



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

INTELLIGOV: UMA ABORDAGEM SEMÂNTICA PARA VERIFICAÇÃO DE
CONFORMIDADE EM ARQUITETURAS ORIENTADAS A SERVIÇO

Haroldo Maria Teixeira Filho

Orientadores

Leonardo Guerreiro Azevedo

Sean Wolfgang Matsui Siqueira

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2014

INTELLIGOV: UMA ABORDAGEM SEMÂNTICA PARA VERIFICAÇÃO DE
CONFORMIDADE EM ARQUITETURAS ORIENTADAS A SERVIÇO


Haroldo Maria Teixeira Filho

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA
OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-
GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO
DO RIO DE JANEIRO (UNIRIO), APROVADA PELA COMISSÃO ABAIXO
ASSINADA.

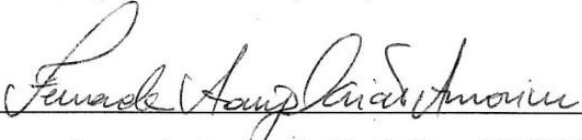
Aprovada por



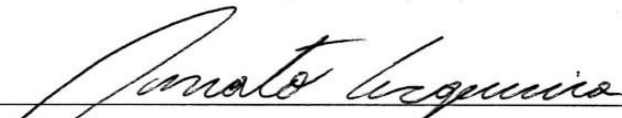
Leonardo Guerreiro Azevedo, D.Sc. – IBM Research - Brasil; PPGI-UNIRIO



Sean Wolfgang Matsui Siqueira, D.Sc. – PPGI-UNIRIO



Fernanda Araujo Baião, D.Sc. – PPGI-UNIRIO



Renato Fontoura de Gusmão Cerqueira, D.Sc. – IBM Research - Brasil



Geraldo Zimbrão da Silva, D.Sc. – UFRJ

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2014

T266 Teixeira Filho, Haroldo Maria.
Intelligov: uma abordagem semântica para verificação de
conformidade em arquiteturas orientadas a serviço / Haroldo Maria
Teixeira Filho, 2014.
159 f. ; 30 cm

Orientador: Leonardo Guerreiro Azevedo.
Co-Orientador: Sean Wolfgang Matsui Siqueira.
Dissertação (Mestrado em Informática) - Universidade Federal do
Estado do Rio de Janeiro, Rio de Janeiro, 2014.

1. Arquitetura orientada a serviços (Computador). 2. Governança
corporativa. 3. Ontologia. 4. Semântica. I. Azevedo, Leonardo
Guerreiro. II. Siqueira, Sean Wolfgang Matsui. III. Universidade
Federal do Estado do Rio de Janeiro. Centro de Ciências Exatas e
Tecnológicas. Curso de Mestrado em Informática. IV. Título.

CDD – 004.65

Agradecimentos

Foram dois anos de trabalho para realizar este objetivo que já tinha a muito tempo: encarar o desafio de um Mestrado. Foi uma árdua, porém especial jornada, na qual aprendi muito e cresci como profissional e pessoa. O resultado foi um trabalho que contribui não apenas para meu desenvolvimento, mas também para a comunidade acadêmica e, em especial, para pessoas que trabalhem nas áreas de SOA e integração de sistemas. Este resultado não seria possível sem o apoio e incentivo de várias pessoas.

Em primeiro lugar, gostaria de agradecer a minha esposa Nanda, que teve muita paciência e compreensão ao longo destes dois últimos anos. Adiamos diversos planos e acabamos limitando nossas vidas para realizar este sonho, sacrificando não apenas o meu dia a dia, mas o dela também. Não tenho palavras para agradecer por isso.

Também gostaria de agradecer a minha mãe, Célia, a meu falecido pai, Haroldo, e as minhas duas irmãs, Amélia e Mariana. Vocês sempre me incentivaram a estudar e crescer como profissional. Sem seu apoio, orientação e discussões ocasionalmente acaloradas, provavelmente não estaria escrevendo estas palavras.

Outro ponto fundamental para conseguir realizar esta dissertação foi o apoio e liberação que consegui no trabalho para estudar e montar um estudo de caso em um ambiente real. Para tal, não posso deixar de agradecer a meus gerentes Haroldo Leão e João Bosco pela liberação e compreensão e aos meus colegas de trabalho Leonardo Lemos e Pedro Ivo Dantas, que acabaram tendo que absorver uma carga maior de trabalho devido ao tempo que dediquei ao Mestrado. Agradeço também a Danilo Felix, Rafael Lima, Benno Albert e Rosane Sfair, que em diversos momentos contribuíram com palavras, exemplos e ideias que ajudaram a estruturar este trabalho.

Uma boa orientação também foi um fator crítico para conseguir chegar até aqui. Para tal, tenho muito o que agradecer aos meus orientadores, Prof. Leonardo Azevedo e Prof. Sean Siqueira, que me sinalizavam caminhos, incentivavam sempre positivamente, sempre me traziam novos desafios e me ensinaram muito enquanto produzia esta dissertação.

Também gostaria de agradecer a Prof^ª. Fernanda Baião, ao Prof. Renato Cerqueira e ao Prof. Geraldo Zimbrão por terem aceitado o convite para avaliar este trabalho e ajudar na melhoria do seu conteúdo e de suas contribuições.

Por fim, não poderia deixar de agradecer aos diversos professores e colegas que me acompanharam ao longo desta jornada. Deste modo, gostaria de destacar o Prof,

Gleison Santos, a Prof^ª. Flavia Santoro, o Prof. Mariano Pimentel, a Prof^ª. Claudia Capelli e ao Prof. Marcio Barros que que, através de aulas, esclarecimento de dúvidas ou participação de bancas em seminários, me ajudaram muito a encontrar o rumo no Mestrado. Considerando meus colegas, gostaria de agradecer ao Anselmo Guedes, Daya Rodrigues, Felipe Abrantes, Felipe Leão, Helio Braga, Marcius Armada, Thaissa Dirr e Thiago Procaci, por sugestões, dicas, boas conversas, troca de experiências e ombros amigos em muitos momentos nestes dois anos.

TEIXEIRA FILHO, Haroldo M. **IntelliGOV: Uma Abordagem Semântica Para Verificação de Conformidade Em Arquiteturas Orientadas a Serviço**. UNIRIO, 2014. 159 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

Resumo

Arquiteturas Orientadas a Serviço (SOA) vem sendo adotadas por organizações para tratar a complexidade de seu conjunto de sistemas, reduzir custos e melhorar o atendimento a prazos. Para atingir estes objetivos, é necessário garantir que a arquitetura e sua evolução estejam alinhadas com os objetivos do negócio, boas práticas, requisitos legais e regulamentações. Porém, a diversidade de domínios de conhecimento e partes envolvidas em SOA demandam trabalho complexo para verificar a conformidade das soluções construídas segundo este paradigma. Este trabalho propõe *intelliGOV*, uma abordagem que propõe o uso de ontologias, regras e consultas semânticas para representar formalmente as regras que definem o comportamento conforme e para inferir a conformidade dos elementos que compõe o ambiente SOA, como serviços e sistemas envolvidos. Para implementar a abordagem é proposta uma arquitetura baseada em padrões abertos – OWL, SWRL e SQWRL. Um estudo de caso executado em uma empresa brasileira de energia foi utilizado para demonstrar a aplicabilidade da abordagem. Como resultado foi observada uma redução de erros no processo de verificação de conformidade. Adicionalmente, a solução é comparada com outras abordagens identificadas na literatura, evidenciando seus pontos positivos e limitações.

Palavras-chave: SOA; Governança; Conformidade; Ontologia; Semântica

Abstract

Organizations are adopting Service-Oriented Architectures (SOA) to handle system landscape complexity, reduce costs and achieve deadlines. To accomplish these goals, it is necessary to ensure the alignment of the architecture and its evolution with the business goals, best practices, legal and regulatory requirements. However, the diversity of domains and stakeholders involved in SOA demands complex work to check the compliance of the solutions. This work proposes intelliGOV, an approach that proposes the use of ontologies, rules and semantic queries to represent the rules that define compliant behavior and to infer the compliance of the elements that composes a SOA solution, like services and systems. To implement this approach, it is proposed an architecture based on open standards – OWL, SWRL and SQWRL. A case study executed in a Brazilian energy company demonstrates the applicability the approach, identifying reduction of errors in the conformance process. Additionally, the solution was compared to other approaches identified in related works, evidencing it strengths and weaknesses.

Keywords: SOA; Governance; Compliance; Ontologies; Semantics

Sumário

1. Introdução.....	1
1.1 Motivação	1
1.2 Descrição do problema	2
1.3 Objetivo do trabalho	4
1.4 Hipótese e método de avaliação.....	5
1.5 Método de pesquisa utilizado e estrutura da dissertação	6
2. Fundamentação Teórica e Trabalhos Relacionados	8
2.1 Arquitetura Orientada a Serviços.....	8
2.1.1 Conceito e benefícios de SOA.....	8
2.1.2 Serviços	9
2.1.3 Barramentos de serviços	11
2.1.4 Governança SOA	12
2.2 Conformidade e políticas	15
2.3 Trabalhos relacionados	16
2.3.1 Uso de CEP e MDA	17
2.3.2 <i>Templates</i> de governança.....	18
2.3.3 SOBUS e bases relacionais.....	19
2.3.4 Semântica	20
2.3.5 Considerações sobre os trabalhos relacionados.....	21
3. intelliGOV	22
3.1 Visão Geral	22
3.1.1 Ontologias, regras e consultas semânticas.....	22
3.1.2 Método de uso do intelliGOV	24
3.1.3 Arquitetura em alto nível intelliGOV	25
3.2 Arquitetura de implementação	26
3.2.1 Ontologias, regras e consultas semânticas.....	27
3.2.2 Componentes de implementação	28
3.3 Benefícios do intelliGOV	30
4. Estudo de Caso	31
4.1 Contexto da organização.....	31
4.2 Metodologia de estudo de caso	32
4.3 Projeto e preparação do estudo de caso	34

4.4	Execução do estudo de caso.....	36
4.4.1	Perfil dos participantes	37
4.4.2	Passos executados do protocolo	38
4.4.2.1	Apresentação sobre o trabalho	38
4.4.2.2	Seleção das políticas	38
4.4.2.2.1	Política 1.....	40
4.4.2.2.2	Políticas 2 e 3	41
4.4.2.2.3	Políticas 4 a 7	42
4.4.2.3	Seleção dos Serviços.....	44
4.4.2.4	Preparação do Gabarito.....	44
4.4.2.5	Coleta de Dados	44
4.4.2.6	Avaliação Manual	45
4.4.2.7	Definição da Ontologia.....	45
4.4.2.8	Definição das Cargas	48
4.4.2.9	Definição das Regras	49
4.4.2.9.1	Política 1.....	49
4.4.2.9.2	Política 2.....	51
4.4.2.10	Política 3.....	52
4.4.2.11	Política 4.....	53
4.4.2.12	Política 5.....	55
4.4.2.13	Política 6.....	57
4.4.2.14	Política 7.....	58
4.4.3	Avaliação usando o intelliGOV.....	59
4.4.4	Discussão dos resultados	60
5.	Análise dos Resultados.....	61
5.1	Análise das avaliações	61
5.1.1	Política 1.....	62
5.1.2	Políticas 2 e 3.....	63
5.1.3	Políticas 4 a 7.....	64
5.2	Avaliação do esforço	65
5.3	Discussão dos Resultados	66
5.4	Ameaças à validade	70
5.4.1	Tecnologia	70
5.4.2	Método.....	71
5.4.3	Representatividade dos dados.....	72

6.	Comparação com outras abordagens.....	73
6.1	Tecnologias propostas nos trabalhos relacionados	73
6.2	Comparação com o uso de base de dados.....	74
6.3	Comparação com o processamento de eventos.....	80
6.4	Comparação com o uso de MDA.....	86
6.5	Comparação com Data Warehouse.....	87
6.6	Comparação com outras abordagens de web semântica.....	87
6.7	Consolidação da comparação.....	88
7.	Conclusões	89
7.1	Conclusões sobre o trabalho	89
7.2	Contribuições	90
7.3	Trabalhos futuros	91
	Referências	94
	Apêndice I. commonGOV	100
	Apêndice II. Procedimento de revisão de literatura	106
	Apêndice III. OWL, SWRL e SQWRL	111
	Apêndice IV. Ontologia utilizada no estudo de caso	117
	Apêndice V. Problemas identificados no estudo de caso.....	121
	Apêndice VI. Estimativa de esforço para implementação do intelliGOV.....	138
	Apêndice VII. Função <i>validatePolicy</i>	145

Índice de Figuras

Figura 1. Mapeamento entre o método e a estrutura da dissertação.....	6
Figura 2. Ciclo de uso de um serviço, adaptado de PAPAZOGLU (2003).....	9
Figura 3. Estrutura de um ESB, adaptada de PAPAZOGLU (2007).....	11
Figura 4. Modelo para estrutura de governança SOA, adaptado de Niemann (2010)....	13
Figura 5. Verificação de Conformidade com CEP, adaptada de Tran <i>et al.</i> (2011).....	17
Figura 6. Arquitetura para conformidade proposta por Rodríguez (2013).....	19
Figura 7. Abordagem SOBUS (adaptada de Peng, Lui e Chen (2009)).....	20
Figura 8. Arquitetura em alto nível da abordagem intelliGOV.....	25
Figura 9. Módulos utilizados para implementação do intelliGOV.....	28
Figura 10. Detalhamento do módulo de carga.....	29
Figura 11. Metodologia de estudo de caso, adaptada de Yin (2009).....	33
Figura 12. Protocolo de execução do estudo de caso.....	35
Figura 13. Exemplo de classificação de serviço.....	40
Figura 14. Exemplo do modelo de versionamento.....	43
Figura 15. Ontologia utilizada no estudo de caso.....	47
Figura 16. Protótipo para carga, utilizado no estudo de caso.....	49
Figura 17. Distribuição das divergências em relação a experiência em SOA.....	67
Figura 18. Distribuição das divergências em relação a experiência na organização.....	68
Figura 19. Modelo lógico de dados utilizado no protótipo.....	75
Figura 20. Exemplo de EPN, representando a política 3.....	81
Figura 21. Estrutura do commonGOV.....	102
Figura 22. Exemplo de assertivas utilizadas com classes em OWL.....	112
Figura 23. Exemplos de propriedades na OWL.....	112
Figura 24. Exemplos de assertivas aplicadas às propriedades em OWL.....	113
Figura 25. Ontologia do Open Group, adaptada de OpenGroup (2010).....	117

Lista De Abreviaturas Utilizadas

CCI – Centro de Competências de Integração
CEP – Complex Event Processing (Processamento de Eventos Complexos)
DAMA – Data Management Association
DSL – Domain Specific Language (Linguagem Específica de Domínio)
EPN – Event Processing Network (Rede de Processamento de Eventos)
ESB – Enterprise Service Bus (Barramento Corporativo de Serviços)
ETL – Extract, Transform and Load (Extração, Transformação e Carga)
HTTP – Hypertext Transport Protocol
HTTPS – Hypertext Transport Protocol Secure
JEE - Java Enterprise Edition.
JMS – Java Messaging Service
JSON – JavaScript Object Notation
OEP – Oracle Event Processing
OSB - Oracle Service Bus
OWL – Ontology Web Language
MDA – Model-driven architecture (Arquitetura baseada em Modelos)
REST – Representational State Transfer
SOA – Service-Oriented Architecture (Arquitetura Orientada a Serviços)
SOAP – Simple Object Access Protocol
SQWRL – Semantic Query-Enhanced Rule Language
SWRL – Semantic Web Query Language
URI – Universal Resource Identifier
W3C – World Wide Web Consortium
XML – eXtensible Markup Language

1. Introdução

Neste capítulo serão apresentados a motivação deste trabalho, o problema a ser endereçado, a hipótese considerada, os objetivos e o método de pesquisa aplicado, além da organização do texto.

1.1 Motivação

Arquitetura Orientada a Serviços (SOA) é uma abordagem para construção de aplicações baseada na descoberta, invocação e composição de funções distribuídas, acessíveis através de protocolos padronizados, denominadas de *serviços* (PAPAZOGLU *et al.*, 2007). O objetivo é reduzir custos e prazos de desenvolvimento, operação e manutenção de software através do reuso de serviços existentes (ERL, 2005).

No entanto, de acordo com NIEMANN *et al.* (2010), o número de componentes flexíveis e partes envolvidas existente em um ambiente baseado em SOA é elevado, ampliando a sua complexidade. Algumas consequências desta complexidade são relatadas na literatura:

- Desenvolvimento de várias versões semelhantes do mesmo serviço, reduzindo o reuso e ampliando a complexidade do ambiente (NIEMANN *et al.*, 2010);
- Dificuldade para expandir a arquitetura em uma escala corporativa e global, dada a natureza distribuída de SOA e variações existentes em termos de regulamentações, tanto das unidades da própria organização como nos países onde ela atua (HSIUNG, RIVELLI e HUTTENEGGER, 2012);
- Dificuldade para gerir a mudança em um ambiente com várias partes envolvidas (SCHEPERS, IACOB e VAN ECK, 2008).

Uma solução para estes problemas seria a implantação de governança SOA. Diversos autores da academia (HOJAJI e SHIRAZI, 2010; JOACHIM, BEIMBORN e WEITZEL, 2013; NIEMANN *et al.*, 2010; SCHEPERS, IACOB e VAN ECK, 2008) e da indústria (BENNETT, 2012; BROWN, MOORE e TEGAN, 2006; THOMAS MANES, 2009) ressaltam este ponto, propondo modelos de governança que indicam os processos e componentes necessários para implementar governança SOA nas organizações.

Um ponto comum existente nestes modelos é a presença de atividades para definir e verificar conformidade. De acordo com TRAN *et al.* (2012), estas atividades permitem estabelecer meios para assegurar que serviços e aplicações são construídos e utilizados em organizações de maneira alinhada com os objetivos do negócio, leis, regulamentações e boas práticas. Sem a existência de um mecanismo que garanta conformidade, provedores e consumidores de serviços passam a tomar decisões desalinhadas, comprometendo a integridade da arquitetura. NIEMANN *et al.* (2010) por sua vez, definem como principal objetivo da governança SOA a garantia de conformidade com regras que permitam que organizações distintas consigam compartilhar serviços respeitando acordos firmados entre si, padrões de mercado e regulamentações legais. Neste caso, os autores extrapolam o contexto de SOA para a interligação de organizações distintas, tornando ainda mais importante que decisões, padrões e ações sejam realizadas de forma que empresas participantes de soluções orientadas a serviço consigam transacionar de maneira harmonizada.

1.2 Descrição do problema

SCHEPERS, IACOB e VAN ECK (2008) citam a necessidade de dispor de mecanismos que garantam a conformidade com políticas representando as regras das organizações. Tal mecanismo deve ser capaz de checar a aderência dos elementos que compõem o domínio SOA, como serviços, sistemas, contratos e usuários, com políticas que especifiquem regras espelhando regulamentações, normas e boas práticas a serem seguidas na constituição de soluções para SOA.

TRAN *et al.* (2011) apontam que a atividade de verificação de conformidade em SOA é complexa, uma vez que esta depende do conhecimento de diversas perspectivas, muitas vezes baseadas em textos e regras de difícil formalização e dependente do trabalho de especialistas de várias áreas. Para RODRÍGUEZ *et al.* (2013), a atividade de

verificação de conformidade é uma tarefa importante, cara e complexa. Ela é importante devido à constante pressão sobre as organizações por conformidade com diversas regulamentações externas. Ela é cara devido à intrusividade da atividade e à demanda elevada de recursos para disponibilizar informações e acompanhar auditores ao longo do processo. Ela é complexa porque os requisitos de conformidade usualmente são perversivos, cobrindo diversas áreas e processos distintos nas organizações.

