CQ1.1	O estudo descreve soluções de integração no contexto de SoS?	
CQ2	O estudo descreve o domínio de aplicação da integração de SoS?	
CQ2.1	O estudo descreve o tipo de SoS utilizado?	
CQ2.1.1	O estudo descreve características do SoS?	
CQ2.2	O estudo descreve algum problema de integração?	
CQ2.3	O estudo descreve vantagens quando SoS é aplicado para integração?	
CQ3	O estudo descreve ferramenta(s) de apoio ao processo de integração?	
Total:		

Fonte: Elaborada pelo autor.

De acordo com [Dybå e Dingsøyr \(2008\)](#), os principais aspectos para avaliar a qualidade do estudo são: **Rigor** - com o intuito de evidenciar se no estudo tem sido aplicada uma abordagem completa e adequada relacionada aos principais métodos de pesquisa. **Credibilidade** - com o intuito de evidenciar se os resultados foram bem apresentados e significativos. **Relevância** - com o intuito de evidenciar o quão útil são os resultados para a comunidade de pesquisa e/ou indústria.

Tabela 18 – Formulário de Avaliação de Qualidade - Cobertura Sobre o Estudo

Nome	Artigo Exemplo 1	Score
ID	Critérios de Qualidade - Cobertura sobre o Estudo	0/0.5/1
CS1	O estudo é baseado em pesquisa ou são “ <i>lessons learned</i> ”, meramente baseadas na opinião de especialistas ?	
CS2	O estudo descreve claramente o objetivo da pesquisa?	
CS3	O estudo descreve adequadamente o contexto no qual a pesquisa está inserida?	
CS3.1	O estudo apresenta uma visão geral sobre o estado da arte em que foi desenvolvido?	
CS4	O estudo fornece uma justificativa clara sobre a metodologia utilizada?	
CS5	O estudo utiliza métodos apropriados para coleta do dados utilizados?	
CS6	O estudo descreve claramente os resultados e conclusões justificadas?	
CS7	O estudo descreve validação sobre os resultados?	
CS8	O estudo discute limitações e credibilidade sobre os resultados?	
CS9	O estudo descreve trabalhos futuros e contribuições feitas?	
Total:		

Fonte: Elaborada pelo autor.

B.2.3 Sumarização dos Dados

Conforme apresentado nas Tabelas 17 e 18, a terceira coluna destinou-se ao *score* ou pontuação de avaliação. Atribuíram-se valores para CQ (Cobertura das Questões) e CS (Cobertura do Estudo), objetivando classificar os estudos de acordo com seu *score* total e também identificar os estudos mais relevantes.

As regras para pontuar o formulário basearam-se no que foi proposto por Zhang, Kitchenham e Pfahl (2008), em que cada critério de CQ e CS podem ser respondidos com “Yes”, “Partial” ou “No”. O *score* foi convertido a valores numéricos, como: (1 para “Yes”, 0.5 para “Partial” ou 0 para “No”), Tabela 19. Sendo assim, definiram-se as respostas para representar cada estudo com uma pontuação total, obtida pela soma individual dos critérios. Quando divergências ou dúvidas ocorreram, realizaram-se discussões com a orientadora desta proposta de mestrado para solucionar as inconsistências e então, determinar o *score* final.

Tabela 19 – Pontuação dos Critérios de Avaliação de Qualidade

Critérios	Pontuação do Formulário
Yes	1.0
Partial	0.5
No	0.0

Fonte: Elaborada pelo autor.

Estes critérios quando levados em consideração, de forma conjunta, fornecem um enfoque sobre os resultados de um estudo particular. Sendo assim, podem constituir uma contribuição valiosa ao tópico de pesquisa abordado e paralelamente evidenciar a confiabilidade dos resultados extraídos.

Na Tabela 20, é apresentado o *score* total de cada estudo selecionado para extração de dados. O *Score 1* representa o Formulário de Avaliação de Qualidade - Cobertura das Questões de Pesquisa e o *Score 2* representa o Formulário de Avaliação de Qualidade - Cobertura sobre o Estudo. Quando somados os dois valores é possível obter o *score* total para cada estudo e ressaltar aqueles melhores avaliados qualitativamente.

Tabela 20 – Artigos Selecionados para Leitura Completa com seus Respectivos *Score* de Avaliação de Qualidade

ID	Nome do Artigo	Score 1	Score 2	Total
S1	<i>Integration of Quality and environmental management systems</i> (KARAPETROVIC; WILLBORN, 1998)	3.5	3.5	7
S2	<i>Federated Simulations for Systems of Systems Integration</i> (KEWLEY <i>et al.</i> , 2008)	4.5	2	6.5
S3	<i>Rapid Systems of Systems Integration – Combining an Architecture Centric Approach with Enterprise Service Bus Infrastructure</i> (KRUEGER <i>et al.</i> , 2006)	5	3.5	8.5
S4	<i>Cross-Document Dependency Analysis for System-of-System Integration</i> (NAQVI <i>et al.</i> , 2010)	5	4.5	9.5
S5	<i>A System of Systems view on Collaborative Industrial Automation</i> (COLOMBO; KARNOUSKOS; BANGEMANN, 2013)	3	3	6
S6	<i>An Integration Process Model of Enterprise Information System Families Based on System of Systems</i> (WU; WANG; LIN, 2011)	5	4	9
S7	<i>Application of Service-Oriented Architecture to Equipment SoS Joint Simulation</i> (ZE-MIN <i>et al.</i> , 2014)	4	4	8
S8	<i>A Rich Services Approach to CoCoME</i> (DEMCHAK <i>et al.</i> , 2008)	4	6	10
S9	<i>Approaches in addressing System of Systems</i> (IRELAND <i>et al.</i> , 2012)	2.5	1.5	4
S10	<i>Architecture Support for System-of-Systems Evolution</i> (HAN; CHEN, 2002)	5	6.5	11.5
S11	<i>Aspect Oriented Modeling Approach to Define Routing in Enterprise Service Bus Architectures</i> (ERMAGAN; KRÜGER; MENARINI, 2008)	2	3.5	5.5
S12	<i>Detecting Emergent Interference in Integration of Multiple Self-Adaptive Systems</i> (MALAKUTI, 2014)	6	4.5	10.5
S13	<i>Bridging Requirements and Architecture for Systems of Systems</i> (HALEY; NUSEI-BEH, 2008)	3	6	9
S14	<i>Impact of Programmatic System Interdependencies on System-of-Systems Development</i> (MANE; DELAURENTIS, 2009)	5	5.5	10.5
S15	<i>A Discrete Event XML based System of Systems Simulation for Robust Threat Detection and Integration</i> (PARISI; SAHIN; JAMSHIDI, 2008)	4.5	4	8.5
S16	<i>Understanding Patterns for Systems of Systems Integration</i> (UNDERSTANDING... , 2013)	3.5	5	8.5
S17	<i>Federative SCADA Consideration</i> (STOIAN <i>et al.</i> , 2010)	2	3	5
S18	<i>Obligations as constrainers, descriptors, and linkers of Open System of Systems</i> (GUTIERREZ-GARCIA; RAMOS-CORCHADO; KONING, 2009)	3	5	8
S19	<i>Engineering and Mastering Interwoven Systems</i> (TOMFORDE <i>et al.</i> , 2014)	2	4	6
S20	<i>Impact of Identified Causal Factors to “System of Systems” Integration Complexity from a Defense Industry Perspective</i> (JOVEL; JAIN, 2009)	4.5	6	10.5
S21	<i>System of Systems Engineering Facilitates Integration of Large Scale Complex Systems</i> (MAZZUCHI; ALBAKRI, 2011)	4.5	6.5	11
S22	<i>Integration Challenges of Complex Systems</i> (HASKINS; STRIEGEL, 2006)	2	2.5	4.5
S23	<i>Integration, Verification, Validation, Test, and Evaluation (IVVT&E) Framework for System of Systems (SoS)</i> (LUNA <i>et al.</i> , 2013)	3	3	6
S24	<i>System of Systems Integration: Key Considerations and Challenges</i> (MADNI; SIEVERS, 2014a)	8	6	14
S25	<i>Reengineering legacy systems towards system of systems development</i> (RAMOS <i>et al.</i> , 2012b)	8	6	14
S26	<i>A framework to address the impact of system of systems integration using commercially off the shelf (COTS) technology</i> (VANLEER; JAIN, 2013)	8	6	14
S27	<i>Extending statecharts to model system interactions</i> (RAMOS <i>et al.</i> , 2015)	8	6	14
S28	<i>A Model Driven Service Engineering approach to System of Systems</i> (HU <i>et al.</i> , 2014)	8	6	14
S29	<i>System of Systems Collaborative Formation</i> (DIMARIO; BOARDMAN; SAUSER, 2009)	8	6	14