Ao focar o domínio de SOA, a questão de conformidade se torna mais crítica. TRAN *et al.* (2012) consideram que o caráter de integração entre silos originários da aplicação de SOA demanda a integração das preocupações e problemas de conformidade entre áreas distintas da organização. HSIUNG, RIVELLI e HUTTENEGGER (2012) alertam para a dificuldade de manter regras e políticas de conformidade em cenários que integram organizações, muitas vezes em escala global, operando com vocabulários distintos e regulamentações variadas.

TEIXEIRA FILHO e AZEVEDO (2012, 2014) tornaram este fato evidente, ao realizarem uma comparação entre propostas de governança SOA originárias do mercado e da academia, com o intuito de determinar o conjunto necessário de processos para implantação de governança SOA nas organizações. Como resultado, foram identificados 51 processos distintos, dependentes de conhecimento disperso em perspectivas diversas, variando desde questões técnicas, como modelagem e construção de serviços, até aspectos ligados à gestão, como estratégia, controle de custos e gestão de recursos humanos.

O problema tratado por este trabalho pode ser resumido da seguinte forma: **A atividade de verificação de conformidade em SOA é uma atividade complexa devido à necessidade de conhecimento e informações relativas a diversas perspectivas de conhecimento. Isto dificulta a compreensão, transcrição e avaliação das regras que definem comportamento conforme de maneira precisa.**

Por fim, cabe analisar a importância da resolução deste problema. Do ponto de vista técnico, conforme citado anteriormente, a atividade de verificação de conformidade é central na implementação de mecanismos de governança SOA. Segundo o OPEN GROUP (2009), a maioria das organizações tem dificuldade em estender ganhos obtidos em projetos piloto de SOA para uma escala corporativa devido à ausência de mecanismos efetivos de governança. Para BRAZIER *et al.*, 2012, a medição de conformidade permite avaliar a aderência de provedores de serviços a acordos e normas entre organizações, viabilizando uma classificação de bons

provedores e permitindo o uso de SOA em uma escala interorganizações. Do ponto de vista de negócio, RODRÍGUEZ *et al.* (2013) apontam que erros e falhas no atendimento à conformidade podem ocasionar danos à imagem das organizações, riscos para segurança, multas, falências e, até mesmo, prisão dos dirigentes.

1.3 Objetivo do trabalho

O objetivo deste trabalho é propor uma solução que permita realizar verificações de conformidade no contexto SOA com menor quantidade de erros e com baixa dependência de especialistas para executar esta atividade. Desta forma, será permitido deslocar especialistas para atividades de maior valor agregado e garantir uma baixa quantidade de erros neste processo. Portanto, a solução proposta deve garantir redução de erros e reduzir a complexidade da solução em relação às outras abordagens.

Para atingir este objetivo, este trabalho propõe a abordagem denominada *intelliGOV*. Seu foco é capturar e formalizar o conhecimento dos domínios envolvidos através de ontologias e explicitar políticas através de regras e consultas semânticas. Com esta formalização, torna-se possível o uso de agentes computacionais para realizar a verificação de conformidade com redução na quantidade de erros, ampliando a precisão do resultado obtido.

Uma arquitetura de software foi desenvolvida para permitir a extração de dados do ambiente SOA de uma organização, sua conversão em instâncias da ontologia e a avaliação de conformidade destas instâncias considerando as regras e consultas representando as políticas.

Utilizando esta arquitetura, especialistas são envolvidos durante a etapa de especificação da ontologia, reduzindo seu trabalho na etapa de verificação de conformidade. Deste modo, estes podem dedicar a maior parte de seu tempo ao tratamento de problemas identificados nas verificações, reduzindo o tempo gasto em atividades de coleta e investigação de dados de baixo valor agregado.

O uso das regras e consultas semânticas permite reduzir a complexidade e erros através do reuso de condições já existentes na ontologia, que representem restrições ou cenários específicos das organizações e utilizando um vocabulário padronizado constituído dos termos da ontologia. Isto facilita o processo de especificação das regras e consultas que definem o comportamento conforme.

1.4 Hipótese e método de avaliação

Considerando o contexto da abordagem intelliGOV, a hipótese a ser corroborada neste trabalho pode ser enunciada da seguinte maneira: **“SE aplicarmos uma abordagem baseada em ontologias, regras e consultas semânticas no processo de verificação de conformidade em SOA, ENTÃO serão obtidas reduções de erros e dependência de especialistas nesta atividade”**.

Para avaliar se a solução corrobora esta hipótese, optou-se por um estudo de caso. YIN (2009) define estudo de caso como uma pesquisa empírica que investiga a fundo um fenômeno contemporâneo em seu contexto real, especialmente quando os limites entre o fenômeno e o contexto não estão evidentes, não existe controle sobre as variáveis comportamentais e o foco é em eventos ocorrendo no instante da execução do estudo de caso. No contexto de sistemas de informação, RUNESON e HÖST (2008) frisam que as atividades de construir, manter e operar sistemas são realizadas por pessoas em diferentes contextos e que o aspecto multidisciplinar da área permite que fatores sociais e políticos impactem no resultado, tornando difícil a diferenciação entre os limites entre fenômeno e contexto. Tal ponto torna as atividades de desenvolvimento de software passíveis de pesquisa através de estudos de caso.

No contexto deste trabalho, deve ser considerado o comportamento de pessoas na execução da verificação de conformidade, uma vez que se busca uma solução que reduza a quantidade de erros em comparação à avaliação realizada por especialistas. Para tal, é fundamental avaliar o desempenho humano, com as suas imprevisibilidades, para uma comparação com os resultados obtidos pela abordagem proposta. Adicionalmente, devem ser considerados problemas e erros que possam vir a ocorrer em um contexto de uma organização, visando a geração de cenários de erro de conformidade para análise. A obtenção destes dados não seria efetiva em um ambiente controlado, uma vez que seria impossível reproduzir o comportamento humano e condições existentes em uma organização real. Como se busca o comportamento de pessoas na avaliação de conformidade, sem interferência do pesquisador e não se espera alterar em um primeiro momento o trabalho dos envolvidos, não se aplica a Pesquisa-ação.

1.5 Método de pesquisa utilizado e estrutura da dissertação

Esta dissertação apresenta o resultado da aplicação de um método de pesquisa que contempla uma avaliação baseada em estudo de caso no contexto de conformidade em SOA. Deste modo, a estrutura deste texto descreve os itens gerados ao longo da pesquisa. A Figura 1 apresenta um mapeamento entre as etapas realizadas e a estrutura desta dissertação.

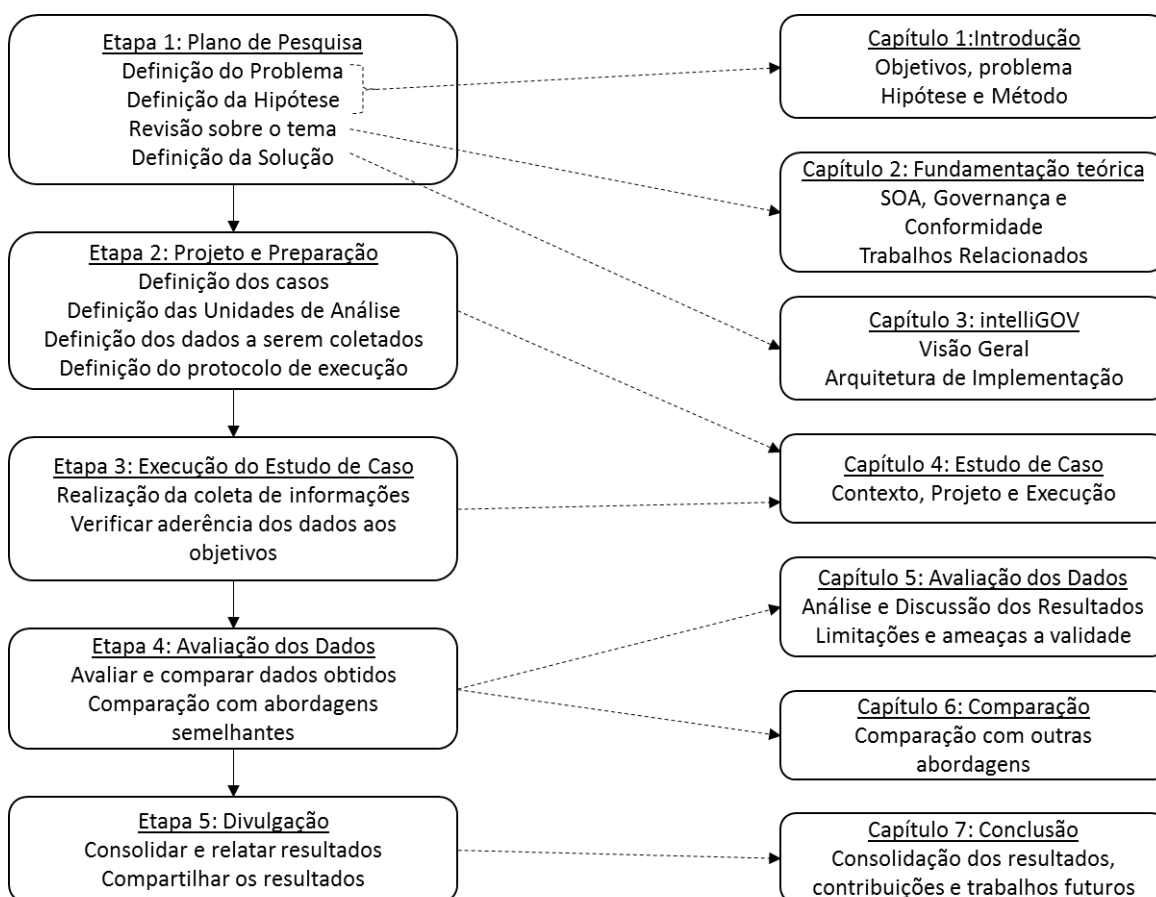


Figura 1. Mapeamento entre o método e a estrutura da dissertação

O planejamento da pesquisa possui partes nos capítulos 1, 2 e 3. Este capítulo inicial descreve os objetivos, problema, hipótese a ser avaliada e método utilizado. O capítulo 2 provê o embasamento teórico, listando os conceitos relativos a SOA, governança e conformidade utilizados ao longo do trabalho e são apresentados trabalhos identificados na literatura focados em tratar conformidade em SOA. A solução é apresentada no capítulo 3.

O projeto e preparação do estudo de caso são apresentados no capítulo 4, no qual são descritos o contexto da organização na qual o estudo foi realizado, o protocolo previsto de trabalho e o relato de sua execução.

A avaliação dos resultados é apresentada nos capítulos 5 e 6. No capítulo 5, é realizada uma análise e discussão dos resultados obtidos no estudo de caso e são avaliadas suas limitações e ameaças à validade. No capítulo 6, conclusões identificadas no estudo de caso e protótipos utilizando dados obtidos neste cenário são utilizados para realizar uma comparação entre a solução proposta e as soluções identificadas na literatura sobre o tema.

Por fim, o capítulo 7 consolida a divulgação deste trabalho, apresentando uma visão integrada dos resultados, contribuições identificadas e propondo trabalhos futuros nesta área.

2. Fundamentação Teórica e Trabalhos Relacionados

O objetivo deste capítulo é apresentar os principais elementos tratados nesta dissertação. Na Seção 2.1, são tratados os aspectos relativos a SOA. Na Seção 2.2, são tratados aspectos relacionados a conformidade e políticas. Na Seção 2.3, os trabalhos relacionados sobre o tema são apresentados.

2.1 Arquitetura Orientada a Serviços

Nesta seção são definidos os conceitos de arquitetura orientada a serviços, serviços e técnicas para sua construção e, por fim, ferramentas utilizadas para a implementação e gestão de SOA.

2.1.1 Conceito e benefícios de SOA

Orientação a serviços é um paradigma de projeto que tem como objetivo disponibilizar unidades lógicas, denominadas serviços, que executem funções de negócio de forma que estas possam ser amplamente utilizadas por diversos usuários para compor novos processos e aplicações (ERL, 2011).

Para PAPAZOGLU *et al.* (2007), a orientação a serviços permite o uso de uma programação baseada em serviços, na qual software é desenvolvido a partir da composição de serviços que podem ser dinamicamente localizados e invocados remotamente. Deste modo, ao invés de constantemente construir os mesmos componentes de software, desenvolvedores passam a focar na busca e reuso de serviços existentes, reduzindo o ciclo de desenvolvimento.

O uso da orientação a serviços permite a transformação de arquiteturas de TI baseadas em aplicações isoladas por silos de negócio em conjuntos de serviços interoperáveis que compõem as aplicações e processos das organizações, concretizando uma arquitetura orientada a serviços (PAPAZOGLU, 2003). Neste contexto, a

organização passa a gerar aplicações flexíveis, capazes de tratar distribuição de sistemas e funções, múltiplas partes envolvidas e heterogeneidade (JOSUTTIS, 2007).

Como benefícios da aplicação de SOA, ERL (2011) cita melhorias na interoperabilidade entre sistemas e processos, maior desacoplamento das soluções de TI, maior volume de opções de diversificação de fornecedores, maior alinhamento entre tecnologia e negócio, maior retorno nos investimentos de TI e melhorias na agilidade das organizações. PAPAOGLOU *et al.* (2007) e JANIESCH *et al.* (2009) apontam também a maior facilidade para integrar processos da organização com processos externos, através de serviços disponibilizados na Internet e consumidos por aplicações existentes na organização.

2.1.2 Serviços

Os elementos centrais de SOA são serviços, definidos como entidades capazes de executar funções diversas, variando desde consultas a dados até a execução de processos de negócio. Serviços são passíveis de serem descritos, localizados e invocados de maneira padronizada (PAPAOGLOU *et al.*, 2007).

Segundo ERL (2005), todo serviço deve dispor de um contrato, que compreende todo o conjunto de descritores e documentos que definam as funcionalidades ofertadas pelo serviço, protocolos e interfaces para acesso e atributos não funcionais, como disponibilidade e tempo de resposta. Cabe ao usuário do serviço conhecer apenas as informações referentes ao contrato, sem necessidade de conhecer detalhes de sua implementação.

Para disponibilização e uso de um serviço, PAPAOGLOU (2003) propõe o fluxo apresentado na Figura 2.



Figura 2. Ciclo de uso de um serviço, adaptado de PAPAOGLOU (2003)

Neste fluxo, provedores disponibilizam serviços publicando suas informações de descrição em um registro de serviços. Este atua como um catálogo de serviços existentes (passo 1 da Figura 2). Na sequência, interessados no uso do serviço,

denominados consumidores, realizam buscas no catálogo para obter os dados descritivos do serviço e poder incorporar o seu uso em suas aplicações. Esta busca pode ser realizada manualmente por desenvolvedores, durante a etapa de construção da aplicação, ou automaticamente pela aplicação, durante a sua execução (passo 2 da Figura 2). Por fim, uma vez identificado o serviço e determinado o meio de acesso, este passa a ser invocado pelos consumidores, através do envio e recebimento de mensagens previstas na descrição do serviço (passo 3 da Figura 2).

Para implementação dos serviços, diversos autores apontam a tecnologia de *web services* como a mais utilizada (ERL, 2011; JOSUTTIS, 2007; PAPAZOGLU *et al.*, 2007). Existem dois tipos principais de *web services*: baseados no protocolo XML/SOAP, chamados de WS-* (ERL, 2005), e baseados em REST (FIELDING, 2000).

Web services SOAP se baseiam nos seguintes elementos:

- Uso de mensagens XML¹, descritas segundo o formato SOAP²;
- Interfaces e endereços descritos em arquivos XML denominados WSDL³;
- Possibilidade de uso de diversos protocolos de comunicação para transporte de dados (HTTP⁴, HTTPS⁵ e JMS⁶);
- Repositórios de serviços baseados no protocolo UDDI⁷, que normaliza consultas e dados a respeito de serviços.

Web services REST se baseiam em trocas de mensagens para manipular recursos, que representam documentos, imagens e informações disponíveis na web e acessíveis através de uma URI. Sua manipulação é executada utilizando as operações do protocolo HTTP: *PUT*, para criar recursos; *GET* para obter um recurso, *POST*, para alterar um recurso; e eventualmente *DELETE* para excluir um recurso. Os dados são trafegados através de mensagens descritas em XML ou JSON⁸.

¹ XML: eXtensible Markup Language – <http://www.w3c.org/XML>

² SOAP: Simple Object Access Protocol - <http://www.w3.org/TR/soap/>

³ WSDL: Web Services Description Language - <http://www.w3.org/TR/wsdl>

⁴ HTTP: Hypertext Transfer Protocol - <http://tools.ietf.org/pdf/rfc2616.pdf>

⁵ HTTPS: HyperText Transfer Protocol Secure - <http://tools.ietf.org/html/rfc2818>

⁶ JMS: Java Messaging Service - <http://www.oracle.com/technetwork/java/docs-136352.html>

⁷ UDDI: Universal Description Discovery & Integration - <http://uddi.org/pubs/uddi-v3.0.2-20041019.htm>

⁸ JSON: JavaScript Object Notation – <http://www.json.org/>

2.1.3 Barramentos de serviços

De acordo com HEWITT (2009), JOSUTTIS (2007) e PAPAZOGLU (2007), um dos elementos mais importantes para SOA é o uso de um barramento de serviços ou *Enterprise Service Bus (ESB)*. Um ESB estabelece uma infraestrutura de comunicação entre provedores e consumidores. A Figura 3 ilustra a arquitetura de ESB.

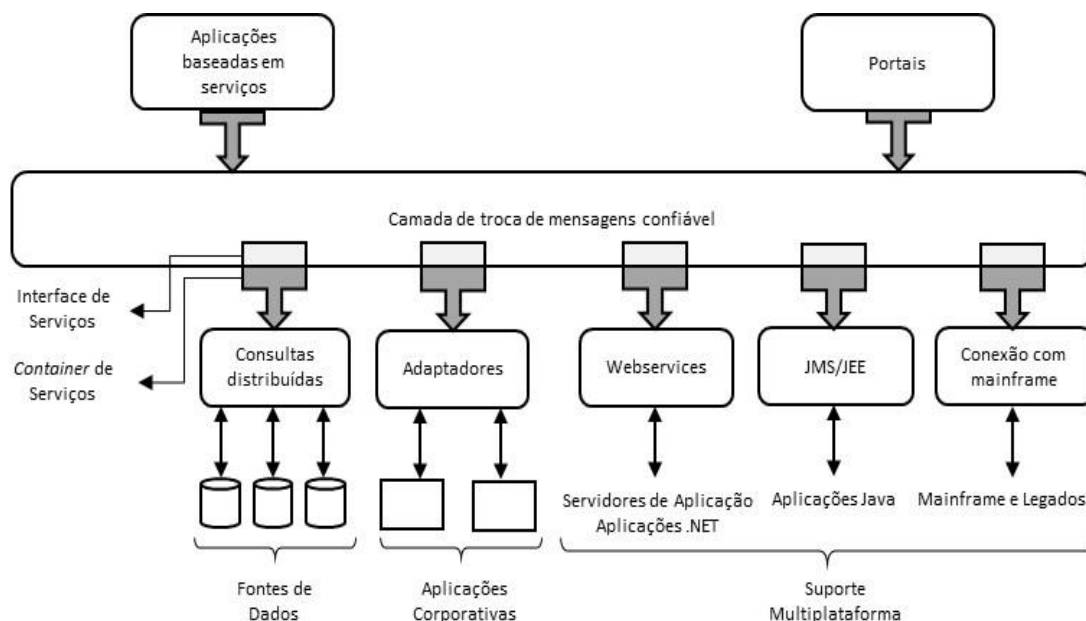


Figura 3. Estrutura de um ESB, adaptada de PAPAZOGLU (2007)

Aplicações e portais que desejam consumir serviços não realizam conexões diretas às aplicações e bases que proveem os serviços, mas através de uma camada de troca de mensagens que garante a entrega de mensagens e monitora as comunicações. Aplicações provedoras de serviços são acessadas pela camada de comunicação através de *containers* de serviços, que permitem priorizar, balancear e controlar acesso aos serviços sob a forma de interfaces de serviços padronizadas, independentes de protocolo de comunicação. Cada *container* permite acesso a mecanismos distintos de integração, facilitando a interoperabilidade com diversas categorias de provedores. Na Figura 3, exemplos são dados acessando fontes de dados através de mecanismos de consultas distribuídas, adaptadores para conexão a aplicações corporativas e pacotes, *web services* para acesso a servidores de aplicação e sistemas construídos em tecnologia .NET, uso de JMS e outros mecanismos JEE⁹ para acesso a aplicações Java, e uso de técnicas para integração com mainframe. Deste modo, as aplicações consumidoras não precisam realizar conversões entre múltiplos protocolos de comunicação, transformar dados ou

⁹ JEE: Java Enterprise Edition - <http://www.oracle.com/technetwork/java/javaee/tech/index.html>

identificar endereços de acesso distintos para cada provedor. O ESB se torna um centralizador deste processo.

ERL (2011) adiciona como possíveis extensões a um barramento corporativo a possibilidade de centralizar regras de negócio, políticas de gestão e monitoramento. Já BEN HAMIDA *et al.* (2012) incluem como funções de um ESB: a capacidade de lidar com composição de serviços, que permite a construção de um novo serviço a partir de serviços existentes, orquestrando suas entradas e saídas a partir de um fluxo de processamento; a possibilidade de manter um catálogo de serviços, indicando quais são os serviços existentes, os dados que os descrevem, mensagens esperadas de entrada e saída e meios de acesso; capacidade de converter e normalizar formatos de dados entre diversas aplicações.

2.1.4 Governança SOA

De acordo com VIERING, LEGNER e AHLEMANN (2009) e JOACHIM, BEIMBORN e WEITZEL (2013), apesar dos aspectos técnicos de SOA terem evoluído, existe dificuldade na implantação de SOA nas organizações.

Para tratar esta dificuldade, academia (HOJAJI e SHIRAZI, 2010; HSIUNG, RIVELLI e HUTTENEGGER, 2012; JOACHIM, BEIMBORN e WEITZEL, 2013; NIEMANN *et al.*, 2010) e indústria (BENNETT, 2012; BROWN, MOORE e TEGAN, 2006; THOMAS MANES, 2009) apontam a necessidade do uso de governança SOA, que é definida por JANIESCH, KORTHAUS e ROSEMANN (2009) como:

“O estabelecimento de estruturas, processos, políticas e métricas que assegurem a adoção, implementação, operação e evolução de SOA alinhada com os objetivos do negócio e conforme com leis, regulamentações e boas práticas. ”

Para organizar governança SOA, NIEMANN *et al.* (2010) propõem um modelo dos elementos necessários para sua estruturação, apresentado na Figura 4.

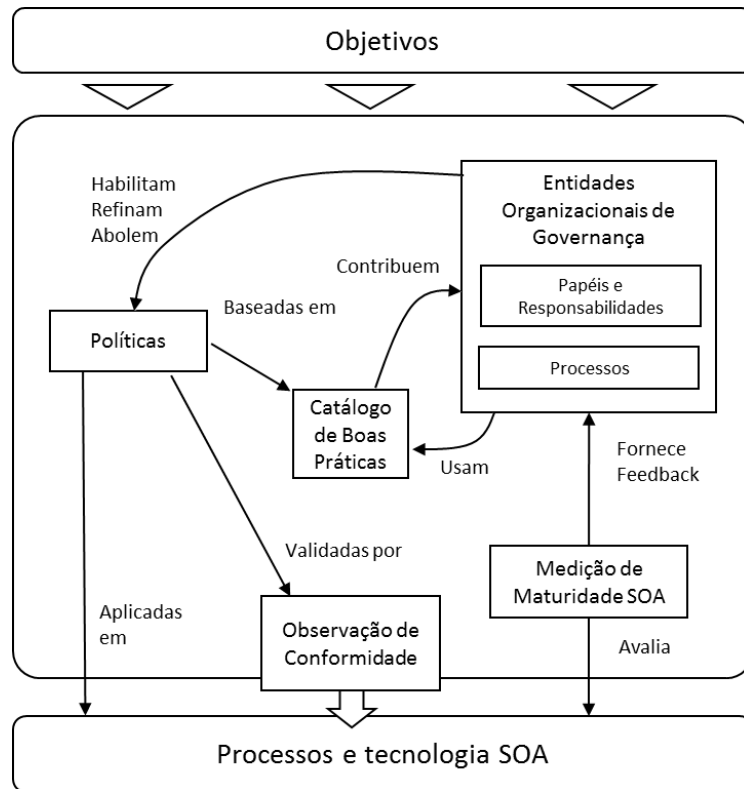


Figura 4. Modelo para estrutura de governança SOA, adaptado de Niemann (2010)

No topo do modelo são representados os objetivos da arquitetura SOA, destacando a motivação de implementação de SOA na organização, os objetivos do negócio e das áreas de tecnologia envolvidas. Para alcançar estes objetivos, são propostos cinco elementos:

- (i) Entidades organizacionais de governança, divididas nos papéis e responsabilidades e em processos de governança, necessários para operação do modelo;
- (ii) Políticas de governança que definem as regras a serem seguidas pela organização em um contexto de interoperação SOA, definindo explicitamente quais os comportamentos esperados para atingir os objetivos da organização e da implantação de SOA;
- (iii) Catálogo de Boas práticas, que serve de orientação para os envolvidos e como base para formulação de políticas e melhoria de processos;
- (iv) Observação da conformidade, para auditar constantemente se as áreas governadas estão aderentes às políticas, verificando desta forma se a arquitetura que está sendo constituída está evoluindo de maneira alinhada com os objetivos do negócio;

- (v) Medição de Maturidade SOA, para avaliar constantemente a evolução do modelo de governança, garantindo que este incorpore tecnologias e práticas de maneira organizada e sustentável.

A parte inferior do modelo representa o sistema que é governado no contexto de SOA. É composto pelos processos utilizados para desenvolver e manter a arquitetura e pela tecnologia utilizada para operacionalizar SOA. Ao descrever o modelo, NIEMANN, MICHAEL *et al.* (2010) citam como tecnologias ESBs, repositórios e os próprios serviços que compõem a arquitetura, mas não detalham os processos necessários para implementar e governar SOA.

Para identificar o conjunto de processos que compõe um modelo de governança SOA, TEIXEIRA FILHO e AZEVEDO (2014) realizaram um levantamento dos modelos de processos para SOA propostos pela academia e indústria e consolidaram estas abordagens, buscando similaridades e divergências. Para tal, foi realizada uma análise dos modelos, listando os processos que são sugeridos por cada abordagem para compor a governança SOA e estes processos foram tabulados, visando identificar que processos são sugeridos com maior frequência e quais seriam as divergências existentes entre os modelos propostos. Com estas informações, foi proposto um modelo comum, resultante da combinação dos modelos analisados. Um resumo deste trabalho está descrito no Apêndice I.

O resultado final foi um conjunto de processos denominado *CommonGOV*, composto por 51 processos necessários para implantar e evoluir SOA de maneira governada. Este modelo foi dividido em quatro grupos:

- (i) Um grupo de estratégia, que contempla os processos necessários para definir os objetivos, estabelecer e acompanhar metas, definir modelos financeiros para cobrança e investimento em serviços além de definir a estrutura organizacional para suportar a implantação de SOA;
- (ii) Um grupo de conformidade, responsável por estabelecer mecanismos de padronização e definição de políticas, auditoria e tratamento de exceções aos padrões existentes;
- (iii) Um grupo de execução, que contempla os processos necessários para construir SOA, considerando o desenvolvimento de serviços e de aplicações consumidoras dos serviços e a gestão do conjunto de projetos relacionados ao desenvolvimento de serviços e aplicações;

- (iv) Um grupo de suporte, que lida com os processos relacionados à gestão de mudanças, monitoração e tratamento de incidentes e problemas.

Deste modo, é possível ver que a governança SOA é uma abordagem que depende de uma gama de conhecimentos ampla para ser implementada, variando desde aspectos técnicos, como versionamento e monitoração de serviços, até aspectos de gestão, como definição e acompanhamento de objetivos estratégicos e concepção de modelos de custeio de serviços e aplicações.

2.2 Conformidade e políticas

NIEMANN *et al.* (2010) afirmam que o principal objetivo da governança SOA é garantir conformidade com padrões e regulamentações da organização, normas e leis. Se considerarmos a definição de JANIESCH, KORTHAUS e ROSEMAN (2009), apresentada na seção 2.1.4, observamos que a implantação de SOA deve ser “alinhada com os objetivos do negócio e conforme com leis, regulamentações e boas práticas”, novamente reforçando a importância da garantia de conformidade no contexto de SOA.

TRAN *et al.* (2012) definem conformidade no contexto de sistemas de informação como os meios que garantam que aplicações de uma organização atuem de maneira adequada a leis, regulamentações e objetivos do negócio.

No contexto de SOA, as regras que especificam o comportamento esperado pela organização são denominadas políticas. Segundo SCHEPERS *et al.* (2008), uma política é uma regra de negócio formal que descreve como serviços devem ser desenvolvidos ou utilizados. ERL (2011) complementa este conceito, afirmando que políticas podem ser aplicadas a qualquer ativo de SOA e que estas formalizam os preceitos que representam os objetivos definidos pela organização. GORTON *et al.* (2009) indicam que políticas podem descrever um detalhamento funcional ou não funcional da execução de tarefas de processo, podem representar restrições do negócio que devem ser consideradas durante a sua execução ou indicar soluções para erros comuns na execução dos processos.

Considerando o modelo de NIEMANN *et al.* (2010), apresentado na seção 2.1.4, as políticas são aplicadas sobre os processos e tecnologia que são utilizados para compor o ambiente SOA e são validadas através do componente de verificação de conformidade.

Para implementar um mecanismo de garantia de conformidade, RAMEZANI, FAHLAND e VAN DER AALST (2012) definem cinco passos a serem seguidos:

- (i) Elicitação da conformidade, para definir quais são as políticas adequadas à organização;
- (ii) Formalização da conformidade, na qual as políticas são refinadas e descritas de maneira precisa;
- (iii) Implementação da conformidade, na qual sistemas são implementados para aferir conformidade;
- (iv) Verificação de conformidade, na qual é efetivamente verificada a conformidade do ambiente; e
- (v) Melhoria da conformidade, na qual são executadas melhorias nos processos e sistemas envolvidos para ampliar a conformidade.

Os autores também distinguem duas possíveis formas de verificar conformidade: *forward checking*, na qual são incluídos mecanismos em tempo de desenvolvimento que permitem checar conformidade antes de se implantar um processo; e *backward checking*, que analisa os dados obtidos no ambiente real, atuando de maneira corretiva.

Um ponto complexo para verificação de conformidade em SOA está ligado à multidisciplinaridade da área. Conforme visto na seção 2.1.4, TEIXEIRA FILHO e AZEVEDO (2014) identificaram não apenas um volume alto de processos para manter uma arquitetura de serviços controlada, mas também uma diversidade de domínios de conhecimento necessários para executar e manter os processos.

Esta diversidade de processos implica em diversos tipos de políticas a serem consideradas ao se montar um modelo de governança. NIEMANN *et al.* (2010) citam nove domínios distintos de políticas para governança: arquitetura, gestão de projetos, finanças, operação de serviços, padronização de dados, gestão de ativos, tecnologia, segurança e organização. Isto exige conhecimento das pessoas responsáveis pela definição, manutenção e verificação das políticas em todas estas nove áreas. ZHOU *et al.* (2010) sinalizam que mecanismos de garantia de conformidade em SOA devem ser capazes de lidar com diversos tipos e níveis de granularidade de políticas e devem ser capazes de lidar com a heterogeneidade inerente a SOA.

2.3 Trabalhos relacionados

Nesta seção são apresentadas abordagens identificadas na literatura para tratar a questão de conformidade, cobrindo propostas de métodos e ferramentas que atuam na representação de políticas e verificação de conformidade em SOA. O procedimento

utilizado para identificar estes trabalhos é apresentado no Apêndice II. Uma comparação destas abordagens com o trabalho proposto nesta dissertação é apresentada no capítulo 6.

2.3.1 Uso de CEP e MDA

Uma primeira linha de pesquisa é baseada no uso da tecnologia de processamento de eventos complexos (*Complex Event Processing* ou *CEP*) (LUCKHAM, 2002). CEP lida com a identificação da ocorrência de condições ou padrões de comportamento durante a ocorrência de eventos no ambiente da organização

No contexto de conformidade em SOA, BIRUKOU *et al.* (2010) e TRAN *et al.*, (2011) apresentam o uso de CEP para identificar conformidade. A filosofia seguida é apresentada na Figura 5.

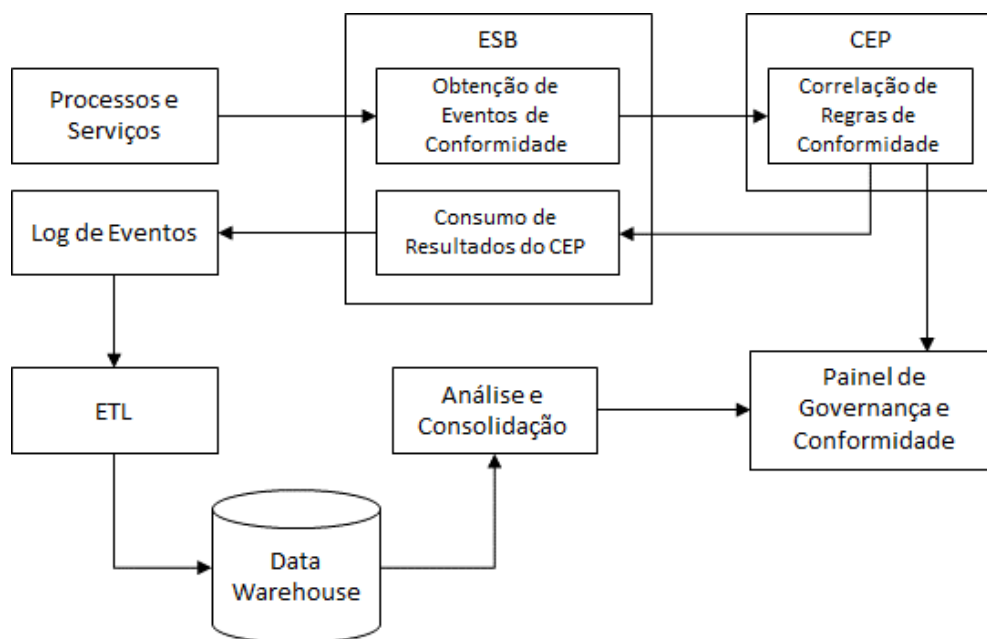


Figura 5. Verificação de Conformidade com CEP, adaptada de Tran *et al.* (2011)

Através do uso do ESB, processos e serviços são monitorados com o objetivo de capturar eventos que sejam relevantes para a análise de conformidade. Esta coleta de informações é implementada através do uso de *Model Driven Architecture* (MDA), paradigma no qual sistemas são gerados a partir de modelos (MELLOR *et al.*, 2002). Expressões descritas em uma linguagem específica de domínio (*Domain Specific Languages* – DSL) e interpretáveis pelas ferramentas que geram código são agregadas aos modelos que descrevem os serviços. Os componentes de software que implementam

os serviços são gerados automaticamente, incorporando gatilhos que notificam o ESB a respeito de eventos relevantes.

Uma vez capturados os eventos, estes são encaminhados para a plataforma de CEP, que verifica a conformidade aplicando regras para identificar casos de não conformidade. As regras são descritas no formato fornecido pela ferramenta de CEP, no caso a *Esper Event Processing Language*¹⁰.

Cada vez que é identificada uma não conformidade, esta é sinalizada pelo mecanismo de CEP para o ESB. O ESB registra a ocorrência em um log de eventos e a exibe para os usuários em um painel de governança e conformidade. Periodicamente, as entradas do log de eventos são carregadas para uma plataforma de Data Warehouse através de mecanismos de Extração, Transformação e Carga (ETL), passando por um processo de análise e consolidação. Ao final desta etapa, são gerados indicadores que permitem acompanhar, em um nível mais alto de abstração o estado de conformidade da arquitetura. Estes indicadores também são apresentados no painel de governança.

Em outro trabalho, TRAN *et al.* (2012) apresentam uma proposta que permite utilizar mecanismos de MDA para modelar as regras que especificam o comportamento conforme e gerar estas regras em conjunto com os serviços e sistemas envolvidos. A verificação de conformidade torna-se então uma funcionalidade agregada aos elementos da arquitetura SOA. Ao se realizar uma ação não conforme, o próprio serviço sinaliza os problemas, alimentando os outros mecanismos da plataforma – ESB, log de eventos e painel de governança. Os autores citam que, caso se julgue necessário, é possível usar esta abordagem para gerar as regras em linguagens utilizadas por ferramentas de CEP.

2.3.2 *Templates de governança*

Outra abordagem, proposta por RODRÍGUEZ *et al.* (2013), substitui o uso de MDA e CEP por uma estrutura ilustrada na Figura 6. Os fluxos representados em linhas tracejadas representam intervenções ou especificações realizadas manualmente, enquanto que os fluxos em linhas cheias representam as trocas de dados automatizadas. Neste caso, o foco é sobre a conformidade dos processos originários da composição de serviços.

Para aferir esta conformidade, são definidos *templates* de conformidade e indicadores de conformidade em uma ferramenta de edição. Estes *templates* especificam as regras a serem respeitadas pelos processos desenvolvidos na arquitetura. Já os

¹⁰ Esper Event Processing Language - <http://esper.codehaus.org/>

indicadores descrevem as fórmulas a serem utilizadas para calcular grandezas relacionadas à conformidade.

Os *templates* são utilizados para definir uma política de sinalização, que é utilizada pelas plataformas de modelagem de processos e pelo motor de execução de processos para capturar os eventos relevantes para conformidade. Estes eventos são enviados ao barramento de serviços que gera um log de eventos. Por fim, com base no *template* de conformidade e nos indicadores de conformidade, são geradas extrações que alimentam um Data Warehouse que permite a análise e exposição dos resultados. Em todos os casos, os documentos atuam como ferramenta para captura de requisitos, servindo de insumos para times de desenvolvimento implementarem políticas de sinalização, cargas e cálculos de indicadores.