Fonte: Elaborada pelo autor.

B.2.4 Mapeamento de Dados

O mapeamento de dados é utilizado para representar um conjunto de métodos aplicados para resumir, integrar, combinar, e comparar os resultados de diferentes estudos relacionados a um tópico específico (COOPER; HEDGES; VALENTINE, 2009). Cruzes e Dybå (2011) argumentam que para se ter um nível de confiança aceitável sobre a evidência de um estudo particular, é necessário considerar seus resultados paralelamente com resultados de outros estudos que retratam as mesmas ou questões similares.

Os estudos com resposta “Sim” são aqueles que apresentaram informações relevantes sobre a questão. Estudos com respostas “Sim” foram marcados em negrito com o intuito de evidenciar aqueles estudos que melhor se enquadram ao escopo desta revisão. Os estudos denotados como “Parcialmente” são aqueles que apresentaram respostas inferidas, obtidas por meio de avaliações subjetivas, como por exemplo estudos que não apresentaram informação direta sobre soluções de integração mas discutiram problemas de integração. Os estudos com resposta “Não” são aqueles que não responderam a questão. Na Tabela 21, são apresentadas as respostas para cada campo do formulário de extração de dados.

Tabela 21 – Visão Geral das Respostas da Revisão Sistemática

CQ	Resposta	ID Artigos	Total
FE 1	Sim	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29	29
	Parcialmente		0
	Não		0
FE 1.1	Sim	S2, S3, S4, S6, S7, S8, S10, S12, S14, S15, S16, S20, S21, S23, S24, S26	16
	Parcialmente	S11, S13, S22, S25, S27, S28, S29	7
	Não	S1, S5, S9, S17, S18, S19	6
FE 2	Sim	S1, S2, S3, S4, S5, S7, S9, S12, S14, S20, S24	11
	Parcialmente		0
	Não	S6, S8, S10, S11, S13, S15, S16, S17, S18, S19, S21, S22, S23, S25, S26, S27, S28, S29	18
FE 2.1	Sim	S24	1
	Parcialmente		0
	Não	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21, S22, S23, S25, S26, S27, S28, S29	28
FE 2.1.1	Sim	S5, S6, S19, S24	4
	Parcialmente	S7, S18,	2
	Não	S1, S2, S3, S4, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S20, S21, S22, S23, S25, S26, S27, S28, S29	23
FE 2.2	Sim	S4, S9, S12, S14, S17, S20, S24	7
	Parcialmente	S1, S2, S3, S10, S13, S15, S21	7
	Não	S5, S6, S7, S8, S11, S16, S18, S19, S22, S23, S25, S26, S27, S28, S29	15
FE 2.3	Sim	S1, S24	2
	Parcialmente	S9, S16	2
	Não	S2, S3, S4, S5, S6, S7, S8, S10, S11, S12, S13, S14, S15, S17, S18, S19, S20, S21, S22, S23, S25, S26, S27, S28, S29	25
FE 3	Sim	S3, S4, S8, S10, S12, S15, S21	7
	Parcialmente		0
	Não	S1, S2, S5, S6, S7, S9, S11, S13, S14, S16, S17, S18, S19, S20, S22, S23, S24,, S25, S26, S27, S28, S29	22

Fonte: Elaborada pelo autor.

Relacionado à Integração de Sistema de Sistemas, foco desta revisão sistemática, vários estudos não fizeram contribuições significativas a respeito das questões de pesquisa, como

Tabela 22 – Respostas Obtidas para FE1 da Revisão Sistemática

	Proposta do Estudo				
	Modelagem	Framework	Ontologia	Abordagem	Padrão Arquitetural
Total	13	3	1	10	2
ID Artigo	S1, S2, S6, S7, S8, S11, S12, S14, S15, S18, S21, S22, S29	S20, S23, S26	S24	S4, S5, S9, S10, S13, S17, S19, S25, S27, S28	S3, S16

Fonte: Dados da pesquisa.

B.3.1 Resposta obtida para a subquestão FE1 - Qual é a proposta do artigo em relação à integração de SoS?

Nesta questão objetivou-se descobrir qual o tipo de contribuição feita sobre integração de sistemas no contexto de SoS. Na tabela 22 são apresentados os tipos identificados e os respectivos estudos, classificados em 5 categorias: modelagem, *framework*, ontologia, abordagem e padrão arquitetural.

De um total de 24 artigos, 12 retrataram algum tipo de modelagem. Embora não lidassem diretamente com o problema da integração dos sistemas constituintes do SoS, eles descreveram uma modelagem de integração como sendo uma parte importante do processo de desenvolvimento em geral, artigo S22 (HASKINS; STRIEGEL, 2006) é um exemplo desta situação, em que a integração requer um processo disciplinado que inicia na validação dos requisitos e passa pelas atividades de planejamento e verificação.