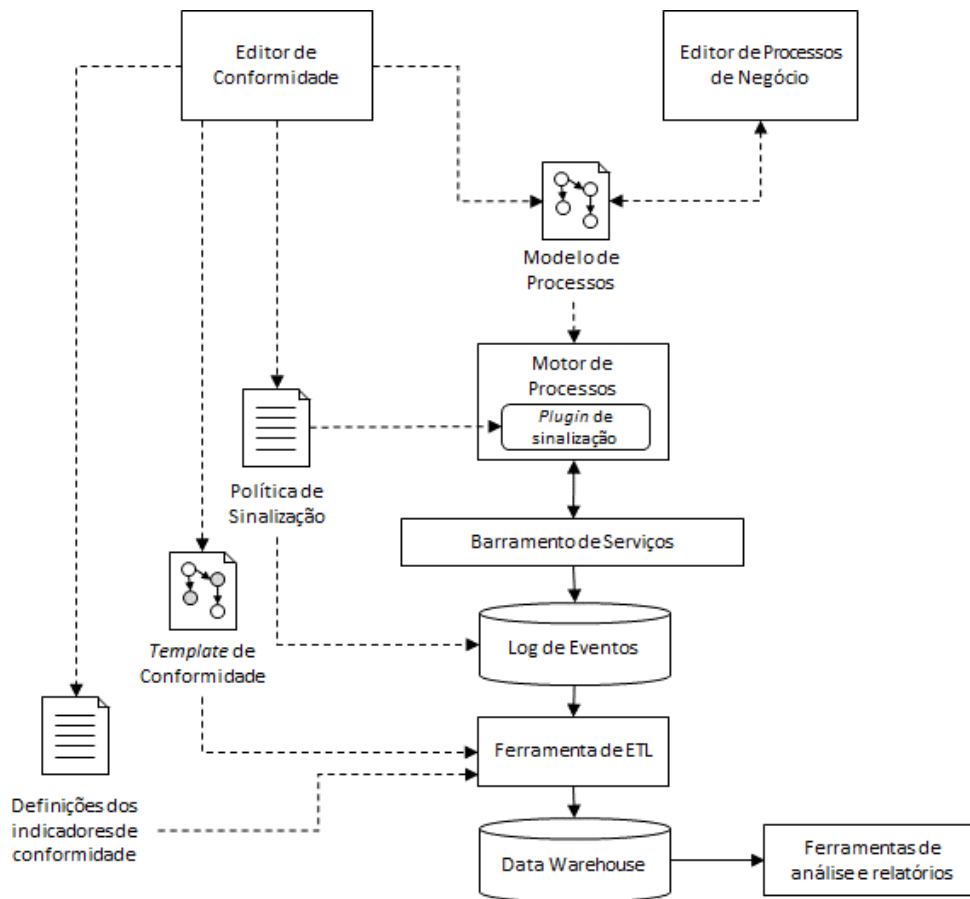


Figura 6. Arquitetura para conformidade proposta por Rodríguez (2013)

2.3.3 SOBUS e bases relacionais

PENG, LUI e CHEN (2008) propõem uma abordagem para identificar e alarmar erros na arquitetura SOA, denominada SOBUS. Sua arquitetura é apresentada na Figura 7.

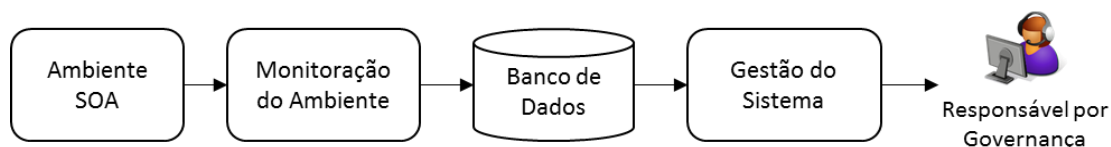


Figura 7. Abordagem SOBUS (adaptada de Peng, Lui e Chen (2009))

Neste caso, um módulo de monitoração do ambiente SOA extrai informações que podem ser utilizadas para verificar conformidade em um ambiente SOA e as fornece para um banco de dados relacional. Este banco é consultado pelos responsáveis por governança através de um módulo de gestão do sistema. Peng, Lui e Chen (2009) trataram apenas políticas técnicas relacionadas a volume e tempo de resposta de serviços, mas apontam para uma evolução para tratar políticas de segurança.

2.3.4 Semântica

Por fim, algumas abordagens se baseiam em padrões e ferramentas de web semântica para operarem.

Seguindo esta linha, ZHOU *et al.* (2010) propuseram uma solução baseada nos padrões *Service Modeling Language (SML)* e ISO Schematron. O primeiro padrão lida com a descrição de um conjunto de serviços e sistemas utilizando um esquema XML padronizado para estas informações, proposto por PANDIT, POPESCU e SMITH (2009). O segundo expande as possibilidades de verificação do XML Schema, através da possibilidade de declaração de assertivas sobre os dados presentes em um XML (JELLIFFE, 2002).

Com base nestes padrões, o contexto SOA de um determinado domínio, composto pelos serviços e sistemas, é representado a partir de arquivos XML. Estes arquivos são alimentados em um mecanismo que agrega as políticas e ações a serem disparadas em caso de não conformidade, através de inclusão de novas *tags* nos arquivos XML e realiza a verificação de conformidade utilizando uma combinação de validação de Schematron com inferência. O resultado é expresso sob a forma de relatórios e pode disparar ações, como desativação de um serviço ou implantação de um serviço em outro ambiente para ampliar sua disponibilidade. Todos estes mecanismos são implementados em módulos de código, devido a necessidade de ampliação do padrão Schematron para tratar o contexto de governança e a inexistência de *reasoner* que possibilitem inferência sobre o formato SML.

Por fim, SPIES (2012) propõe o uso de ontologias para validar conformidade em arquiteturas de TI. Através da modelagem de um determinado domínio utilizando

ontologias, população desta com instâncias representando os elementos do domínio e a classificação das instâncias como conformes ou não conformes utilizando axiomas e regras definidas na ontologia, se torna possível determinar conformidade dos elementos da arquitetura.

2.3.5 Considerações sobre os trabalhos relacionados

Podemos observar que foram propostas diversas soluções distintas para implementação de ferramental para implementar a verificação de conformidade. Porém estas apresentam limitações.

As abordagens baseadas em MDA demandam intrusão no código dos serviços para gerar eventos que devem ser tratados por um mecanismo de CEP (TRAN *et al.*, 2011) ou que implementam a própria verificação agregada à operação do serviço (TRAN *et al.*, 2012). Se considerarmos cenários de organizações que utilizam serviços disponibilizados por provedores externos, promovendo integração interorganizações, esta abordagem tem sua implementação dificultada, uma vez que não existe acesso aos modelos utilizados para gerar os serviços.

Abordagens baseadas em CEP (TRAN *et al.*, 2011) e consultas diretas a bases de dados (PENG, LUI e CHEN, 2008) podem parecer soluções simples, porém levam à implementação de consultas que necessitam de dados a respeito de todas as variáveis envolvidas no contexto da política, tornando-as complexas.

Neste sentido, uma abordagem baseada em semântica poderia simplificar o tratamento destes vocabulários, porém a abordagem proposta por ZHOU *et al.* (2010) demanda a implementação de componentes de software para operacionalizar sua solução e a abordagem proposta por SPIES (2012) não trata mecanismos para carregar as informações e, como poderá ser visto ao longo desta dissertação, não consegue representar regras utilizadas pelas organizações para expressar políticas de governança SOA.

Face a estas limitações, torna-se necessário propor uma nova abordagem para lidar com estas limitações. Esta dissertação propõe o uso de intelliGOV, uma abordagem baseada em ontologias, regras e consultas semânticas para representar e verificar políticas de governança SOA. Para tornar mais claros os ganhos da abordagem, no Capítulo 6 serão revisitadas as limitações citadas nesta seção e as abordagens identificadas na literatura serão comparadas com resultados obtidos da aplicação do intelliGOV em um cenário real.

3. intelliGOV

Neste capítulo é apresentada a abordagem intelliGOV. A seção 3.1 descreve uma visão geral da solução. A Seção 3.2 apresenta uma visão de implementação da solução, descrevendo a construção de protótipos. Na seção 3.3, são apresentados os possíveis ganhos da solução.

3.1 Visão Geral

A abordagem intelliGOV é baseada em uma arquitetura modular que combina ontologias, regras e consultas semânticas com um mecanismo de inferência para: (i) permitir a formalização de políticas de conformidade sob a forma de consultas e regras semânticas; e, (ii) obter uma visão acurada do estado de conformidade. Apoiando este modelo, são propostos módulos que permitem extrair os dados do ambiente SOA para a ontologia, bem como interfaces com o usuário, que facilitam a criação de regras e a execução de relatórios de conformidade.

Para apresentar esta arquitetura, inicialmente serão descritos os elementos centrais da solução – ontologias, regras e consultas semânticas. Em seguida, como estes elementos apoiam a verificação de conformidade. Por fim, é apresentado como os elementos são combinados com módulos de carga e interação com o usuário para compor a solução.

3.1.1 Ontologias, regras e consultas semânticas

ERL (2011) alerta que um fator crítico para implementação de governança SOA é a descrição precisa de políticas para aferir a conformidade. Porém, uma dificuldade neste contexto é a necessidade de unificar e converter vocabulários que representam os diversos domínios que devem ser tratados ao especificar estas políticas. Deste modo, uma solução para governança passa pela definição de: (i) um vocabulário unificado para

descrever um domínio; (ii) expressões que utilizem este vocabulário para descrever, de maneira precisa o comportamento esperado.

Para tratar a questão do vocabulário, a abordagem intelliGOV tem como elemento central uma *ontologia*. Segundo GRUBER (1993), uma ontologia é a representação explícita de uma conceitualização. Esta descreve conceitos e a forma como estes se relacionam em um domínio.

EUZENAT (2007) enriquece esta definição, afirmando que uma ontologia é uma teoria lógica. Sua interpretação não depende dos usuários que leem suas representações gráficas ou sistemas de gestão de conhecimento que as mantêm. A interpretação está explicitamente declarada. Deste modo, regras permitem restringir o significado de termos que poderiam dispor de várias interpretações em um mesmo contexto.

Para GUARINO (1998), esta interpretação é possível através da definição de um vocabulário, que descreve um subconjunto da realidade, e de assertivas explícitas que definindo a intenção do vocabulário no contexto no qual a ontologia está sendo utilizada.

Um dos fatores que tem despertado interesse no estudo de ontologias é a sua capacidade de ser compreendida tanto por atores humanos quanto por agentes automatizados devido ao formalismo utilizado para sua descrição (SU e ILEBREKKE, 2005).

No contexto deste trabalho, uma ontologia é utilizada para representar os conceitos e relações que compõem o contexto de SOA e integrar estes elementos com o contexto da organização endereçado pelas políticas de governança. Instâncias desta ontologia representam os elementos que compõem a arquitetura. Um exemplo é o conceito serviço, que pode ser representado como uma classe da ontologia. Suas instâncias representam os serviços que efetivamente existem na organização.

Para descrever as expressões que compõe as políticas, é importante dispor de um mecanismo que permita lidar com os conceitos expressos na ontologia e que reutilize os axiomas que esta compreenda. Isto evita que seja necessário repetir restrições inerentes ao domínio na descrição de uma política e simplifica a escrita das políticas.

Considerando esta proposta, uma política é descrita como uma consulta baseada em regras, utilizando como operandos termos existentes na ontologia. Regras são assertivas que descrevem comportamento das organizações, expressando normas organizacionais, políticas das empresas, regulamentações externas e padrões (BAJEC e KRISPER, 2005). Um exemplo é uma política do tipo “Um serviço em estado

operacional deve possuir um contrato de serviço ativo”. O ponto importante é que o vocabulário utilizado por esta regra deve se basear nos conceitos da ontologia, de forma que, caso existam outros axiomas que permitam obter dados relevantes para a política, estes sejam devidamente reaproveitados pela política, como, por exemplo, condições que indicam se um contrato de serviços está ativo.

Para ampliar a expressividade da solução, regras podem ser combinadas com consultas, permitindo operações complexas de agregação e manipulações de conjuntos. Deste modo, se pode estabelecer uma regra como “A média dos tempos de resposta de um serviço não pode ser maior do que o valor contratado por um consumidor do serviço”.

3.1.2 Método de uso do intelliGOV

Considerando as etapas do ciclo de garantia de conformidade proposto por RAMEZANI, FAHLAND e VAN DER AALST (2012), descrito na seção 2.2 deste trabalho, o intelliGOV atua em três das cinco etapas do ciclo:

- (i) **Formalização das políticas:** uma vez definidas as políticas a serem aplicadas pela organização, o domínio envolvido é modelado sob a forma de uma ontologia e a equipe responsável pela formalização das políticas pode montar suas expressões combinando termos da ontologia com operadores disponíveis na linguagem de regras e consultas. Deste modo, é possível verificar se as políticas estão coerentes, se elas estão sintaticamente corretas e se os termos utilizados estão sendo aplicados da maneira alinhada com o seu efetivo significado corporativo. Adicionalmente, considerando múltiplas iterações do método, a tendência é que a cada nova iteração seja reduzido o esforço de modelagem da ontologia, devido ao uso de um modelo cada vez mais completo a respeito do domínio considerado pela organização;
- (ii) **Implementação das políticas:** através de uma arquitetura modular, apresentada na próxima seção, é possível implementar um mecanismo de verificação de conformidade com baixo impacto nos serviços e sistemas envolvidos. É importante observar que, na arquitetura proposta, é utilizada a mesma linguagem de formalização das políticas para sua

execução, sendo necessária apenas a implementação dos mecanismos de carga dos dados na ontologia;

- (iii) Verificação das políticas: utilizando a mesma arquitetura, é possível obter um diagnóstico de conformidade, utilizando os dados extraídos dos ambientes envolvidos. Estas informações são utilizadas pelas consultas e regras que descrevem as políticas para inferir se estas estão sendo atendidas ou não. Novamente, a representação utilizada para inferência é a mesma utilizada para formalizar e implementar as políticas, simplificando o processo.

Se considerarmos as duas abordagens de verificação de conformidade proposta por estes autores (*Forward* e *Backward*), podemos classificar o intelliGOV como uma abordagem *Backward Checking*, que verifica a conformidade no ambiente operacional, possibilitando correção de problemas após a sua ocorrência. O uso desta abordagem permite verificar a conformidade de serviços expostos por parceiros externos, nos quais seria difícil implementar mecanismos de verificação *Forward*, e permite validar o comportamento dos serviços durante sua fase de operação.

3.1.3 Arquitetura em alto nível intelliGOV

Para apoiar a abordagem, foi projetada uma arquitetura que utiliza as ontologias e regras de forma que estas possam se integrar a um ambiente corporativo e possam ser manipuladas pelas equipes que lidam com governança SOA nas organizações. Uma visão desta arquitetura é apresentada na Figura 8.

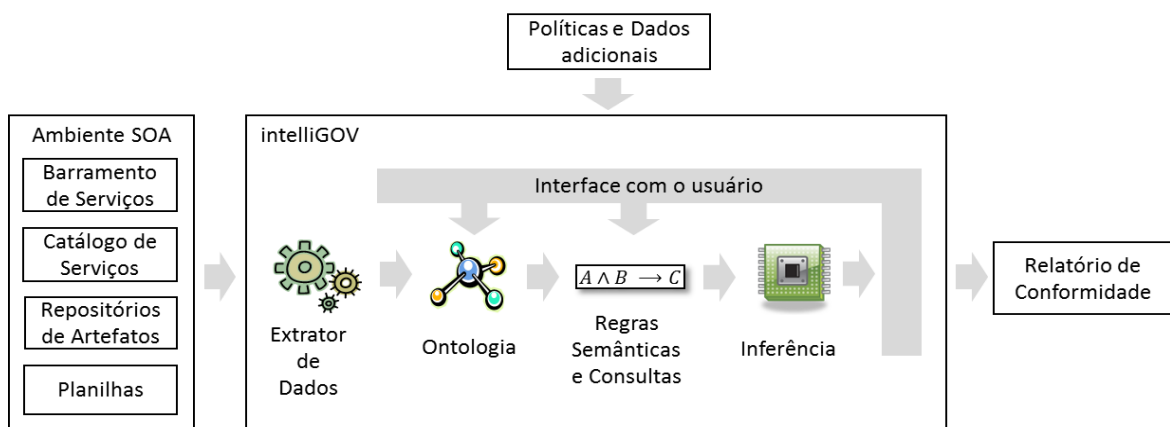


Figura 8. Arquitetura em alto nível da abordagem intelliGOV

Um módulo de **extração de dados** coleta dados do ambiente SOA da organização, que é composto por todas as ferramentas utilizadas para manter e monitorar a arquitetura. Alguns exemplos do **ambiente SOA** seriam barramentos e catálogos de serviços, repositórios de artefatos que mantenham evidências relevantes e planilhas com informações de controle. O módulo **extrator** captura os dados sobre os serviços e gera instâncias na **ontologia** representando estas informações. Como exemplo, considere a publicação de um novo serviço em um barramento. Esta publicação seria capturada pelo módulo extrator, que geraria uma instância da classe serviço na ontologia, com todas as propriedades do serviço passíveis de extração e relevantes para o domínio. Informações que não possam ser extraídas automaticamente podem ser inseridas manualmente, através de uma **interface com o usuário** ou através de cargas de planilhas, visando inserir informações em lote.

As **regras** que representam as políticas de governança são incluídas através de **interface com o usuário**, que disponibiliza o vocabulário expresso na ontologia e o conjunto de operadores e funções permitidos pela linguagem utilizada para o usuário definir de maneira precisa as políticas e realizar validações destas regras. Deste modo, o usuário pode garantir que suas políticas contemplam os termos corretos do ambiente no qual trabalha, reduzindo eventuais erros de especificação originários de uma eventual dualidade de termos e podendo testar o funcionamento destas políticas utilizando dados do próprio ambiente SOA. Por meio desta mesma interface, o usuário pode solicitar a emissão de **relatórios de conformidade**, obtidos através da execução de consultas, submetidas ao **mecanismo de inferência**, que combina as regras e axiomas da ontologia para obter os resultados.

3.2 Arquitetura de implementação

Nesta seção serão detalhados os aspectos de implementação da arquitetura em alto nível descrita na seção anterior. Para tal, foi selecionado um conjunto de ferramentas e tecnologias, baseados em padrões abertos, visando maior interoperabilidade e facilidade de uso. Cada subseção a seguir descreve aspectos de implementação dos componentes da arquitetura.

3.2.1 Ontologias, regras e consultas semânticas

Para a implementação da ontologia, regras e consultas semânticas, foram utilizadas as tecnologias OWL, SWRL e SQWRL respectivamente. Uma descrição detalhada destas tecnologias está disponível no Apêndice III.

Para descrever a ontologia, foi utilizada a *Ontology Web Language (OWL)* (HITZLER *et al.*, 2012). Sua escolha se baseou no fato de que a OWL é recomendada pela W3C como linguagem padrão para descrição de ontologias e existem diversas ferramentas e APIs abertas que permitem a construção de aplicações e integração com motores de inferência que abstraem operações de leitura, manipulação e inferência. Uma outra opção, também indicada pela W3C, seria o uso do *Resource Description Framework (RDF)* (CYGANIAK, WOOD e LANTHALER, 2014). RDF permite descrever conceitos e suas interligações e também dispõe de uma ampla disponibilidade de bibliotecas e ferramentas para manipulação. Porém esta abordagem é mais limitada em termos de capacidade de representação, uma vez que não permite múltiplas categorias de relações, como relações transitivas e inversas e não permite a criação de propriedades de dados. Na realidade, a própria OWL é construída sobre o RDF, incrementando sua capacidade de representação.

Para descrever as regras, optou-se pela *Semantic Web Rule Language (SWRL)* (HORROCKS *et al.*, 2004). Utilizando o vocabulário descrito em ontologias OWL, a SWRL permite a descrição de assertivas representando as regras, que podem ser avaliadas utilizando ferramentas, bibliotecas e ferramentas de inferência compatíveis com OWL.

Por fim, para ampliar a expressividade das políticas especificadas e verificadas usando intelliGOV, foi utilizada a *Semantic Query-Enhanced Web Rule Language (SQWRL)* (O'CONNOR e DAS, 2009). O uso de SQWRL ocorreu devido à necessidade de expandir as opções disponíveis na SWRL, aproveitando o conhecimento disponível na ontologia. Ao contrário de linguagens de consulta como SPARQL-DL (SIRIN e PARSIA, 2007), a SQWRL fornece operadores que permitem implementar operações de manipulação de conjuntos e agregações de dados, ampliando a diversidade de políticas que podem ser tratadas pela solução.

3.2.2 Componentes de implementação

Os módulos utilizados para materializar a arquitetura proposta no item 3.1.2 são apresentados na Figura 9.

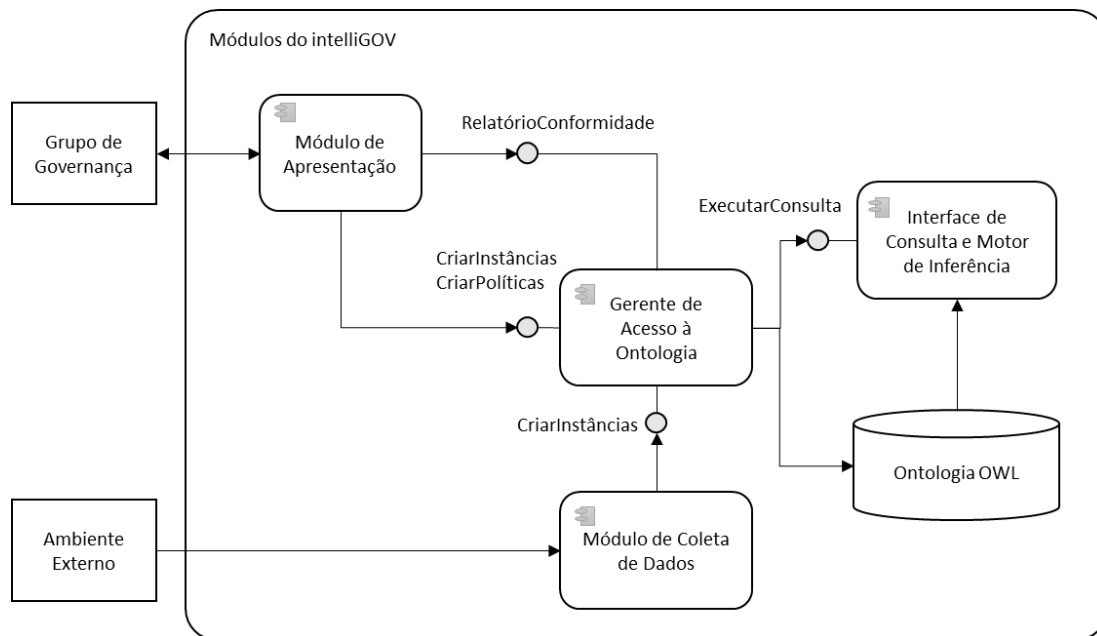


Figura 9. Módulos utilizados para implementação do intelliGOV

Dados do ambiente externo são alimentados como instâncias da ontologia através de um **módulo de coleta de dados**. Este módulo utiliza funcionalidades de um componente denominado **gerente de acesso à ontologia**, que atua como um centralizador dos acessos à estrutura e funcionalidades disponíveis na mesma. Já os usuários, compostos por membros de grupos de governança, responsáveis por definir e auditar políticas, realizam estas atividades através de um **módulo de apresentação** que acessa o gerente de acesso à ontologia. Através destes módulos, os usuários podem inserir e alterar regras e solicitar ao motor de inferência que realize as consultas necessárias para obter os relatórios de conformidade.

Um protótipo que contempla o módulo de apresentação e o gerente de acesso à ontologia foi implementado como projeto final de Daniel Karam (2013). Para tal, estes módulos foram implementados em Java e implantados em um servidor JBoss¹¹. Foram importadas as classes referentes à API do Protégé¹² para prover acesso aos dados da

¹¹ Jboss: <http://www.jboss.org>

¹² Protégé API: <http://protege.stanford.edu/doc/dev.html>

ontologia, atuando como **interface de consulta** e como **motor de inferência**, foi utilizado o Drools Rule Engine¹³, compatível com SWRL e SQWRL.

Para a implementação da **ontologia OWL**, foi utilizada a ferramenta Protégé 3¹⁴, que permite a edição de ontologias e oferece funcionalidades para escrita e testes de regras e consultas baseadas em SWRL/SQWRL. Estas permitiram a depuração da ontologia e das regras.

Por fim, o módulo de carga foi implementado em Java, utilizando APIs das plataformas existentes na organização para consultar os dados e a API do Protégé para criar as instâncias na ontologia. Visando reutilizar este módulo para realizar comparações com outras abordagens utilizando os mesmos dados, a interface de carga foi implementada de acordo com a Figura 10.

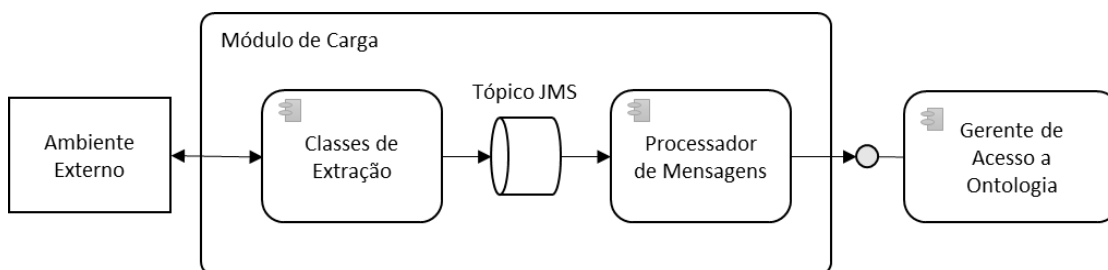


Figura 10. Detalhamento do módulo de carga

Classes de extração foram implementadas utilizando as APIs do ambiente do qual os dados serão extraídos. Sua função é chamar as funções de consulta nos ambientes externos e converter seus retornos em mensagens padronizadas de acordo com os conceitos existentes na ontologia. Desta forma, é construída uma visão genérica e independente de plataforma dos dados. Estas mensagens são publicadas em um **tópico JMS** para consumo por **processadores de mensagens**. Estes recebem as mensagens e as convertem nas instâncias a serem criadas na ontologia, chamando as funções do **gerente de acesso a ontologia** para sua efetiva criação.

A inclusão do tópico JMS¹⁵ permite desacoplar as tecnologias empregadas em cada organização da estrutura de criação de instâncias. Deste modo, caso uma organização deseje alternar sua tecnologia de barramento ou repositório de serviços, basta criar uma nova classe de extração que gere as mensagens no tópico no formato esperado, reduzindo o impacto de manutenção.

¹³ Drools Rule Engine: <https://www.jboss.org/drools/>

¹⁴ Protégé: <http://protege.stanford.edu/>

¹⁵ http://docs.oracle.com/cd/E13222_01/wls/docs51/classdocs/API_jms.html#ovr_destinations

Da mesma maneira, novos processadores de mensagens permitem capturar os dados e encaminhar para outros tipos de abordagem, flexibilizando o modelo e permitindo fazer comparações com os mesmos conjuntos de dados entre diversas abordagens de implementação.

O processador de mensagens permite também o uso de um arquivo de configuração que define quais são os elementos da arquitetura SOA que serão verificados, permitindo assim uma filtragem para verificar serviços de uma área específica ou de um conjunto específico de interesse da organização.

3.3 Benefícios do intelliGOV

A arquitetura de implementação do intelliGOV foi projetada buscando alcançar ganhos que simplifiquem sua aplicação nas organizações. São estes:

- Ausência de codificação para inclusão de novas regras, obtida através da montagem de expressões representando as políticas utilizando as SQWRL e SWRL;
- Facilidade de configuração destas políticas, através do uso de listas de conceitos e propriedades existentes na ontologia para compor as expressões, minimizando a chance de erro;
- Capacidade de extração de dados de diversas plataformas SOA, viabilizada pela desacoplamento entre os módulos extratores e módulos de processamento através do uso de tópicos JMS e mensagens padronizadas nas solução de carga;
- Uso de tecnologias abertas – OWL, SWRL, SQWRL, Protégé e Drools – evitando custos de licenciamento e permitindo a configuração da solução sem a necessidade de codificação para processamento das políticas e manipulação da ontologia.

Para analisar o comportamento da solução, esta foi empregada em um contexto real de SOA no Brasil. Este estudo de caso é apresentado no próximo capítulo.

4. Estudo de Caso

Neste capítulo serão descritos a estrutura e a execução de estudo de caso utilizado para avaliar a solução proposta neste trabalho. Para tal, será descrito o contexto da empresa na qual o estudo de caso foi executado, a estrutura proposta para o estudo de caso, o procedimento executado e os dados coletados. A avaliação dos resultados será apresentada no capítulo 5.

4.1 Contexto da organização

Para avaliar a solução, foi executado um estudo de caso em uma empresa brasileira do segmento de energia, com atuação global. Esta empresa está executando iniciativas em sua área de tecnologia da informação para adotar SOA e para melhorar a gestão das informações cuja custódia é de sua responsabilidade.

A área de Tecnologia de Informação e Comunicações (TIC) da empresa é responsável por capturar requisitos, desenvolver e manter soluções de software e pacotes de software além de integrar estas plataformas com soluções e serviços na nuvem providas por parceiros externos. A empresa conta com equipes de desenvolvimento de software distribuídas em seis estados do país e com fábricas de software de diversos parceiros. As tecnologias utilizadas para desenvolvimento são Java, .NET, Lotus Notes, SQL e SAP ABAP, baseadas em ambientes Windows, UNIX e *mainframe*.

Para tratar a questão de SOA, foi criado um Centro de Competências de Integração (CCI), responsável por orientar o desenvolvimento de soluções de integração de sistemas, promover a cultura de SOA corporativa e governar o ambiente de serviços da organização. Este tipo de estrutura é proposto por JOSUTTIS (2007) e MARKS e BELL (2006), com o objetivo de facilitar a implantação de SOA nas organizações. Não cabe a esta área o desenvolvimento dos serviços propriamente ditos, mas somente sua avaliação, publicação em ferramentas de *middleware*, como barramentos de serviços,

sua documentação no repositório de serviços corporativo e gestão e monitoração após entrada em produção.

As ferramentas utilizadas pelo CCI são: (i) Oracle Service Bus¹⁶ e SAP Process Integration¹⁷ como barramentos; (ii) Oracle Enterprise Repository¹⁸ como repositório e catálogo de serviços; (iii) Clear CASE¹⁹ como repositório e controlador de versões de documentos e código fonte; e, (iv) planilhas Excel para controle de projetos e de atendimentos a incidentes em produção.

A equipe do CCI é composta por 19 colaboradores, sendo quatro funcionários próprios e quinze contratados, que realizam atividades de projeto, avaliação e gestão de serviços. Para desenvolvimento, o CCI utiliza uma fábrica de software externa.

Após quatro anos de operação, o CCI produziu 118 serviços, dominando a técnica de construção, especificação e gestão de serviços. Porém, a quantidade de reusos de serviços é baixa (apenas dez serviços possuem mais do que cinco reusos por sistemas diferentes) e problemas em monitoração e controle de versões de serviços ocasionaram falhas na operação que impactaram sistemas críticos da organização. A equipe do CCI considera que um processo de verificação de políticas relacionadas a estes temas poderia garantir um ambiente mais íntegro e um aumento no reuso de serviços.

4.2 Metodologia de estudo de caso

Para organizar o estudo de caso, foram realizados os passos apresentados na Figura 11, propostos por YIN (2009).

¹⁶ <http://www.oracle.com/technetwork/middleware/service-bus/overview/index.html>

¹⁷ <http://help.sap.com/nwpi>

¹⁸ <http://www.oracle.com/technetwork/middleware/repository/overview/index.html>

¹⁹ <http://www-03.ibm.com/software/products/pt/clearcase/>

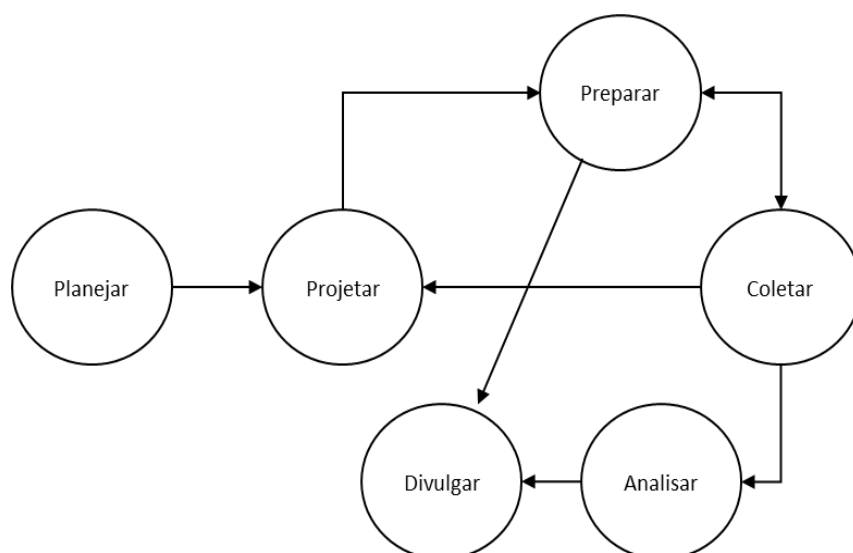


Figura 11. Metodologia de estudo de caso, adaptada de Yin (2009)

Ao Planejar o estudo de caso, é necessário formular questões que deverão ser respondidas (estudos de caso exploratórios) ou hipóteses a serem corroboradas (estudo de caso explanatório). Também é neste momento que é verificada a aplicabilidade do estudo de caso e reconhecidas suas limitações e forças. No caso deste trabalho, neste capítulo foram apresentados o problema, hipótese e método. Para refinar este contexto, foi realizado um estudo sobre os conceitos teóricos envolvidos – SOA, governança e conformidade – e uma revisão de literatura para identificar possíveis soluções para o problema de conformidade e suas limitações.

Na segunda etapa, Projetar, são identificados os casos e unidades de análise a serem consideradas, os dados a serem coletados, a efetiva adequação dos dados aos objetivos da pesquisa e as ameaças a sua validade. Neste trabalho, foi identificado um caso a ser trabalhado – uma empresa brasileira do ramo de energia – e definidas como unidades de análise políticas de governança.

Na etapa Preparar, é definido o protocolo de execução do estudo de caso, obtidas eventuais autorizações necessárias para a sua realização e executados treinamentos ou capacitações necessárias. No contexto desta dissertação, foi elaborado um protocolo que compreende a identificação de políticas, sua avaliação manual e automatizada e a análise dos resultados.

Na quarta etapa, Coletar, o ambiente é observado e os dados são coletados e registrados e eventuais esclarecimentos são obtidos, sendo verificado se estes atendem ou não aos objetivos da pesquisa, podendo levar a uma revisão do projeto caso a

resposta seja negativa. Nesta dissertação, os dados foram coletados em ambiente real através de ferramentas de software.

Na etapa Analisar, os dados são avaliados e organizados de forma que possam ser extraídas conclusões que respondam as questões de pesquisa em estudos exploratórios ou verifiquem se hipótese de pesquisa é corroborada ou negada em estudos explanatórios. Comparações com outras abordagens são realizadas neste momento, para explicitar ganhos e limitações da abordagem proposta. Para este trabalho, foi realizada uma análise considerando os resultados obtidos no estudo de caso e realizada uma comparação com as abordagens identificadas na literatura que atuam sobre o mesmo problema.

Por fim, na etapa Divulgar, os dados são compilados e relatórios são gerados para compartilhamento com a comunidade científica, seja sob a forma de artigos ou dissertações.

4.3 Projeto e preparação do estudo de caso

Nesta seção, serão descritos o projeto e preparação do estudo de caso, tratando três tópicos: a estrutura, o protocolo de execução e os dados a serem coletados.

No cenário desta dissertação, o ambiente SOA gerido pelo CCI foi definido como caso e políticas de governança foram definidas como unidades de análise.

Segundo YIN (2009), um estudo de caso contemplando apenas um caso é aplicável em diversas situações, dentre estas o cenário em que o caso é representativo no contexto avaliado. Dados que a empresa opera em escala global, dispõe de um amplo espectro de tecnologias, utilizadas em outras organizações, e deve interoperar com outras empresas, podemos considerar este cenário como representativo.

Para executar o estudo de caso, foi definido um procedimento, divulgado aos envolvidos, com o objetivo de minimizar eventuais vieses gerados pelas intenções do pesquisador ou por condições ou opiniões específicas dos participantes. O procedimento foi composto pelas etapas apresentadas no modelo de processos apresentado na Figura 12. Este modelo foi desenhado empregando a notação BPMN²⁰.

²⁰ Business process model and notation version 2.0, <http://www.bpmn.org/>

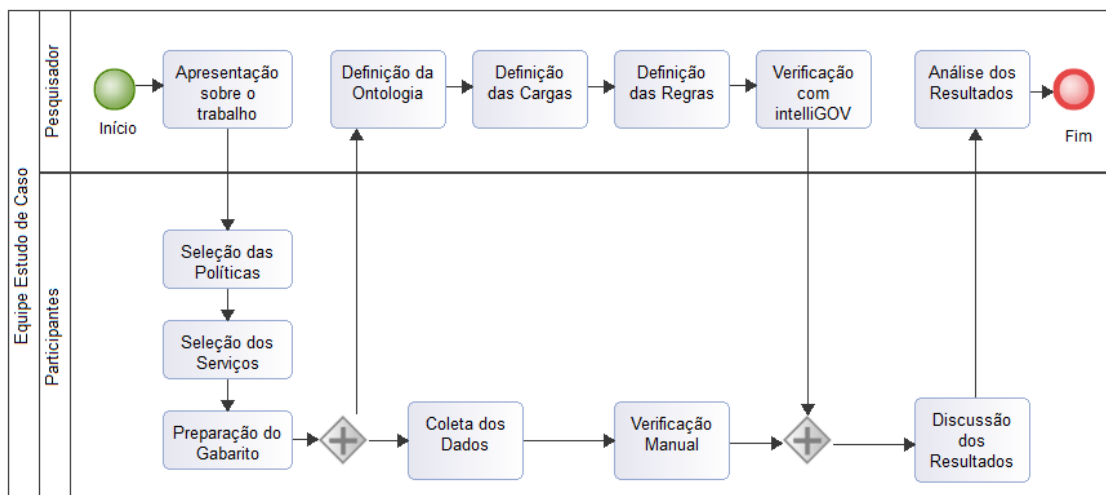


Figura 12. Protocolo de execução do estudo de caso

Inicialmente, foi planejada uma **Apresentação sobre o Trabalho** para os participantes do estudo de caso, visando definir objetivos e planejar as atividades. Na sequência, foram definidas duas atividades que tinham como objetivo obter as unidades de análise e definir a massa de dados a ser utilizada, denominadas respectivamente de **Seleção de Políticas** e **Seleção de Serviços**. Ambas têm como objetivo reduzir o esforço da atividade de avaliação, uma vez que a avaliação de todos os 118 serviços considerando todas as possíveis políticas existentes no ambiente da organização exigiria esforço elevado, que comprometeria a execução do estudo de caso. Por outro lado, as escolhas deveriam ser bem direcionadas a fim do escopo ser amplo o suficiente para a validação.

A terceira etapa, denominada **Preparação do Gabarito**, compreende a avaliação rigorosa de conformidade por um profissional experiente, visando criar um gabarito para verificar erros durante as etapas de avaliação.

Em seguida, o trabalho se desdobra em duas frentes em paralelo. Uma frente corresponde à avaliação manual, na qual integrantes da equipe do CCI realizam manualmente a avaliação dos serviços de acordo com as políticas selecionadas. Eles acessam as ferramentas disponíveis no ambiente de trabalho para realizar a **Coleta de Dados** e realizar efetivamente a atividade de **Verificação Manual**. A outra frente corresponde à avaliação com intelliGOV, ou seja, aplicação da abordagem proposta neste trabalho no mesmo cenário para fins de comparação.