Dois artigos retrataram *frameworks* para SoS, que abordam vários componentes, processos, metodologias relacionados a SoS e também não lidam somente com o processo de integração, mas sim com padrões de interfaces arquiteturais e estratégias de teste. Um artigo retratou uma ontologia de integração de SoS, que visou padronizar as terminologias de SoSI e propôs definições claras sobre os conceitos que colaboram para a integração de sistemas, como por exemplo o termo interoperabilidade.

Sete artigos retrataram abordagens, apresentando um conjunto de teorias e conhecimentos sobre o tópico e investigando como chegar ao objetivo. Tais artigos foram utilizados para reforçar os conceitos relacionados a SoSI. Por fim, dois artigos retrataram algum tipo de padrão arquitetural para integração.

B.3.2 Resposta obtida para a subquestão FE 1.1 - Qual contribuição de integração de SoS foi abordada pelo estudo?

Nesta questão objetivou-se sinalizar quais são as principais contribuições de integração de SoS e evidenciar como e quais aspectos referentes à engenharia de software tem sido utilizados no processo de integração. Pôde-se observar que 62.5% dos artigos fizeram alguma contribuição.

Conforme mostrado na Tabela 21, 15 artigos responderam a esta questão, são eles: **S2, S3, S4, S6, S7, S8, S10, S12, S14, S15, S16, S20, S21, S23 e S24**. Na Seção B.4 são detalhados os estudos e suas contribuições a respeito de integração entre membros de SoS.

B.3.3 Resposta obtida para a subquestão FE 2 - Qual o domínio de aplicação que tem sido alvo no contexto de integração de SoS?

Nesta questão objetivou-se descobrir em quais domínios de aplicação a integração de SoS têm sido empregada. Conforme mostrado na Tabela 21, 11 artigos responderam completamente esta questão, o que corresponde a 45.83% do artigos.

Na Tabela 23, apresentam-se os principais domínios de aplicação quando SoSI é relacionado. Pôde-se observar que 54.13 % dos estudos não definiram o domínio de aplicação no qual a proposta está inserida.

Tabela 23 – Domínio de Aplicação

Domínio de Aplicação	Número de Estudos	ID Artigo
Nenhum domínio de Aplicação Específico	16	S6, S8, S10, S11, S13, S15, S16, S17, S18, S19, S21, S22, S23, S26, S27, S29
Defesa e Segurança Nacional	5	S3, S4, S14, S20, S28
Militar	1	S2
Automação Industrial	1	S5
Aeroespacial	1	S9
Gerenciamento de Qualidade	2	S1, S25
Sistema de Observação da Terra	1	S24
Simulação	1	S7
Auto Adaptação	1	S12
Total	29	

Fonte: Dados da pesquisa.

No decorrer de cada artigo, observou-se exemplos de SoS em domínios de aplicação diferente daqueles citados na Tabela 6. Contudo, estes exemplos não foram contabilizados porque não estavam diretamente relacionados ao problema de integração. Os domínios especificados na Tabela 6 foram exclusivamente identificados quanto à Integração de Sistema de Sistemas.

B.3.4 Resposta obtida para a subquestão FE 2.1 - Qual o tipo de SoS foi utilizado?

Dos 29 artigos selecionados para extração de dados, apenas um artigo definiu o tipo de SoS a ser utilizado para integração de sistemas. De acordo com Madni e Sievers (2014a), SoS Direcionados ou Reconhecidos são pré-especificados, que faz deles serem previsíveis e compatíveis com métodos tradicionais de validação e verificação. Por outro lado, SoS Virtuais

ou Colaborativos não são pré-especificados. Isto significa que não se pode ter prévia expectativa que caminhos e/ou conteúdos de informação são consistentes no SoS.

B.3.5 Resposta obtida para a subquestão FE 2.1.1 - Quais as características do SoS foram detalhadas?

Nesta questão objetivou-se identificar as principais características utilizadas na integração dos sistemas constituintes, ou seja, foi avaliado se o processo de integração garante as características essenciais de um SoS, tais como: independência operacional e gerencial, desenvolvimento evolucionário, comportamento emergente e distribuição geográfica. Porém, ao analisar sistematicamente os 29 artigos selecionados para extração de dados, observou-se que apenas quatro deles sinalizaram este tipo avaliação, o que corresponde a somente 16.66% dos artigos encontrados.

Especificamente, os artigos **S5, S6, S19, S24** demonstraram uma precaução maior, em relação às características, ao propor alguma solução de integração. Porém, estes estudos também não evidenciaram de maneira clara tais características no SoS final. Contudo, esses artigos foram classificados na Tabela 21, como “Sim” (responderam totalmente) porque além de detalhar as características básicas propostas por Maier (1998), retrataram alguma outra característica que não tenha sido citada pelos demais, como por exemplo: desenvolvimento adaptativo, conectividade, autonomia, diversidade, reconfiguração e princípios de modularidade.

B.3.6 Resposta obtida para a subquestão FE 2.2 - Quais problemas referentes à integração de SoS foram abordados?

Nesta questão objetivou-se sinalizar quais são as principais dificuldades quando se realiza integração de sistemas com características de SoS. Pôde-se observar que 58.33% dos artigos citaram algum problema/dificuldade de integração. Aqueles que detalharam mais de um problema, foram considerados como artigos que responderam totalmente a esta questão. Conforme mostrado na Tabela 21, sete artigos responderam completamente, o que corresponde a 29.16% dos artigos (**S4, S9, S12, S14, S17, S20, S24**).

A seguir são apresentados os problemas/dificuldades apontados pelos estudos:

1. **Gerencial:** O gerente de projeto deve saber claramente o conceito de sistemas e gerenciamento de sistemas para conseguir integrar sistemas individuais.
2. **Modelagem única:** tentar desenvolver um modelo único que represente o SoS como um todo é uma tarefa complicada, pois dificilmente um único modelo conseguiria representar todas as complexidades do SoS.