A frente de avaliação com intelliGOV é dividida em quatro passos: (i) **Definição da ontologia**, no qual é definida a ontologia a ser usada; (ii) **Definição das Cargas**, na qual são estendidos os respectivos mecanismos de carga; (iii) **Definição das Regras**,

sendo realizada a tradução das políticas em regras SWRL e consultas SQWRL; e, por fim, (iv) **Verificação** utilizando a abordagem.

Encerradas as duas frentes, o próximo passo é uma reunião para **Discussão dos Resultados** com os envolvidos e identificação de causas e eventuais erros ocorridos ao longo da execução. Com base nas informações coletadas e comentários obtidos nesta discussão, foi realizada uma **Análise dos Resultados**.

O resultado e detalhamento da execução de cada um destes passos serão apresentados na seção 4.4.

Por fim, os dados coletados durante a execução do estudo de caso devem servir de base para corroborar a hipótese enunciada no item 1.4, ou seja, devem permitir mostrar reduções de erros com o uso do intelliGOV.

Para tal, para cada avaliação realizada, considerando um serviço e política, foram anotados o resultado da avaliação e o resultado esperado no gabarito. Cada divergência foi sinalizada como um erro de avaliação. Para cada política, foram somadas as divergências obtidas nas avaliações, obtendo assim o total de erros por política. Este dado foi coletado tanto para a avaliação manual, quanto para a avaliação com intelliGOV, possibilitando a comparação entre as duas abordagens, considerando as políticas como unidades de análise.

Adicionalmente, foi considerado o cálculo do esforço. Para cada avaliação manual foi medido o tempo gasto para realizar a avaliação, com auxílio de cronômetro de relógio digital. Para cada unidade de análise, estes tempos foram somados, dando o tempo total de avaliação da política. Para a abordagem automática, os instantes iniciais e finais das avaliações foram registrados e foi calculada a diferença ao final do processamento. Tal informação permite analisar se também houve reduções de esforços na atividade de verificação de conformidade.

Para a abordagem automática, também foi estimado o esforço necessário para definição da ontologia, das regras e dos mecanismos de carga. Este cálculo é apresentado na seção 5.3.

4.4 Execução do estudo de caso

Nesta seção serão descritos os resultados da execução de cada uma das atividades planejadas para o estudo de caso, apresentando o procedimento realizado e os

dados coletados. A seção 4.4.1 apresenta o perfil dos participantes e os itens seguintes descrevem as etapas listadas na seção 4.2.

4.4.1 Perfil dos participantes

Para a execução do estudo de caso, foram envolvidos 10 dos 19 analistas do CCI, com perfis e experiências variadas, que possuíam disponibilidade de tempo para participar do estudo de caso sem comprometer os projetos da área. A Tabela 1 apresenta a experiência de cada um dos envolvidos. A primeira coluna atribui um número a cada analista. A segunda representa a etapa em que o analista participou do estudo de caso. A terceira a quantidade de anos de experiência do analista em SOA e tecnologias relacionadas, como integração e desenvolvimento de sistemas envolvendo arquiteturas distribuídas. Por fim, a quarta apresenta a quantidade de anos de experiência na organização.

Tabela 1. Características dos participantes do estudo de caso

Analista	Etapas	Experiência em SOA (anos)	Experiência na empresa (anos)
1	Seleção das políticas Seleção dos serviços	6	6
2	Seleção das políticas Seleção dos serviços	10	9
3	Preparação do gabarito Avaliação com intelliGOV	6	6
4	Avaliação Manual	2	1
5	Avaliação Manual	4	2
6	Avaliação Manual	3	8
7	Avaliação Manual	2	4
8	Avaliação Manual	7	1
9	Avaliação Manual	4	2
10	Avaliação Manual	3	3

Visando obter políticas que representassem efetivamente regras relevantes da organização, optou-se por utilizar dois analistas com elevada experiência tanto na área de SOA, quanto na organização. O mesmo nível de experiência foi considerado para selecionar o especialista responsável pela validação do gabarito, uma vez que este seria fundamental para validar os resultados de ambos os métodos de avaliação. Para as avaliações manuais, optou-se por uma diversificação dos níveis de experiência, inclusive considerando pessoas com ampla experiência na organização ou em SOA para poder verificar o comportamento do resultado de acordo com a variação de

conhecimento dos participantes. Visando evitar que quaisquer discussões entre os analistas que definiram as políticas interferissem nos resultados, uma vez que estes debateram aspectos como complexidade e fontes para as informações necessárias para verificação, estes analistas não participaram da verificação manual. Por realizar a validação do gabarito, o analista responsável por esta tarefa também não participou da verificação manual.

4.4.2 Passos executados do protocolo

Nesta seção, são apresentados os passos realizados para a execução do estudo de caso. Cada seção representa um item do protocolo descrito na seção 4.2.

4.4.2.1 Apresentação sobre o trabalho

Inicialmente, foi realizada uma reunião com todos os envolvidos, na qual foi apresentado o protocolo a ser seguido e os objetivos do estudo de caso. O foco era alinhar as atividades e reforçar a importância do trabalho junto aos participantes. Neste momento não foram apresentados processos, políticas, ferramentas ou técnicas para verificação de conformidade visando evitar qualquer tipo de indução por parte do pesquisador que pudesse comprometer a validade dos resultados.

4.4.2.2 Seleção das políticas

Na sequência, foram selecionadas as políticas utilizadas para verificação de conformidade. Visando evitar vieses inseridos pelo pesquisador, dois membros do CCI (analistas 1 e 2 citados na Tabela 1) executaram esta tarefa.

A primeira etapa focou na determinação dos processos que, na visão dos participantes, demandam maior controle e alinhamento entre as equipes que desenvolvem serviços. Para tal, foi apresentado o modelo CommonGOV (TEIXEIRA FILHO, e AZEVEDO, 2012; TEIXEIRA FILHO e AZEVEDO, 2014), que consolida as abordagens propostas pela academia e mercado, apresentando uma lista dos processos necessários para governança. Dentre este conjunto de processos, os participantes destacaram dois processos que originaram problemas graves ao longo de 2013: **modelagem de serviços** e **versionamento de serviços**. A modelagem contempla todas as atividades necessárias para refinar os requisitos funcionais e não funcionais do serviço e definir modelos que descrevam o seu funcionamento. O versionamento compreende atividades que permitam controlar versões dos serviços considerando seu uso nas etapas de desenvolvimento e operação.

Do ponto de vista de modelagem, dois problemas foram diagnosticados pelos participantes: **dificuldades de reuso e segurança**. Considerando reuso, foi informado que existe um catálogo com os serviços disponíveis, porém os potenciais consumidores de serviços tinham dificuldades de localizar os serviços. A visão da equipe é que inexistem mecanismos de classificação adequados para os serviços para facilitar a busca. Já do ponto de vista de segurança, ocorreram erros de seleção de protocolos de comunicação que ocasionaram em incidentes de segurança de informação, devido ao uso de tecnologias pouco seguras utilizadas pelos serviços para trafegar informações.

Para o processo de versionamento, foi identificada a existência de diversas versões de serviços em produção, ampliando o esforço de gestão de mudanças e reduzindo o reuso.

Para melhorar estes dois processos, foi solicitado aos analistas que estes descrevessem políticas para verificar se serviços estão sendo modelados e versionados de forma que os problemas citados sejam reduzidos. O texto descrevendo estas políticas é apresentado na Tabela 2. Uma descrição detalhada de cada política é apresentada nas subseções a seguir.

Tabela 2. Políticas para o estudo de caso

Id	Processo	Política
1	Modelagem	Todo serviço deve ser classificado de acordo com as áreas de assunto associadas às informações manipuladas pelo serviço.
2	Modelagem	Todo serviço deve ser classificado de acordo com os níveis de segurança da classificação de segurança de informação corporativa.
3	Modelagem	Todo serviço com classificação igual ou superior a NP-2 deve utilizar protocolo que implemente criptografia de canal.
4	Versionamento	Todas as variantes de uma <i>release</i> de serviço devem dispor da mesma interface (ou seja, informações de entrada e de saída iguais).
5	Versionamento	Todas as <i>releases</i> de um serviço devem ter interfaces diferentes (ou seja, informações de entrada ou saída distintas).
6	Versionamento	Somente uma variante de cada <i>release</i> de serviço deve estar ativa (disponível para uso) no barramento de serviços.
7	Versionamento	Toda <i>release</i> de serviço que possua um contrato ativo deve estar ativa (disponível para uso) no barramento de serviços.

A seguir são apresentados detalhes de cada uma das políticas.

4.4.2.2.1 Política 1

A primeira política trata as questões de reuso, provendo subsídios para buscas mais precisas no catálogo de serviços. Do ponto de vista da organização, apesar de existir uma funcionalidade de cadastro de taxonomias e classificações na ferramenta utilizada como catálogo, não existia um critério claro para classificar as informações trafegadas pelos serviços, o que facilitaria a busca.

Para resolver este problema, foi especificado pelas equipes de gestão de informações da organização e pelo CCI um mecanismo para classificação de serviços baseado no conceito de **áreas de assunto**. Para a organização, tratam-se de grupos de dados relacionados por um tema comum, como um processo (Exemplo: Logística) ou função (Exemplo: Parceiros de Negócio). No contexto da organização, estas áreas temáticas são organizadas hierarquicamente, permitindo múltiplos níveis de organização. Um exemplo é a área de recursos humanos, que pode ser dividida em gestão de pessoal, clima organizacional e benefícios. A área de benefícios, por sua vez, pode ter mais uma subdivisão, como saúde e previdência.

Utilizando este conceito, um serviço seria associado às áreas de assunto dos dados que compõe suas mensagens de requisição e de resposta. Um exemplo é apresentado na Figura 13, que descreve um serviço hipotético denominado **Empregado**, com uma operação **buscarCentroCustoEmpregado**.

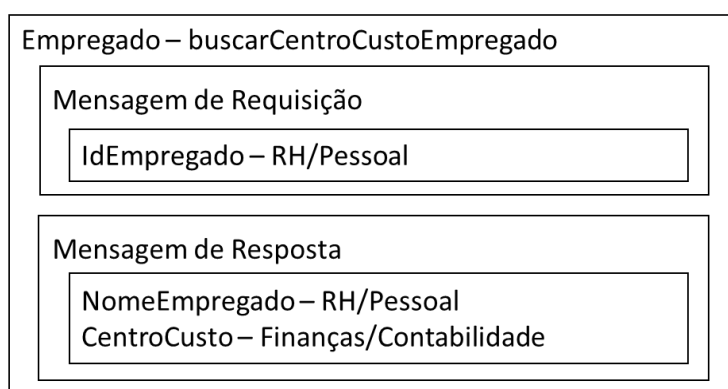


Figura 13. Exemplo de classificação de serviço

A requisição deste serviço contém um campo denominado **idEmpregado**, que está classificado na área de assunto **RH/Pessoal**. Já a resposta contém dois campos, denominados **NomeEmpregado** e **CentroCusto**, classificados respectivamente nas áreas **RH/Pessoal** e **Finanças/Contabilidade**. Considerando o modelo proposto pelo CCI, o serviço deveria estar associado às áreas **RH/Pessoal** e **Finanças/Contabilidade**.

O problema detectado pelos analistas é que as equipes responsáveis pela modelagem dos serviços não estavam verificando as áreas de assunto dos serviços na etapa de modelagem, o que levava a um cadastro incompleto na ferramenta de busca, originando dificuldades no reuso. Deste modo, a política determina que todo serviço deve estar associado às respectivas áreas de assunto das informações.

4.4.2.2.2 Políticas 2 e 3

A segunda e a terceira políticas tratam questões de segurança. Para definir estas políticas, é necessário compreender como a organização trata a questão de segurança de informação. A organização dispõe de um modelo de níveis de segurança de informação que determina quais mecanismos de segurança devem ser aplicados ao se manipular informações em determinado nível. No contexto de serviços, estes níveis determinam a escolha do protocolo de comunicação a ser utilizado e eventuais mecanismos de segurança a serem aplicados, como criptografia de conteúdo e assinatura digital.

Inicialmente, a organização classificava as informações em cinco níveis com criticidade crescente – público, corporativo, reservado, confidencial e secreta. Quanto maior o nível, maior o conjunto de mecanismos de segurança necessários para manipular as informações.

Porém, esta classificação teve que ser alterada no ano de 2013, visando o atendimento à Lei de Acesso a Informação (BRASIL 2011), que segregou as informações em duas categorias – públicas e empresariais, o que poderia gerar confusões com os nomes, uma vez que toda informação que não era pública passou a ser denominada como empresarial, palavra próxima de corporativa. Deste modo, os nomes dos níveis foram alterados, sendo criado um mapeamento entre a classificação anterior e a classificação nova, apresentado na Tabela 3.

Tabela 3. Correspondência entre os modelos de classificação de segurança

Nome do nível na classificação anterior	Nome do nível na classificação nova
Público	Público
Corporativo	NP-1
Reservado	NP-2
Confidencial	NP-3
Secreto	NP-4

Informações públicas continuaram a ser classificadas com este nome, porém informações corporativas foram divididas em quatro Níveis de Proteção, denominados

NPs. De maneira semelhante à classificação anterior, os níveis possuem uma ordem, sendo o público o mais baixo e NP-4 o mais alto, que indicam a maior necessidade de mecanismos de segurança para manipular informações.

Foram identificados dois problemas pela equipe do CCI: o primeiro é a ocasional ausência de classificação do serviço, que não é elicitada corretamente durante a etapa de modelagem. Para tratar esta questão, foi proposta a política 2, exigindo que os analistas responsáveis pela modelagem identifiquem o nível de segurança do serviço nesta etapa.

O segundo é o erro de seleção de protocolos e mecanismos de segurança adequados ao nível de segurança do serviço. Neste ponto, a política 3 trata este problema, indicando que para serviços classificados em níveis maiores ou iguais a NP-2 devem prover algum mecanismo de criptografia de canal.

Para ambas as políticas, dado que existe um legado de serviços que foram desenvolvidos e documentados antes da adequação do modelo de segurança, o mapeamento entre os níveis apresentado na Tabela 3 seria considerado como válido e aplicado em todas as avaliações, sem alterações na documentação existente.

4.4.2.2.3 Políticas 4 a 7

As políticas quatro a sete representam as regras definidas pelo CCI para endereçar as questões de versionamento, baseadas em dois elementos: variantes e *releases*.

Uma *release* de serviço corresponde a uma versão do serviço na qual ocorreu alteração em sua interface, o que pode originar mudanças nos consumidores existentes do serviço. Visando minimizar o impacto, sempre que uma nova *release* é implantada, é mantida em produção as *releases* anteriores com contratos ativos e é emitido um comunicado para os usuários informando a sua publicação e solicitando que estes alterem suas aplicações para consumirem a nova *release*. Conforme estas alterações são realizadas, novos contratos são definidos e, ao serem finalizados todos os contratos de uma *release* depreciada, esta é desativada. Deste modo, é possível dispor de várias *releases* do mesmo serviço em produção em paralelo.

Já a variante é qualquer versão do serviço que não implica em alterações de interface, mantendo os contratos existentes válidos. Neste caso, a nova versão sobrescreve a anterior. Toda variante é baseada em uma *release*, sendo que no contexto

de uma *release*, apenas uma variante pode estar ativa em produção. Um exemplo é apresentado na Figura 14.

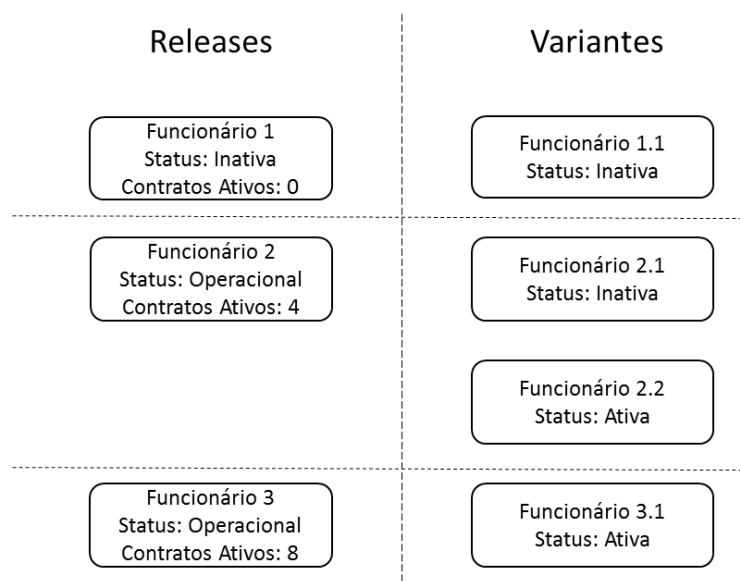


Figura 14. Exemplo do modelo de versionamento

O serviço hipotético *Funcionário* possui três *releases* e quatro variantes. A *release* 1 possui apenas a variante 1.1 e ambas se encontram inativas, uma vez que não existem contratos ativos representando consumidores utilizando esta versão. Já a *release* 2 possui duas *releases*, sendo a primeira inativa e a segunda ativa. Por fim, a *release* 3 possui apenas uma variante ativa. A *release* 2 mantém uma variante ativa devido a existência de quatro consumidores dependentes desta versão, representados pelos quatro contratos ativos. No momento em que estes migrarem da *release* 2 para a *release* 3, a variante 2.2 poderá ser desativada.

Para fins de padronização de nomenclatura, foi definido pelo CCI que toda versão será numerada seguindo o formato R.V, onde R é o número da *release* e V o número da variante em relação a *release*. Para não haver conflitos de nomes, como as variantes 1 das *releases* 2 e 3, ao se indicar uma variante é necessário informar o nome completo da versão. Deste modo, considerando o exemplo da Figura 14, Funcionário 3.1 corresponde à *release* 3 e variante 3.1 do serviço empregado.

O problema identificado pela equipe do CCI é que apesar deste modelo ter sido normatizado e incluído em treinamentos para todas as equipes de desenvolvimento, existe uma dificuldade de diferenciação entre os conceitos de *release* e variante, que levam à duplicação desnecessária de serviços em produção.

Deste modo, as políticas 4 e 5 estabelecem regras para definir se uma alteração em um serviço implica respectivamente em uma nova variante ou uma nova *release*. Já

as políticas 6 e 7 determinam respectivamente o *status* de atividade para variantes e *releases*.

4.4.2.3 Seleção dos Serviços

Uma vez definidas as políticas, a etapa seguinte contemplou a seleção dos serviços a serem avaliados. Cada analista que estava alocado para participar da avaliação manual pôde selecionar, dentre os serviços existentes, de dois a cinco serviços, a fim de não se ter um trabalho muito elevado. Os seguintes fatores foram considerados:

- Cada serviço seria avaliado por apenas um analista, evitando desta forma um montante elevado de avaliações a ser executado por analista e garantindo uma diversidade de dados para análise;
- Os serviços selecionados deveriam ser implementados em tecnologias distintas, com o objetivo de obter resultados que não sejam contaminados por problemas ou características de uma plataforma específica de implementação de serviços;
- A estimativa inicial de esforço de avaliação não poderia passar de quatro horas por pessoa, visando evitar que a execução do estudo de caso compromettesse o andamento dos projetos do CCI.

Deste modo, foi obtida uma lista de 25 serviços.

4.4.2.4 Preparação do Gabarito

Antes de iniciar as avaliações, foi executada pelo autor uma avaliação manual de todas as políticas, considerando todos os serviços, para compor um gabarito que permitisse a verificação de acertos e erros no processo de avaliação de conformidade. Para evitar qualquer tipo de viés, este gabarito foi validado por um analista da equipe do CCI (analista 3 da Tabela 1).

4.4.2.5 Coleta de Dados

Para coletar dados e realizar suas avaliações, cada analista recebeu uma planilha Excel com uma aba por serviço e cada aba contendo uma tabela com os seguintes campos:

- Identificador da Política: contém o número da política, de acordo com o campo Id, presente na Tabela 2;

- Expressão da Política: texto contendo a política a ser validada, de acordo com o campo Política, presente na Tabela 2;
- Situação: campo preenchido pelo analista para indicar o estado de conformidade do serviço, de acordo com a política avaliada, podendo assumir os valores conforme ou não conforme;
- Justificativa: campo livre para o analista preencher com o motivo pelo qual julgou o serviço conforme ou não para esta política.

Para a coleta de dados, os analistas acessaram os ambientes que julgaram mais adequados para obtenção de informação, incluindo os consoles de administração do barramento, cliente da ferramenta ClearCASE e repositório de serviços.

4.4.2.6 Avaliação Manual

Para a avaliação manual, considerando os dados coletados na etapa anterior, cada analista preencheu os campos **Situação** com o seu diagnóstico (Conforme ou Não Conforme) e **Justificativa** com a razão pela qual julgou um determinado serviço não conforme com a política, sendo obrigado a preencher este campo somente em caso de não conformidade.

Durante esta etapa, o tempo foi medido marcando-se a quantidade de segundos com auxílio de relógios dos analistas.

4.4.2.7 Definição da Ontologia

Inicialmente, foi necessária a definição de uma ontologia que permitisse o uso da abordagem no contexto do estudo de caso. Como o foco deste trabalho é a avaliação da abordagem e não a definição de uma ontologia completa para governança SOA, optou-se pelo uso da metodologia 101 (NOY e MCGUINNESS, 2001), por ser uma metodologia de relativa simplicidade de aplicação (SOUZA *et al.*, 2009).

Como primeiro passo da metodologia, é necessário definir questões de competência, que permitam definir os conceitos e propriedades existentes na ontologia. Para tal, foram tomadas como base as políticas definidas na Tabela 2 e foram geradas as questões de competência apresentadas na Tabela 4:

Tabela 4. Questões de competência para o estudo de caso

Questão	Descrição
Q1	Qual é o conjunto de serviços de uma organização?
Q2	Quais são as informações de entrada e saída que definem a interface de um determinado serviço?
Q3	Quais são as áreas de assunto relacionadas às informações trafegadas por um serviço?
Q4	Quais são as áreas de assunto relacionadas a um serviço?
Q5	Qual é a classificação de segurança de um serviço?
Q6	Qual é o protocolo de comunicação utilizado por um serviço?
Q7	Quais são os mecanismos de segurança implementados por um protocolo de comunicação?
Q8	Quais são as <i>releases</i> disponíveis de um serviço?
Q9	Quais são as variantes existentes para um serviço?
Q10	Qual é o <i>status</i> de um serviço no barramento corporativo?
Q11	Quais são os contratos de serviço existentes para um serviço?
Q12	Quais são os contratos ativos e inativos existentes?

Com base nestas questões, foi verificado se existiam ontologias disponíveis que atendessem a estas questões. Baseado em estudo realizado por TEIXEIRA FILHO e AZEVEDO (2013), no qual foram analisadas propostas diversas de ontologias para governança SOA e descritas as suas características, optou-se pelo uso da ontologia proposta pelo OPEN GROUP (2010). Esta ontologia foi escolhida por focar nos conceitos que descrevem os elementos que compõe uma arquitetura orientada a serviços e, dentre as ontologias analisadas, dispor de maior número de classes e propriedades que possibilitam o atendimento às questões de competência. Esta ontologia foi estendida para contemplar todos os itens constantes do domínio do estudo de caso. Uma visão da ontologia resultante é apresentada na Figura 15. A descrição completa da ontologia pode ser vista no Apêndice IV.

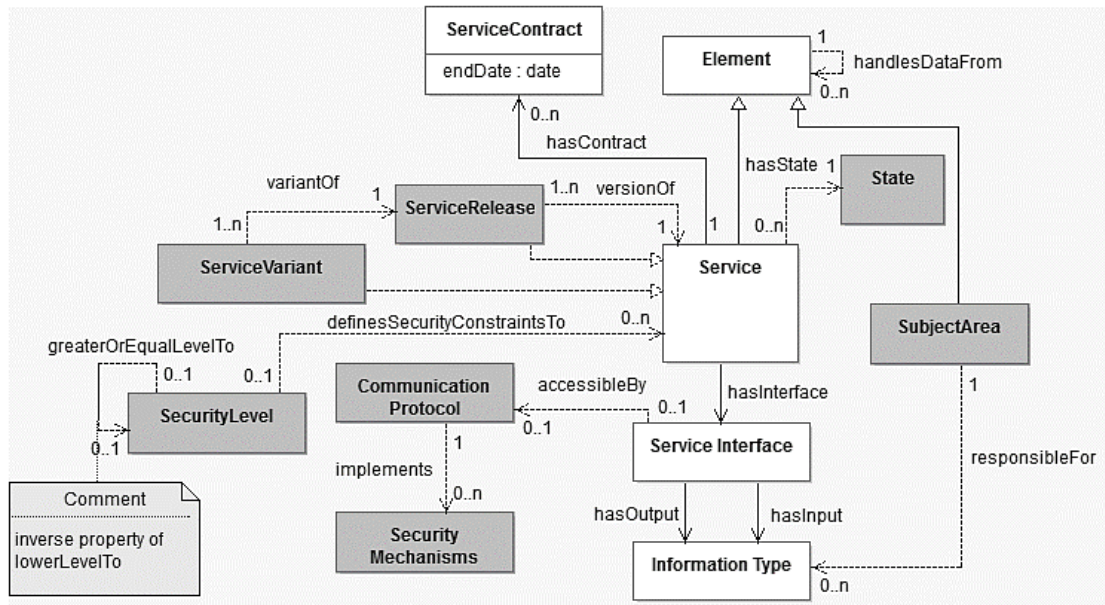


Figura 15. Ontologia utilizada no estudo de caso

Os conceitos aproveitados da ontologia do Open Group são expressos na cor branca, enquanto que os conceitos estendidos estão apresentados em cinza. As relações originárias da ontologia do Open Group estão representadas por linhas cheias, enquanto que as relações estendidas são representadas com linhas tracejadas. Visando simplificar o desenho, propriedades inversas não foram apresentadas, exceto nos casos em que existe relevância para este trabalho. Também não foram representados conceitos e propriedades existentes na ontologia do Open Group que não foram utilizados neste trabalho.

Considerando a ontologia do Open Group, o conceito *Element* representa os elementos que podem existir em um ambiente SOA, sendo possível especializá-los em diversos tipos de elementos, como serviços, áreas de assunto, unidades organizacionais e sistemas. No contexto deste trabalho, foi aproveitada uma especialização, denominada *Service*, que representa os serviços disponíveis na organização. Contratos de serviço, representando relações entre provedores e consumidores de serviços, são representados pelo conceito *ServiceContract* e relacionados aos serviços através da propriedade de objeto *hasContract*. O conceito *ServiceInterface* representa as possíveis interfaces disponíveis de serviços, relacionadas através da propriedade *hasInterface*, e definem entradas e saídas respectivamente através das propriedades de objeto *hasInput* e *hasOutput*, que relacionam a interface com os tipos de informação trafegadas, representadas por instâncias do conceito *InformationType*.

Para tratar a questão de classificação de informações, foi criada uma nova especialização de *Element* denominada *SubjectArea*, para representar as áreas de assunto disponíveis na organização. Também foi incluída uma propriedade de objeto *handlesDataFrom* em *Element*, indicando a possibilidade de um elemento manipular dados de outro elemento, como por exemplo um serviço que manipula dados de uma área de assunto ou de um outro sistema. Por fim, foi incluída uma propriedade de objeto *responsibleFor*, indicando que Área de Assunto é responsável por cada tipo de informação.

Para tratar a questão de segurança, foram incluídos conceitos *Security Level* para representar os níveis de segurança, *Communication Protocol* para representar os protocolos de comunicação e *Security Mechanisms* para representar mecanismos de segurança, como criptografia de canal e de conteúdo. Deste modo, é possível relacionar: (a) uma interface de serviço com o protocolo utilizado para expor seus dados - pela propriedade *accessibleBy*; (b) os mecanismos de segurança implementados por um protocolo - através da propriedade *implements*; (c) um serviço com seu nível de segurança - através da propriedade *definesSecurityConstraintsTo*. Já a ordenação dos níveis de segurança é expressa pela propriedade *greaterOrEqualLevelTo*, com propriedade inversa *LowerLevelTo*.

Por fim, a questão de versionamento é tratada através dos conceitos *ServiceRelease* e *ServiceVariant*, que herdam todas as características de serviços, porém são diferenciadas pelas propriedades *versionOf*, que relaciona serviço com *release*, e *variantOf*, que relaciona variante com *release*. Para indicar o status de serviços, foi criada a classe *State*, relacionada com o serviço através de propriedade *hasState* e herdada pelas variantes e *releases*. O status é padronizado pela equipe do CCI, podendo assumir os valores *Operational*, *Development*, *Deprecated* e *Retired*.

Ao final, o modelo foi validado com o analista 3 do CCI, para assegurar sua correspondência com o domínio da empresa. Na sequência, esta ontologia foi implementada em OWL, com apoio da ferramenta Protegé.

4.4.2.8 Definição das Cargas

Para carga das informações, o modelo de interface de carga apresentado na Figura 10 foi instanciado através de um protótipo, seguindo a arquitetura apresentada na Figura 16.

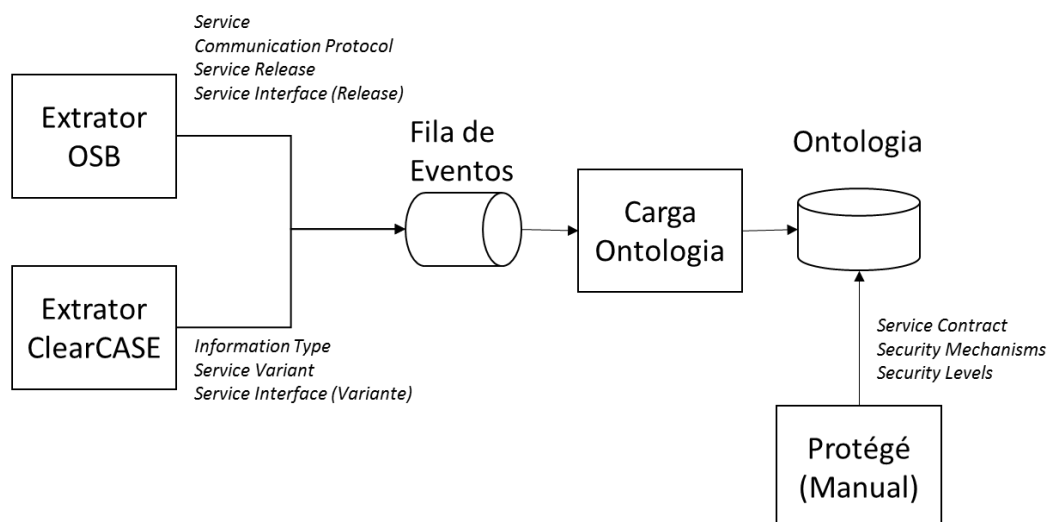


Figura 16. Protótipo para carga, utilizado no estudo de caso

Foram implementados dois extratores. Um para o barramento corporativo utilizado, denominado Extrator OSB (Oracle Service Bus), que obtém os dados das instâncias dos conceitos *Service*, *Service Release*, *Communication Protocol* e *Service Interfaces* referentes às *releases* de serviços, utilizando a API do Oracle Service Bus para consultar as informações. Outro extrator denominado Extrator ClearCASE executa scripts baseados na ferramenta cleartool para obter *Service Variants* e *Information Types* e *Service Interfaces* referentes às variantes. As informações referentes a *Service Contracts*, *Security Mechanisms* e *Security Levels*, por estarem controladas em planilhas ou outras ferramentas foram carregadas manualmente. Como o volume é pequeno, o esforço de entrada foi reduzido.

4.4.2.9 Definição das Regras

Nesta etapa, cada uma das políticas foi descrita sob a forma de uma consulta SQWRL, utilizando os conceitos expressos na ontologia. Nesta seção, descreveremos cada regra.

4.4.2.9.1 Política 1

Para a primeira política, definida como:

“Todo serviço deverá ser classificado de acordo com as áreas de assunto associadas às informações manipuladas pelo serviço”

Foi descrita inicialmente a seguinte consulta SQWRL, para retornar todos os serviços conformes com esta regra:

1 *soa:Service(?s) ∧ soa:ServiceInterface(?si) ∧ soa:InformationType(?input) ∧*
2 *soa:InformationType(?output) ∧ igov:SubjectArea(?saServ) ∧*
3 *igov:SubjectArea(?saInp) ∧ igov:SubjectArea(?saOut) ∧*
4 *igov:handlesDataFrom(?s, ?saServ) ∧ soa:hasInterface(?s, ?si) ∧*
5 *soa:hasInput(?si, ?input) ∧ soa:hasOutput(?si, ?output) ∧*
6 *igov:responsibleFor(?saInp, ?input) ∧ igov:responsibleFor(?saOut, ?output) ∧*
7 *sqwrl:makeSet(?setSubServ, ?saServ) ∧ sqwrl:makeSet(?setSubInp, ?saInp) ∧*
8 *sqwrl:makeSet(?setSubOut, ?saOut) ∧ sqwrl:groupBy(?setSubServ, ?s) ∧*
9 *sqwrl:groupBy(?setSubInp, ?s) ∧ sqwrl:groupBy(?setSubOut, ?s) ∧*
10 *sqwrl:difference(?dif1, ?setSubOut, ?setSubServ) ∧ sqwrl:difference(?dif2,*
11 *?setSubInp, ?setSubServ) ∧ sqwrl:size(?siz1, ?dif1) ∧ sqwrl:size(?siz2, ?dif2) ∧*
12 *swrlb:equal(?siz1, 0) ∧ swrlb:equal(?siz2, 0) → sqwrl:select(?s)*

Esta regra é composta por quatro blocos.

No primeiro bloco, compreendido entre as linhas 1 e 3, são identificados os elementos envolvidos: serviços denotados por *s*, interfaces de dados, definidas por *si*, tipos de entrada e saída de dados, identificados por *input* e *output*, a área de assunto relacionada ao serviço, definida por *saServ*, e as áreas de assunto relacionadas às informações de entrada e saída, dadas respectivamente por *saInp* e *saOut*.

No segundo bloco, compreendido entre as linhas 4 e 6, os dados são relacionados, através de suas propriedades de objeto. Para tal, são definidas as relações entre o serviço e sua área de assunto (termo *igov:handlesDataFrom*), entre o serviço e sua interface (termo *soa:hasInterface*), entre a interface e suas entradas (termo *soa:hasInput*) e suas saídas (termo *soa:hasOutput*). Também são associadas neste momento as áreas de assunto relativas às entradas e saídas (termos *igov:responsibleFor*). Se fosse executada neste ponto, a consulta retornaria todos os serviços que possuem todas estas relações estabelecidas. Para aplicar as condições definidas pela política, são necessários os outros dois blocos.

O terceiro bloco da consulta, compreendido entre as linhas 7 e 9, cria conjuntos para agrupar as áreas de assunto relativas ao serviço, entradas e saídas, e consolidar estes grupos por serviço. Deste modo, é possível obter para cada serviço, quais são as áreas de assunto relacionadas ao serviço e as áreas de assunto relacionadas aos dados utilizados na interface. Os termos *sqwrl:makeSet* montam os conjuntos e os termos *sqwrl:groupBy* realizam o agrupamento. Desta forma, são gerados três conjuntos: o

conjunto de áreas relacionadas aos serviços *setSubServ*, e o conjunto de áreas relacionadas às informações de entrada (*setSubInp*) e saída (*setSubOut*) do serviço.

O último bloco da consulta, compreendido entre as linhas 10 e 12, verifica quais serviços possuem o conjunto *setSubServ* igual aos conjuntos *setSubIn* e *setSubOut*, assegurando que o serviço está relacionado a todas as áreas de assuntos associadas aos dados. Desta forma, a consulta retorna apenas os serviços cujas áreas de assunto são iguais às áreas de seus dados.

Um ponto importante deve ser considerado na operação de comparação entre conjuntos. Inicialmente, era suposto ser possível fazer esta atividade utilizando o operador *sqwrl:equal*. Porém, ao ser implementada, foi identificado que o operador não consegue tratar os agrupamentos dos conjuntos. Deste modo, foi necessário replicar o comportamento do operador, através de uso de operações para diferença entre dois conjuntos (*sqwrl:difference*) e verificando se o tamanho deste resultado é igual a zero (*sqwrl:size* para calcular o tamanho do conjunto e *swrlb:equal* para comparar este valor com zero). É importante observar que a função *swrlb:equal* difere da função *sqwrl:equal*. A primeira compara valores, a segunda compara conjuntos.

4.4.2.9.2 Política 2

A segunda política verifica a associação de serviços a níveis de segurança de informação. Sua descrição é revista a seguir.

“Todo serviço deve ser classificado de acordo com os níveis de segurança da classificação de segurança de informação corporativa.”

Para representar esta regra, foi descrita a seguinte consulta SQWRL para retornar os elementos conformes:

```
1 soa:Service(?oServ)          ^          igov:SecurityLevel(?oLevel)          ^
2 igov:definesSecurityConstraints(?oLevel, ?oServ) →
3 sqwrl:selectDistinct(?oServ)
```

Esta regra é composta por dois termos que declaram o serviço e os níveis de segurança, respectivamente *soa:Service(?oServ)* e *igov:SecurityLevel(?oLevel)*, representados na linha 1, e por um termo que aponta para a relação entre o serviço e o nível de segurança, representado na linha 2. Como resultado, todos os serviços que

estiverem associados a um nível de segurança estarão classificados como conformes, sendo retornados pelo termo *sqwrl:selectDistinct(?oServ)*. Ao contrário da política apresentada na seção anterior, não foi necessário nenhum tipo de ajuste para a regra ser executada de maneira adequada.

4.4.2.10 Política 3

A terceira política trata a verificação de aplicação de mecanismos corretos de criptografia de acordo com o nível de segurança aplicado ao serviço. Sua especificação é revista a seguir:

“Todo serviço com classificação igual ou superior a NP-2 deve utilizar protocolo que implemente criptografia de canal.”

O enunciado desta política traz um diferencial em relação às anteriores, uma vez que esta cria restrições apenas para um subconjunto dos serviços existentes, no caso os que possuem classificação de segurança igual ou superior ao nível NP-2. Desta forma, um serviço classificado em nível inferior estaria automaticamente conforme com a política. Tal comportamento levaria a uma estrutura do tipo OU ao ser descrita sob forma de uma expressão lógica, como por exemplo:

(Serviço com classificação menor que NP-2) OU (serviço com classificação igual ou superior a NP-2 E que utilizam criptografia de canal) → serviço conforme

Para expressar uma operação do tipo OU em SWRL é necessário quebrar a expressão em duas assertivas, informando que ambos os comportamentos implicam em um serviço conforme, uma vez que não existe operador OU em SWRL. Deste modo, para compor a consulta SQWRL que lista os serviços conformes, foi necessário executar a seguinte sequência de passos:

- Criação de uma regra SWRL classificando como conformes os serviços com classificação menor que NP-2;
- Criação de uma regra SWRL classificando como conformes os serviços com classificação igual ou superior a NP-2 e que atendam ao requisito de criptografia de canal;
- Execução de uma consulta para listar os serviços conformes.