3. **Complexidade de interações:** Em SoSI um dos problemas mais desafiadores é lidar com a complexidade de interações entre as entidades do SoS resultante. A tarefa de integração de sistemas é capturar a lógica de negócio que lida com as interações dos diferentes sistemas.
4. **Complexidade dos sistemas constituintes:** os sistemas constituintes em um SoS são frequentemente complexos e problemas de integração podem surgir por causa da incapacidade de compreender completamente quais são as funcionalidades do sistema constituinte.
5. **Colaboração:** Mesmo no caso de uma integração pre-planejada, problemas imprevistos podem surgir pois sistemas constituintes estão sobre o controle de diferentes organizações. Estabelecer uma relação de sucesso entre os objetivos do SoS e os objetivos individuais de cada sistema não é uma tarefa trivial. A heterogeneidade dos domínios tem contribuído para uma baixa colaboração entre os sistemas e o desalinhamento dos objetivos entre os sistemas.
6. **Incompatibilidade de Interfaces:** sistemas evoluem durante sua vida útil para satisfazer novas demandas, porém os protocolos e interfaces que os sistemas definem são frequentemente não significativos para comunicação, colaboração e evolução. Além disso, cada sistema constituinte possui independência operacional e gerencial, logo cada equipe de desenvolvimento possui diferentes preferências para implementar seu sistema.
7. **Evolução:** há a dificuldade em evoluir os sistemas com um todo, de forma a colaborar para o SoS. Geralmente, não existe um processo de integração bem definido e a primeira responsabilidade é de otimizar o próprio desempenho do sistema individualmente, para benefício próprio e não para o objetivo geral do SoS, evidenciando problemas de sincronização e agregação.
8. **Constante atualização:** os sistemas constituintes estão em constante mudanças de software ou hardware, tais como atualizações ou adição de nova funcionalidade. Logo, aumenta-se a complexidade.
9. **Documentação:** a documentação dos sistemas legados não está sempre disponível, detalhada, coerente ou mesmo completa. É necessária documentação individual para entender quais são as funcionalidades técnicas presentes nos sistemas legados. Muitos engenheiros que trabalham com sistemas legados vão para novos projetos ou novas empresas, sem deixar seu conhecimento documentado.
10. **Roteiro:** falta de roteiros ou tutoriais que auxiliem engenheiros de software e/ou gerentes de projeto a usar/colocar em prática a integração de sistemas no contexto de SoS.

Tabela 24 – Vantagens Apontadas pelos Estudos da Revisão Sistemática

ID	Vantagem
1	Envolvimento mais abrangente de todas as partes interessadas.
2	Uso reduzido de múltiplos recursos.
3	Desenvolvimento de sistemas coerentes.
4	Forma harmonizada e unificada de resolver problemas.
5	Melhor custo-benefício.
6	Aumenta a flexibilidade e possibilidade para incluir outros sistemas.
7	Melhor desempenho operacional em conjunto.
8	Maior motivação da equipe.
9	Redução de custos e reengenharia mais eficiente.
10	Manutenção das características originais do sistema constituinte.
11	Independência operacional e gerencial.
12	Melhor escalabilidade.
13	Maior flexibilidade.
14	Melhor interoperabilidade.
15	Facilidade de manutenção.

Fonte: Dados da pesquisa.

B.3.7 Resposta obtida para a subquestão FE 2.3 - Quais as vantagens obtidas quando utiliza o conceito de SoS para integrar sistemas?

Nesta questão objetivou-se evidenciar quais são as principais vantagens que surgem ao utilizar SoS para integração de sistemas. Pôde-se observar que 16.66% dos artigos citaram alguma vantagem ao utilizar SoS para integrar. Aqueles que detalharam mais de uma vantagem, foram considerados como artigos que responderam totalmente a esta questão. Conforme mostrado na Tabela 21, dois artigos responderam completamente, o que corresponde a 8.33% dos artigos (S1, S4). Na Tabela 24, apresentam-se as vantagens apontadas pelos estudos:

B.3.8 Resposta obtida para a subquestão FE 3 - O estudo apresentou alguma ferramenta de integração?

Nesta questão objetivou-se especificar se o estudo aborda, alguma ferramenta que tenha sido utilizada para alcançar a integração. Observou-se que 29.16% dos artigos citaram alguma ferramenta para integrar membros de SoS. Conforme mostrado na Tabela 21, sete artigos responderam completamente, são eles: (S3, S4, S8, S10, S12, S15, S21).

1. **FireScrum:** Para guiar o esforço de integração adotaram-se processos de desenvolvimento baseados em Scrum S3.
2. **Mind mapping tool:** para criar uma representação intermediária do processo de negócio realizaram, por meio do processo de análise da arquitetura, refinamentos iterativos de *mind maps*. Eles, ajudaram a encontrar inconsistências nos documentos de processos de negócio S3.

3. **RDL - Requirements Description Language:** linguagem natural pode ser utilizada como base para análise de dependências e especificações de composições ou especificar restrições que resolvem inconsistências. Para isto, utilizou-se a ferramenta suporte para RDL, VMatriz NLP, que possibilita extrair informações relevantes a partir de um documento texto **S4**.
4. **Tool chain:** dividiu-se em estágios que abrangem todo o ciclo de vida de desenvolvimento: Elicitação de Requisitos, Especificação de Serviços e Projeto de Arquitetura, Implementação e Verificação. A Ferramenta SODA possibilita rastrear requisitos de sistemas orientados a serviços e gera documentos de arquitetura e requisitos. A ferramenta M2Code facilita a modelagem gráfica do serviço. A ferramenta Glue mapeia as máquinas de estado dos componentes para arquiteturas-alvo mais complicadas, por exemplo, componentes de instâncias replicadas **S8**.
5. **SENSE:** fornece suporte arquitetural para evolução do SoS, em particular a ferramenta utiliza XML. **S10**.
6. **UPPAAL:** ferramenta que facilita modelagem formal, simulação e modelo de verificação das propriedades comportamentais **S12**.
7. **DEVS:** *Discrete Event System Specification* é uma ferramenta que possibilita especificar os componentes de um sistema em um evento de simulação discreta. São especificados modelos básicos e como esse modelos são interconectados **S15**.
8. **M-Model:** é a ferramenta (a nível de modelagem) utilizada pelos Engenheiros de Sistema de Sistemas para integrar sistemas complexos com outros sistemas **S21**.

B.4 Contribuições dos artigos selecionados para extração de dados

1. **S2 (KEWLEY et al., 2008):** Apresenta uma abordagem que avalia os sistemas constituintes de um SoS em uma federação, ou seja, Sistema de Sistemas são vistos como uma Federação de Sistemas e sua integração é referenciada como interação. Sistemas de Combate do Exército criam complexas interações e a análise destas interações demandam um paradigma de simulação que seja reconfigurável e robusto. Sendo assim, podem ser chamados de simulações federadas e devem ser tão ágeis e interativas quanto o ambiente real em que os sistemas constituintes estariam sendo executados. Modelos únicos para análise de Sistema de Sistemas não são efetivos para representar toda a complexidade, pois o processo de desenvolvimento de modelos em grande escala é longo, difícil e em alguns casos uma abordagem alternativa é desenvolver os modelos dos sistemas constituintes

para alcançar o objetivo da atual integração. Modelos federados possuem as seguintes características:

- **Sistema de Troca de Informação:** a capacidade de passar informação útil entre os sistemas constituintes.
 - **Representação do espaço:** a capacidade de sistemas federados referenciar um ambiente correlacionado e compartilhado em que as entidades interagem.
 - **Representação da Entidade:** a capacidade de sistemas federados referenciar informações conceitualmente alinhadas sobre as entidades.
 - **Modelos:** modelos internos de cada sistema federado deve ser validado e coordenado com a federação de sistemas.
 - **Coleção de Dados:** a capacidade de coletar informação útil a partir da execução da simulação.
2. **S3 (KRUEGER *et al.*, 2006):** Apresenta uma abordagem de infraestrutura orientada a serviços em conjunto com um processo de desenvolvimento ágil da arquitetura do sistema, na qual menos detalhes de infraestrutura devem ser explicitamente modelados no nível arquitetural. Essa combinação é particularmente bem adequada para atender aos requisitos de integração. Em específico, ESB - *Enterprise Service Bus* tem emergido como um meio para cumprir muitos dos requisitos de flexibilidade e integração. O uso de processo ágil permite ter sempre *stakeholders* envolvidos no desenvolvimento de soluções de integração, assim como o uso de uma abordagem iterativa com reuniões mensais permite detectar conflitos de requisitos, expectativas inadequadas e novas restrições quando possível.
3. **S4 (NAQVI *et al.*, 2010):** O principal desafio para os interessados em um SoS é de compreender ou aprender sobre outros sistemas e influências externas. É necessário compreender melhor as necessidades das partes interessadas no SoS, assim como identificar e/ou resolver eventuais inconsistências entre os objetivos globais do SoS e dos seus sistemas constituintes. No artigo foram investigados três principais problemas: independência gerencial, incompatibilidade de interface e complexidade dos sistemas constituintes. Sendo assim, o proposto é: processamento de linguagem natural para ajudar a descobrir inconsistências nos documentos de requisitos dos sistemas constituintes. Qualquer análise de documento deve fornecer três principais pontos:
- Consulta de vários documentos de requisitos de forma que identifique diversos estilos de escrita e uso de diferentes termos para se referir aos mesmos conceitos, com o objetivo de reduzir incompatibilidade de interfaces.
 - Especificar restrições em vários documentos, a fim de capturar explicitamente estratégias de resolução de conflitos e, posteriormente, cenários de integração consistentes para o SoS.

- Suporte de automação, para auxiliar os engenheiros e outros interessados em consultas e especificações dos documentos de requisitos, durante a integração de SoS.
4. **S6 (WU; WANG; LIN, 2011):** A integração de sistemas muitas vezes ocorre em um ambiente dinâmico, onde novos sistemas são adicionados e sistemas legados são removidos, substituídos e/ou modificados em direção a um novo sistema. Ele deve ser visto como um processo dinâmico e favorece a evolução do SoS. O processo de integração proposto é iterativo, em que cada integração é vista como um sistema constituinte do SoS, que por sua vez, podem ser integrados em SoS maiores. Um SoS pode ser dividido logicamente em um nível superior e um nível inferior de sistemas constituintes. Sendo assim, os autores propuseram um modelo de processo de integração que pode ser dividido em sub-processos de integração de nível superior do SoS e integração em nível de sub-processos dos sistemas constituintes. Cada sub-processo é ainda sub-dividido em quatro fases e cada vez que um processo de integração está completo, ele aguarda a próxima necessidade de evolução do SoS.
 5. **S7 (ZE-MIN *et al.*, 2014):** Apresenta uma abordagem de simulação dos sistemas constituintes e integração. Com isso, os autores enfatizaram que SOA pode aumentar a flexibilidade da simulação de integração e suportar de forma eficiente a integração dos sistemas legados. SOA é particularmente aplicável para arquitetar sistema de sistemas. Em uma arquitetura orientada a serviços, todas as unidades funcionais devem ser definidas como serviços que tenham interfaces claras e bem definidas. No aspecto de simulação de integração em SOA, nota-se as seguintes características: (1) Utilização dos recursos existentes, (2) Capacidade para integração de sistemas complexos, (3) Maior flexibilidade, (4) Redução de custos e aumento da reutilização. SOA baseada em simulação inclui cinco camadas: (1) camada de simulação da aplicação, (2) camada de serviço, (3) camada de componente de simulação, (4) camada de serviço de dados e (5) camada de dados e modelos. Nesta hierarquia, quanto maior o nível, o mais próximo está da aplicação. Combinado com as características da tecnologia baseada em SOA, as aplicações podem ser integradas a sistemas transversais através dos serviços de dados, modelos e ferramentas. A fim de satisfazer as exigências e eficiência da simulação em execução, utilizou-se uma implementação *open-source* denominada TAO (The Orb ACE). Ao integrar os sistemas existentes na simulação, a reconstrução de interface devem ser feitas por um serviço requester e um serviço responder, ambos devem ser implementados como serviços SOA.
 6. **S10 (HAN; CHEN, 2002):** A evolução de sistemas necessita de apoio computacional para lidar com a crescente demanda e desafios em grande organizações. Em particular, interfaces são desenvolvidas para facilitar efetivamente a evolução do sistema, que é usualmente dirigida por dois principais fatores: mudanças de negócio e avanços tecnológicos. Uma questão chave no gerenciamento da evolução de um SoS é manter a identidade e autonomia dos sistemas constituintes ao apoiar a integração no sistema geral da organiza-

ção. O conhecimento necessário sobre um sistema constituinte para alcançar a integração com outros sistemas deve ser abstrato, captando apenas as suas características observáveis externamente. Este conhecimento abstrato do sistema constituinte é a base para a construção das arquiteturas de sistemas. A composição ou integração de grandes sistemas a partir de sistemas novos ou já existentes envolve interconectar os sistemas constituintes sob uma arquitetura particular. Nota-se que as arquiteturas dos sistemas constituintes não devem impor uma regra no processo de composição do sistema. Em vez disso, elas devem representar as características dos sistemas constituintes no processo de integração. A evolução de sistema pode fazer que em um determinado momento um sistema constituinte precise ser substituído por uma nova versão e até mesmo um novo sistema. Nesta situação, é preciso ter certeza que as novas funções do sistema constituinte ou versão funcionam corretamente no lugar do sistema antigo. A disponibilidade de informação da interface arquitetural do sistema antigo e da arquitetura do sistema faz com que a substituição seja mais fácil. Isso poderia ser feito por demonstrar a equivalência arquitetural de interface entre os sistemas ou mediante a realização de uma análise da interoperabilidade do sistema com o sistema constituinte antigo que está sendo substituído por um novo.

7. **S12 (MALAKUTI, 2014):** Apresenta uma abordagem de SAS (do inglês *Self-Adaptive Systems*), que são modelados separadamente e verificados para assegurar o seu comportamento correto quando executado individualmente. Os modelos de SAS individuais são integrados um com o outro para formar os SoS-alvo. Se as propriedades individuais não se mantêm, pode haver interferência emergente entre vários SAS do SoS alvo. Detecção de interferência entre múltiplos SAS requer amplo conhecimento sobre o comportamento de cada SAS individualmente, as partes do sistema que se adaptam e os impactos de tais adaptações em outros SAS. Portanto, o apoio metodológico e ferramenta adequada devem ser fornecidos aos engenheiros de software. Modelagem e verificação formal podem ser consideradas como meios apropriados para esse fim, já que verificando-se os modelos é possível definir o comportamento individual de cada SAS, bem como o comportamento integrado de múltiplos SAS. Antes de investigar se há qualquer interferência indesejada emergente, devido à integração de múltiplos SAS, assegura-se que cada SAS tem um comportamento correto quando executado individualmente. Portanto, primeiro modela-se cada SAS individualmente, define-se suas características e verifica-se suas propriedades.
8. **S14 (MANE; DELAURENTIS, 2009):** A implementação e integração de diferentes tipos de sistemas se torna mais complexo pela natureza evolutiva do SoS. Embora muitos sistemas possam ser dependentes de outras entidades durante as fases de implementação e/ou integração, o processo para atingir o SoS-alvo muitas vezes não é controlado centralmente. Isto requer que os desenvolvedores dos sistemas constituintes tenham um incentivo para colaborar uns com os outros. Uma modelagem única que represente toda a complexidade do processo de integração de um SoS, em pouco tempo e esforço é impraticável. O mo-

delo conceitual proposto é baseado em 16 técnicas básicas de gerenciamento de SoS e a integração é uma delas.

9. **S15 (PARISI; SAHIN; JAMSHIDI, 2008):** A integração implica que cada sistema pode se comunicar e interagir com o SoS, independentemente das suas características de hardware e/ou software. Isso significa que eles precisam ter a capacidade de se comunicar sem problemas de compatibilidade. Para este efeito, um SoS necessita de uma linguagem comum entre os sistemas constituintes, sem ter uma linguagem comum, os constituintes do SoS não podem ser totalmente funcionais e o SoS pode não ser adaptativo. Foi proposto um quadro de simulação de SoS baseado em *Extensible Markup Language* (XML), desenvolvido a fim de envolver os dados provenientes de diferentes sistemas em uma forma comum. O XML é usado para descrever cada componente do SoS e os seus dados de uma forma unificada. Em um SoS, sistemas constituintes muitas vezes interagem uns com os outros por causa de interoperabilidade e toda a integração do SoS. Estas interações podem ser alcançadas através de uma comunicação eficaz entre os sistemas ponto-a-ponto ou através de um coordenador central num dado SoS. Uma vez que os sistemas dentro de um SoS são funcionalmente independentes, as interações entre sistemas são geralmente assíncronas por natureza.
10. **S16 (UNDERSTANDING..., 2013):** Para alcançar a cooperação entre os sistemas, precisa-se realizar alguma forma de integração. O problema de integração é abordado neste artigo e descreve-se um primeiro passo no sentido de apoiar os arquitetos de software na concepção da integração de sistemas de software para tornar-se um sistema de sistemas. Caracteriza-se o problema de integração de software e ilustra-se a amplitude de soluções em termos de padrões de integração. Para caracterizar o problema, primeiro precisa-se diferenciar duas situações fundamentalmente diferentes: a integração de um novo sistema em um SoS existente ou criação de um novo SoS. Em particular, a integração no contexto de SoS é muito mais ampla do que apenas a troca de informações. Antes de escolher um padrão de integração, o arquiteto de software deve saber qual forma de integração deve ser alcançada. Existem duas categorias de projeto que são essenciais para a seleção de padrões de integração para SoS: como os dados são compartilhados e como controle será gerenciado. No contexto de SoS, sistemas são incrementalmente integrados e não são construídos com o SoS em mente, o que pode reduzir de maneira significativa as opções de projeto de integração. Então, definiram-se três categorias de contexto de desenvolvimento:
 - **Greenfield:** situação quando não há implementações pre-existentes que restringem o projeto de integração. Então, qualquer abordagem de desenvolvimento que for escolhida pode ser implementada. Implementar *Greenfield* no contexto de SoS significa que não há restrições arquiteturais, a não ser aquelas que estão intrínsecas ao objetivo do SoS alvo.

Tabela 25 – Padrão Arquitetural

Atributo	Caraterização
Nome	Breve descrição
Escopo do SoS	Sistema de Sistemas, Sistemas
Contexto do Desenvolvimento	<i>Greenfield, Brownfield, Closed Source</i>
Propósito de Integração	<i>One-Directional, Bi-Directional</i>
LISI	Isolado, Conectado, Funcional, Domínio
Atributos PAID	Procedimentos, Aplicações, Infraestrutura e Dados
Nível de Integração	Troca de Informação, Interação de Comportamento Básico, Interação de Comportamento Complexo
Nível de Abstração de Dados	Estrutural, Semântico, Sintático
Nível de Integração de Dados	Transferência de Arquivos, Passagem de Mensagem, <i>Streams</i>
Estilo de Integração	<i>Send, Call, Call-Return</i>
Atributos de Qualidade da Integração	Confiabilidade, Segurança, Escalabilidade

Fonte: Adaptada de [Understanding... \(2013\)](#).

- **Brownfield:** situação quando implementação ou sistemas já existem, porém modificações podem ser feitas. Devido à considerações de custo, normalmente adaptações mínimas são feitas. No contexto de SoS, isso significa que todos os sistemas constituintes podem ser adaptados.
- **Closed Source:** situação quando já existe uma implementação e não se tem acesso a ela. Neste caso o cenário de projeto é significativamente limitado e pode-se utilizar adaptadores externos. A integração com outros sistemas pode ser alcançada ao criar uma interface que conecte os sistemas sem ter a necessidade de alterar o sistema constituinte limitado.

Na Tabela 25 é apresentado o padrão arquitetural de integração identificando os diferentes aspectos do sistema e seus níveis de integração.

11. **S20 (JOVEL; JAIN, 2009):** Quando um sistema é projetado para ser um sistema interoperável, a possibilidade de adicionar uma nova funcionalidade ou atualizações é alta e tem custo mais acessível, sem a necessidade de ter que começar do zero. Além disso, o ciclo de vida do sistema é estendido. Um dos fatores que adiciona complexidade à integração dos sistemas é a contínua mudança de software ou hardware. Na Tabela 26 é apresentado o nível de esforço de integração de sistemas. Quanto maior o nível de esforço maior será a complexidade de integração.
12. **S21 (MAZZUCHI; ALBAKRI, 2011):** Integrar sistemas complexos se torna cada vez mais necessário, e envolve planejamento e coordenação entre os diferentes sistemas constituintes, e permite que sistemas trabalhem juntos como um só. Foi proposto o M-Model, é uma abordagem de planejamento para alcançar a integração baseada no V-Model. Contudo, o M-Model exige que os objetivos do SoS estejam bem definidos. Este modelo é projetado para ser aplicado em sistemas legados complexos. Em geral, existem três casos

Tabela 26 – Nível de Esforço de Integração

Questão	Possível Solução	Fase	Nível de Esforço(1-10)
Falta de foco, nas futuras capacidades do sistema, durante a fase de desenvolvimento	Os líderes do projeto devem direcionar o foco para as potenciais capacidades	Requisitos	8
Interoperabilidade dos sistemas existentes com a nova tecnologia integrada, causada por constantes mudanças	O projeto do sistema necessita ser compatível com todas as fases do projeto	Projeto da Arquitetura	10
Compatibilidade do Sistema do Fornecedor	Compartilhamento de informação dos sistemas de fornecedores	Projeto da Arquitetura	1
Mudanças contínuas no sistema	Documentação Sincronizada	Integração	7.5
Melhorias do sistema causam problemas imprevisíveis em outros subsistemas	Tipo da arquitetura do sistema definida deve facilitar a adição de melhorias	Projeto da Arquitetura	3
Novos Engenheiros de Software não estão familiarizados com os sistemas legados	Documentação completa dos sistemas legados	Integração	10

Fonte: Adaptada de [Jovel e Jain \(2009\)](#).

que precisam ser integrados: (1) novos sistemas com novos sistemas, (2) sistemas antigos com novos sistemas, e (3) sistemas antigos com sistemas antigos. O M-Model proposto é composto por quatro estágios: (1) Gerenciamento de integração: referente a definição do problema, planejamento e análise dos requisitos. (2) Decomposição e Definição: utiliza a análise dos requisitos feita na fase anterior para definir o projeto arquitetural, definindo as interfaces funcionais (interna e externa) e projetando uma solução de integração. (3) Validação da Integração: envolve avaliação do processo de integração, se os sistemas estão interoperáveis e podem se comunicar conforme o planejamento de integração. (4) Engenharia de Transição: a fase final é manter um acompanhamento contínuo do sistema para verificar o impacto de qualquer tipo de comportamento inesperado. Portanto, a integração começa em um momento específico após várias tarefas de gestão (ou seja, alocando equipes, construindo o organograma, documentando e esclarecendo os requisitos), em seguida, aplica-se o processo de “reengenharia” ou “engenharia reversa” para chegar a um ponto que é comum em ambos os sistemas.

13. **S23 (LUNA *et al.*, 2013):** É comum em um SoS a complexidade aumentar e tornar-se difícil de manter, pois o número de sistemas, interconexões e protocolos de interface evoluem ao longo do tempo. Além disso, durante a evolução, cada um dos sistemas constituintes deve passar pelo processo de desenvolvimento e documentação, a documentação de sistemas legados é sempre um problema no desenvolvimento do SoS. As tradicionais técnicas de Integração, Verificação, Validação, Teste e Avaliação -IVVT&E precisam ser expandidas para o nível do SoS, a fim de garantir que ele não é afetado pelas atualizações ou alterações dos sistemas constituintes. Os dois componentes, do *framework* proposto, que mais de adequam ao objetivo desta revisão sistemática são: (1) Padrão de Interfaces:

Arquitetar o SoS em torno de um conjunto de normas, permitirá que os sistemas constituintes possam ser substituídos com mínimo esforço de rearquitar interfaces do sistema. Para este fim, é importante selecionar os padrões que são amplamente utilizados em vez utilizar protocolos e/ou procedimentos próprios de cada componente do sistema. A inclusão de padrões de interface arquiteturais podem ajudar na inclusão de novos recursos e funcionalidades nos sistemas constituintes do SoS-alvo. (2) Camadas de Interface do SoS: A presença de camadas de interface na estrutura da arquitetura, ajuda no sentido de facilitar a integração dos sistemas recém evoluídos ou modelar os efeitos de comportamentos emergentes.

14. **S24 (MADNI; SIEVERS, 2014a):** Neste artigo foram apresentados uma série de conceitos importantes para a integração de SoS, tais como: interoperabilidade, integração de sistemas, tipologia e características de SoS, SOA, SoSI, reúso de sistemas constituintes, entre outros. Tais conceitos enfatizam os desafios da Integração de Sistema de Sistemas (SoSI). Ela envolve interfaces que permitam interações entre os sistemas constituintes e possibilitam criar a capacidade emergente necessária para atingir os objetivos do SoS-alvo. A integração deve fazer parte do ciclo de vida de desenvolvimento global do SoS. Quando existem sistemas legados que precisam ser integrados, a complexidade é aumentada e quando há interferência humana inúmeros problemas podem surgir. Há outras considerações importantes que devem ser feitas quando um sistema é integrado a um SoS. Por exemplo, é provável que um certo grau de autonomia será sacrificado quando um sistema participa de um SoS. Mecanismos de integração são um dos principais motores de complexidade, pois eles precisam apoiar a interoperabilidade entre os sistemas constituintes que possuem comportamento dinâmico e podem entrar/sair a qualquer momento. Esse comportamento dinâmico ocorre com base na mudança de requisitos de cada missão. A forma e o rigor de SoSI está diretamente relacionada ao tipo do SoS, por exemplo, o não gerenciado (Virtual) é inerentemente mais difícil de integrar do que um SoS gerenciado (Direcionado).

SoS dos tipos Virtual e Colaborativo não são pré-especificados e normalmente, um SoS é construído e configurado dinamicamente, em seguida são reorganizados conforme necessário para satisfazer as necessidades das missões subsequentes. Em alguns casos, os sistemas, os comportamentos, e conectividade de um SoS são conhecidos somente após uma missão ter sido especificada. Em contrapartida, SoS Direcionado e Reconhecido, são pré-especificados tornando-os previsíveis e compatíveis com métodos tradicionais de validação e verificação.

A definição, gestão e avaliação de abordagens de integração de SoS são dificultadas pela falta de uma terminologia comum e definições claras. Para superar este problema, propôs-se no artigo uma ontologia de integração de SoS. Esta ontologia serve para: (1) Definir uma terminologia padrão para descrever os recursos de integração. (2) Definir

uma lista de assuntos importantes e influências do impacto da integração. (3) Permitir o desenvolvimento de modelos de integração que apoiam o raciocínio. (4) Acomodar todas as necessidades das partes interessadas e seus respectivos pontos de vista. Esta ontologia inclui tanto artefatos (documentação) quanto métricas (medições ou testes utilizados para avaliar o sucesso da integração). Os conceitos estão relacionados com o domínio através de relacionamentos (associações), tais como: SoSI hasStakeholders e SoSI hasIntegration-Resources.

B.5 Conclusão sobre Questões de Pesquisa

B.5.1 RQ1. Como tem sido investigada a integração entre os membros de um SoS?

De acordo com a análise sistemática dos 29 artigos selecionados para extração de dados, pôde-se observar que SoSI possui alta demanda e ainda é uma área de pesquisa que necessita de maiores contribuições, havendo diversos estudos que evidenciaram esse desafio em suas propostas de trabalho. Nos últimos anos, houveram contribuições significativas que agregaram informações relevantes ao estado da arte de SoS. Por exemplo, na Revisão Sistemática da Literatura de SoS com perspectivas arquiteturais proposta por [Nakagawa et al. \(2013\)](#), é possível observar que a partir do ano de 2003 as pesquisas aumentaram significativamente. Paralelamente, na Revisão Sistemática da Literatura de Arquiteturas de SoS proposta por [Klein e Vliet \(2013\)](#), é possível notar que as pesquisas aumentaram significativamente a partir do ano de 2006. Alguns artigos, por exemplo (S2, S3, S4, S8), em seus resumos percebe-se a seguinte frase: “*System of systems integration is a difficult engineering challenge*” ou “*System of systems integration is a fundamental engineering challenge to Software Engineering*”. Sendo assim, nota-se uma crescente preocupação dos pesquisadores no contexto de SoS. Porém, ainda carece de pesquisas e uma maior disseminação dos conceitos entre os pesquisadores da área.

Também é interessante notar que a integração entre membros de um SoS tem sido investigada por meio do uso de SOA (*Service-Oriented Architecture*). Um número considerável de publicações, aproximadamente 54.16%, têm explorado o uso de SOA. Por exemplo, em [Madni e Sievers \(2014a\)](#), os autores desenvolveram uma discussão detalhada sobre o uso de SOA em SoS e suas vantagens no reúso de software. Na Figura 62 é apresentado o panorama geral dos artigos que utilizaram ou citaram SOA no processo de integração.

B.5.2 RQ2. Nesse tipo de estudo, tem sido utilizada alguma ferramenta que auxilia na integração dos sistemas constituintes?

De acordo com a análise sistemática dos 29 artigos selecionados para extração de dados, pôde-se evidenciar que não foram encontradas ferramentas de integração de sistemas no

Figura 62 – Visão Geral dos Artigos que Utilizaram SOA

Fonte: Elaborada pelo autor.

contexto de SoS. As ferramentas encontradas, detalhadas na Seção [B.3.8](#), apoiam alguma fase do desenvolvimento, como por exemplo: modelagem da arquitetura, ferramenta de análise de inconsistências nos documentos de requisitos, entre outras. Com isso percebe-se que há uma falta de ferramentas que auxiliem na integração de sistemas no contexto de SoS.

INFORMAÇÕES COMPLEMENTARES

A seguir são apresentados os três questionários fque oram aplicados neste estudo qualitativo, são eles: (1) Questionário pessoal para caracterizar o perfil e o nível de experiência do participante em relação aos conceitos abordados. (2) Questionário sobre o papel de SoS-User para coletar informações do participante sob o ponto de vista de um usuário comum que esteja interessado em fazer reúso de software. (3) Questionário sobre o papel de SoS-Admin para coletar informações do participante sob o ponto de vista de um usuário com funções gerenciais e responsabilidades maiores para composição de um SoS. Cada participante do estudo qualitativo utilizou os dois diferentes papéis de usuário do simulador e respondeu aos três questionários. Esses questionários visaram coletar dados e evidências dos usuários a fim de responder as questões de pesquisa definidas anteriormente.

C.1 Questionário Inicial

- Escolaridade*;
- Qual é o seu conhecimento sobre SoS?*;
- Qual é o seu conhecimento sobre reúso de software?*;
- Qual seu conhecimento sobre RAS? *
- Você já reusou algum repositório que facilita o reúso de software?* Qual tipo?
- Caso já tenha atuado no desenvolvimento de software, indique quantos anos.

C.2 Questionário do SoS-User

- Você ficou satisfeito com a forma que foi feita a pesquisa/busca por ativo?*;

- Você teria alguma sugestão de melhoria para a busca/pesquisa de ativo?;
- Os atributos de busca selecionados são suficientes?*
- Você teria alguma sugestão de atributos de busca para ser adicionado na busca por ativos?;
- Disponibilizar ao SoS-User a visualização dos atributos (horas de retrabalho, percentual de reúso, e comentários em geral) de um ativo, colabora para ele ser encorajado a fazer reúso. Dessa afirmação, qual sua opinião?*
- Você considera esses atributos do ativo suficientes?*
- Você teria alguma sugestão de atributos para ser adicionado ao ativo?;
- O ambiente integrado é útil para apoiar o reúso de software. Dessa afirmação, qual sua opinião?*
- O quão bom (fácil) é utilizar o ambiente integrado em relação ao fato de não precisar fazer uma busca manual por ativos?*
- Sugerir o melhor ativo, colabora para uma melhor eficiência no reúso de ativos. Dessa afirmação, qual sua opinião?*
- A ferramenta proporciona fácil visualização dos ativos de um CS. Dessa afirmação, qual sua opinião?*
- A hierarquia de CS->Asset->Artifacts está bem representada no SoS-Reuse-Sim?*
- O treinamento foi bom/adequado?;
- Você teria alguma sugestão de melhoria?;
- Se teve dificuldades, por favor descreva-as abaixo.;
- Integrar os CS's em um único ambiente, que forma o SoS-Reuse-Sim, faz sentido? Qual o seu *feedback* em relação a isso?*

C.3 Questionário do SoS-Admin

- O quão trabalhoso é publicar/disponibilizar um CS na ferramenta?*
- As funções de adicionar, remover e editar CS colaboram para melhor gerência do SoS. Dessa afirmação, qual sua opinião?*
- A ferramenta proporciona fácil visualização dos ativos de um CS. Dessa afirmação, qual sua opinião?*

- O quão importante você considera manter uma lista de sistemas constituintes confiáveis?*
- Fazer uma monitoração do atributo *active* de cada CS's colabora para manter o comportamento emergente disponível. Dessa afirmação, qual sua opinião?*
- O ambiente integrado é útil para apoiar o reúso de software. Dessa afirmação, qual sua opinião?*
- Integrar os CS's em um único ambiente, que forma o SoS-Reuse-Sim, faz sentido? Qual o seu *feedback* em relação a isso?*
- Você teria alguma sugestão de melhoria?;
- Se teve dificuldades, por favor descreva-as abaixo.