Deste modo, a primeira expressão foi descrita como:

- 1 $igov:definesSecurityConstraintsTo(?sl, ?s) \wedge igov:lowerLevelTo(?sl, igov:NP2) \rightarrow$
- 2 $igov:CompliantSecurityService(?s)$

Esta regra define que todo serviço s , associado a um nível de segurança sl através da propriedade de objeto $definesSecurityConstraints$ e que possui um nível inferior a NP-2, denotado pela propriedade $lowerLevelTo$, é um serviço conforme com a política de segurança.

A segunda expressão foi descrita em SWRL como:

- 1 $igov:definesSecurityConstraintsTo(?sl, ?s) \wedge igov:greaterOrEqualLevelTo(?sl,$
- 2 $igov:NP2) \wedge soa:hasInterface(?s, ?si) \wedge igov:accessibleBy(?si, ?pr) \wedge$
- 3 $igov:implements(?pr,$ $igov:ChannelCryptography) \rightarrow$
- 4 $igov:CompliantSecurityService(?s)$

Esta regra é composta por duas partes: a primeira limita seu escopo aos serviços de classificação igual ou superior a NP-2, através dos termos $definesSecurityConstraintsTo$ e $greaterOrEqualLevelTo$, representadas na linha 1; a segunda verifica se cada serviço selecionado pela parte anterior implementa criptografia de canal. Para tal, inicia obtendo a interface do serviço através da propriedade $hasInterface$, seguida pela obtenção do protocolo através da propriedade $accessibleBy$ (linha 2) e verificando se o protocolo implementa criptografia de canal através da propriedade $implements$ (linha 3).

Por fim, os resultados de ambas as regras são combinados na nova classe $CompliantSecurityService$, através da expressão SQWRL $igov:CompliantSecurityService(?s) \rightarrow sqwrl:select(?s)$, permitindo que um serviço seja classificado desta forma atendendo qualquer uma das duas assertivas.

4.4.2.11 Política 4

A política 4 é uma das políticas criadas para lidar com as questões de versionamento de serviços. Seu texto é revisitado a seguir:

“Todas as variantes de uma *release* de serviço devem dispor da mesma interface (informações de entrada e de saída)”

Assim como a política 1, foi gerada uma versão inicial desta regra em SQWRL, descrita a seguir:

```

1  igov:variantOf(?minor, ?major)  $\wedge$  soa:hasInterface(?minor, ?minInt)  $\wedge$ 
2  soa:hasInterface(?major, ?majInt)  $\wedge$  soa:hasInput(?minInt, ?minInput)  $\wedge$ 
3  soa:hasInput(?majInt, ?majInput)  $\wedge$  soa:hasOutput(?minInt, ?minOutput)  $\wedge$ 
4  soa:hasOutput(?majInt, ?majOutput)  $\wedge$  sqwrl:makeSet(?setMinInput, ?minInput)
5   $\wedge$  sqwrl:makeSet(?setMajInput, ?majInput)  $\wedge$  sqwrl:makeSet(?setMajOutput,
6  ?majOutput)  $\wedge$  sqwrl:makeSet(?setMinOutput, ?minOutput)  $\wedge$ 
7  sqwrl:groupBy(?setMajInput, ?minor)  $\wedge$  sqwrl:groupBy(?setMinInput, ?minor)  $\wedge$ 
8  sqwrl:groupBy(?setMajOutput, ?minor)  $\wedge$  sqwrl:groupBy(?setMinOutput,
9  ?minor)  $\wedge$  sqwrl:difference(?setDif1, ?setMajInput, ?setMinInput)  $\wedge$ 
10 sqwrl:difference(?setDif2, ?setMinInput, ?setMajInput)  $\wedge$  sqwrl:size(?tamDif1,
11 ?setDif1)  $\wedge$  sqwrl:difference(?setDif3, ?setMajOutput, ?setMinOutput)  $\wedge$ 
12 sqwrl:difference(?setDif4, ?setMinOutput, ?setMajOutput)  $\wedge$ 
13 sqwrl:size(?tamDif2, ?setDif2)  $\wedge$  sqwrl:size(?tamDif3, ?setDif3)  $\wedge$ 
14 sqwrl:size(?tamDif4, ?setDif4)  $\wedge$  swrlb:equal(?tamDif1, 0)  $\wedge$ 
15 swrlb:equal(?tamDif2, 0)  $\wedge$  swrlb:equal(?tamDif3, 0)  $\wedge$  swrlb:equal(?tamDif4,
16 0)  $\rightarrow$  sqwrl:select(?minor)

```

Esta regra é composta por três blocos: o primeiro, compreendido entre as linhas 1 e 4, seleciona as variantes *minor* de uma *release* denominada *major*, através da propriedade de objeto *variantOf*, e obtém seus respectivos conjuntos de entradas e saídas de dados, *minInput*, *minOutput*, *majInput* e *majOutput*, utilizando as propriedades *hasInterface* para determinar a interface e *hasOutput* e *hasInput* para obter as entradas e saídas através das interfaces. O segundo bloco da regra, compreendido entre as linhas 4 e 9, cria conjuntos de entradas e saídas de dados e os agrupa por variante, utilizando os operadores *makeSet* e *groupBy*, obtendo os conjuntos *setMajInput*, *setMajOutput*, *setMinInput* e *setMinOutput*, que representam as entradas e saídas da *release* relacionada e da variante analisada. Por fim, o terceiro bloco, compreendido entre as linhas 9 e 15, compara os conjuntos de entrada e saída, novamente combinando os operadores *sqwrl:difference* e *sqwrl:size* ao invés de

sqwrl:equal, visando obter todas as *releases* em que os conjuntos de entrada e saída sejam mantidos constantes em cada variante.

Outro ponto de atenção surgiu ao ser implementada esta política. Como o operador *sqwrl:difference* (*dif*, *a*, *b*) preenche como nulo o conjunto *dif* caso o conjunto *b* contenha todos os elementos do conjunto *a* e outros elementos não pertencentes a *a*, tornou-se necessário realizar a diferença alternando os termos, garantindo assim a completa similaridade dos conjuntos. Os termos *sqwrl:difference(?setDif1, ?setMajInput, ?setMinInput)* \wedge *sqwrl:difference(?setDif2, ?setMinInput, ?setMajInput)* (linhas 9 e 10) são exemplos da necessidade de aplicação do operador *sqwrl:difference* com alternância de termos, visando obter apenas os conjuntos completamente semelhantes.

4.4.2.12 Política 5

A política 5 lida com outro aspecto de versionamento, avaliando o comportamento das *releases* em relação aos serviços dos quais são versões. A definição da política é revista a seguir:

“Todas as *releases* de um serviço devem ter interfaces diferentes, ou seja, entrada ou saída distintas.”

Esta política foi transcrita inicialmente para o SQWRL conforme a seguir.

```

1  igov:versionOf(?ver1, ?serv) ∧ igov:versionOf(?ver2, ?serv) ∧
2  soa:hasInterface(?ver1, ?int1) ∧ soa:hasInterface(?ver2, ?int2) ∧
3  soa:hasInput(?int1, ?inp1) ∧ soa:hasInput(?int2, ?inp2) ∧
4  soa:hasOutput(?int2, ?out2) ∧ soa:hasOutput(?int1, ?out1) ∧
5  sqwrl:makeSet(?setIn1, ?inp1) ∧ sqwrl:groupBy(?setIn1, ?ver1) ∧
6  sqwrl:makeSet(?setIn2, ?inp2) ∧ sqwrl:groupBy(?setIn2, ?ver2) ∧
7  sqwrl:makeSet(?setOut1, ?out1) ∧ sqwrl:groupBy(?setOut1, ?ver1) ∧
8  sqwrl:makeSet(?setOut2, ?out2) ∧ sqwrl:groupBy(?setOut2, ?ver2) ∧
9  sqwrl:makeSet(?setVer, ?ver1) ∧ sqwrl:groupBy(?setVer, ?serv) ∧
10 sqwrl:size(?szVer, ?setVer) ∧ sqwrl:difference(?setDif1, ?setIn2, ?setIn1) ∧
11 sqwrl:difference(?setDif2, ?setIn1, ?setIn2) ∧ sqwrl:difference(?setDif3,
12 ?setOut1, ?setOut2) ∧ sqwrl:difference(?setDif4, ?setOut2, ?setOut1) ∧
13 sqwrl:size(?szDif1, ?setDif1) ∧ sqwrl:size(?szDif2, ?setDif2) ∧
14 sqwrl:size(?szDif3, ?setDif3) ∧ sqwrl:size(?szDif4, ?setDif4) ∧ swrlb:divide(?rd,
15 1, ?szVer) ∧ swrlb:add(?r, ?szDif1, ?szDif2, ?szDif3, ?szDif4, ?rd) ∧
16 swrlb:greaterThanOrEqual(?r, 1) → sqwrl:select(?ver1)

```

Esta regra é dividida em 4 partes. A primeira, representada na linha 1 declara e relaciona os elementos *ver1*, *ver2* e *serv*, que representam duas *releases* de serviço e o serviço propriamente dito, utilizando as propriedades *versionOf*. Deste modo, a regra compara duas a duas as *releases* de um serviço. Na segunda, compreendida entre as linhas 2 e 4, são obtidas interfaces *int1* e *int2* das *releases* e suas entradas e saídas, representadas respectivamente por *inp1*, *inp2*, *out1* e *out2*, usando as relações *hasInterface*, *hasInput* e *hasOutput*. Na terceira parte, compreendida entre as linhas 5 e 8, estes elementos são consolidados em conjuntos de entradas e saídas, através de uma sequência de operações *makeSet* e *groupBy*, obtendo os conjuntos *setIn1* e *setIn2* relativos às entradas e *setOut1* e *setOut2*, relativos às saídas. Por fim, entre as linhas 9 e 16 são realizadas as comparações para verificar a conformidade.

Inicialmente, para realizar a comparação, devido aos problemas já citados nas políticas 1 e 4 referentes aos operados *sqwrl:equal*, foi realizada a diferença e calculado o tamanho de cada resultado, usando operações *sqwrl:difference* e *sqwrl:size*, representadas entre as linhas 10 e 14. Porém, nesta política, como estamos interessados em classificar como conformes serviços que utilizem entradas ou saídas diferentes, foi

necessário verificar se o tamanho da diferença entre os conjuntos era maior que 0. Adicionalmente, como qualquer diferença tornaria o serviço conforme, seja na entrada ou saída, foi necessário verificar se qualquer uma das quatro possíveis diferenças resultou em valor positivo. Para tal, foi incluído ao final da expressão um termo *swrlb:add* para somar os valores obtidos e o resultado foi testado para verificar se era maior ou igual a 1, usando o operador *swrlb:greaterThanOrEqual*.

Os termos da linha 9, para montagem de um conjunto de *releases* de um serviço e a operação de divisão ao final da linha 14 foram incluídos devido a uma condição capturada nos testes. Foi verificado que a consulta se comportava de maneira adequada para serviços com duas ou mais *releases*. Porém ao testar com um serviço com uma única *release*, a consulta não operava de maneira satisfatória, uma vez que o resultado sempre era diferença nula, o que era originário da comparação da única interface existente com ela mesma.

Para resolver esta questão, tornou-se necessária a inclusão dos novos termos. Inicialmente, é obtido um conjunto contendo todas as *releases*, agrupadas por serviço e é calculado o seu tamanho, utilizando as expressões a seguir, obtidas da linha 9

$$sqwrl:makeSet(?setVer, \quad ?ver1) \quad \wedge \quad sqwrl:groupBy(?setVer, \quad ?serv) \quad \wedge \quad sqwrl:size(?szVer, ?setVer)$$

Em seguida, é verificado se o serviço possui apenas uma *release*. É importante observar que não podemos simplesmente usar um operador de comparação em relação a tamanho, como o *swrlb:greaterThanOrEqual*, pois estes eliminariam serviços com apenas uma *release* do resultado da consulta. Para resolver esta questão, foi utilizado um operador de divisão inteira *swrlb:divide* para obter o valor 1 caso o serviço obtivesse apenas uma *release* ou 0 se obtivesse mais de uma e o resultado desta operação foi acrescentado à operação de soma utilizada anteriormente, fazendo com que o resultado desta soma fosse sempre 1 ao se utilizar uma única *release*. Estes pontos são apresentados na expressão a seguir, obtida das linhas 14 e 15:

$$swrlb:divide(?rd, 1, ?szVer) \wedge swrlb:add(?r, ?szDif1, ?szDif2, ?szDif3, ?szDif4, ?rd)$$

4.4.2.13 Política 6

A política 6 trata outro aspecto de versionamento, descrito pelo texto a seguir:

“Somente uma variante de cada *release* de serviço deve estar ativa no barramento de serviços.”

Esta política foi transcrita para SQWRL conforma expressão a seguir.

```
1   igov:variantOf(?var, ?ver)  $\wedge$  igov:hasState(?var, igov:Operational)  $\wedge$   
2   sqwrl:makeSet(?setVar, ?var)  $\wedge$  sqwrl:groupBy(?setVar, ?ver)  $\wedge$   
3   sqwrl:size(?szVar, ?setVar)  $\wedge$  swrlb:equal(?szVar, 1)  $\rightarrow$  sqwrl:select(?ver, ?var)
```

Inicialmente, na linha 1, são obtidas as variantes *var* de uma *release* de serviço *ver* através da propriedade *variantOf* e são obtidos as variantes que encontram em estado Ativo, denotado por *Operational*. Na sequência, na linha 2, as instâncias de *releases* ativas são incluídas em um conjunto denominado *setVar* e agrupadas por *release* de serviço. Deste modo, são segmentados conjuntos correspondendo ao conjunto de variantes ativas por *release*. Ao final, na linha 3, são calculados os tamanhos destes conjuntos e verificados se estes são iguais a 1. Tamanhos de conjuntos maiores que 1 denotam mais de uma variante de uma *release* em estado ativo, tornando-a não conforme. Deste modo, a consulta retorna todas as variantes de *releases* que se encontram conformes com a política.

4.4.2.14 Política 7

A política 7 trata questões de versionamento relacionadas com o comportamento de *releases* em relação aos contratos de serviços firmados entre provedores e consumidores. O texto que a descreve é revisitado a seguir.

“Toda *release* de serviço que possua um contrato ativo deve estar disponível para uso no barramento de serviços.”

Para facilitar a implementação desta regra em SQWRL, esta foi dividida em dois blocos: uma regra SWRL que define contratos inativos e a consulta que retorna as instâncias conformes.

A regra SWRL pode ser vista a seguir.

```
soa:ServiceContract(?sc)  $\wedge$  igov:endDate(?sc, ?dt)  $\wedge$  temporal:after("now", ?dt)  $\rightarrow$   
igov:InactiveContract(?sc)
```

A regra obtém as datas de término *dt* dos contratos *sc*, através da propriedade *endDate* e utiliza o operador *temporal:after* para verificar se o contrato possui data de término preenchida e anterior à data corrente, tornando-o encerrado.

Já a consulta é descrita pela seguinte expressão SQWRL:

```

1  igov:hasState(?serv, igov:Operational)  $\wedge$  soa:isContractFor(?sc, ?serv)  $\wedge$ 
2  igov:InactiveContract(?isc)  $\wedge$  sqwrl:makeSet(?setCont, ?sc)  $\wedge$ 
3  sqwrl:groupBy(?setCont, ?serv)  $\wedge$  sqwrl:makeSet(?setInat, ?isc)  $\wedge$ 
4  sqwrl:difference(?setAtiv, ?setCont, ?setInat)  $\wedge$  sqwrl:size(?szAt, ?setAtiv)  $\wedge$ 
5  swrlb:greaterThan(?szAt, 0)  $\rightarrow$  sqwrl:select(?serv)

```

Inicialmente, na linha 1, são obtidos os serviços que se encontram operacionais, através da propriedade *hasState* com valor *Operational*, e são obtidos os contratos destes serviços. Todos os contratos inativos são obtidos através da declaração *igov:InactiveContract*, na linha 2, que obtém os contratos inativos classificados pela regra SWRL que apoia esta consulta. São criados então dois conjuntos, um para contratos do serviço (*setCont*) agrupado por serviços, outro para contratos inativos (*setInat*), nas linhas 2 e 3. É realizada a diferença entre estes dois conjuntos (*setCont* menos *setInat*) na linha 4. Seu tamanho é medido e é verificado se este é maior que 0 nas linhas 4 e 5. Caso positivo, implica na existência de pelo menos um contrato ativo, tornando o serviço conforme.

4.4.3 Avaliação usando o intelliGOV

Por fim, a avaliação foi executada utilizando o intelliGOV. Devido à quantidade de conjuntos distintos envolvidos nas regras 4 e 5, a execução destas levou a exceções de estouro de memória ao ser executada utilizando dados reais. Para resolver este problema, foi incluído em cada uma destas regras uma assertiva inicial *sameAs(?s, [instância de serviço])*, que permitia parametrizar a consulta. Deste modo, a política passou a ser verificada serviço a serviço, minimizando a carga de processamento necessária.

Para medir o tempo, foram registrados os instantes iniciais e finais de avaliação de cada política e calculada a diferença.

4.4.4 Discussão dos resultados

Os resultados obtidos nas duas avaliações foram apresentados para a equipe que participou do estudo de caso, com o objetivo de verificar os erros detectados, entender as justificativas apresentadas e avaliar as diferenças entre avaliações realizadas manualmente e com o uso da abordagem proposta.

Para tal, foram apresentadas as planilhas de avaliação e realizadas perguntas para esclarecer as justificativas que não foram compreendidas, apresentado o resultado obtido com o intelliGOV e os pontos de diferença obtidos, visando obter os motivos que levaram as discrepâncias entre os resultados.

Finalizada esta discussão, o autor consolidou os dados obtidos e os pontos de atenção identificados, para realizar a etapa de análise dos resultados. Dada a importância e tamanho desta atividade, sua descrição é apresentada no capítulo 5.

5. Análise dos Resultados

Neste capítulo é apresentada a avaliação dos resultados obtidos durante a execução do estudo de caso. Para tal, inicialmente são apresentados os resultados obtidos nas avaliações manuais e com o uso da abordagem e a comparação entre estes dados. Por fim, são analisadas as limitações da análise e resumidos os principais pontos.

5.1 Análise das avaliações

Nesta seção são apresentados os resultados obtidos ao longo da execução do estudo de caso. Um detalhamento da avaliação de cada política pode ser visto no Apêndice III.

No contexto deste estudo de caso, a qualidade das medições é obtida através da contagem de divergências entre os resultados das avaliações e o resultado esperado no gabarito. Por exemplo, se o resultado da avaliação de um serviço relativo a primeira política é conforme e no gabarito se espera um resultado não conforme, temos uma divergência. Cada divergência identificada foi contabilizada como um erro de verificação. Para cada política, as divergências foram contabilizadas e agrupadas. Esta informação é reproduzida na Tabela 5. A primeira coluna identifica a política, a segunda a quantidade de erros de verificação obtida na avaliação manual, a terceira o percentual destes erros de verificação em relação ao total de avaliações, a quarta a quantidade de erros de verificação obtida com o uso de intelliGOV e a última o percentual destes erros em relação ao total de avaliações.

Tabela 5. Comparação dos erros de verificação obtidos com cada abordagem

Política	Avaliação Manual		Avaliação com intelliGOV	
	Quantidade de Erros	Percentual de Erros	Quantidade de Erros	Percentual de Erros
1	6	24%	0	0%
2	8	32%	0	0%
3	3	12%	0	0%
4	3	12%	0	0%
5	8	32%	0	0%
6	3	12%	0	0%
7	6	24%	0	0%

Assim, observa-se uma média de 21% de erros obtidos na avaliação manual, e zero utilizando o intelliGOV. Tal resultado se torna possível devido ao tratamento de diversas situações que levaram a erros, descritas nas subseções a seguir.

5.1.1 Política 1

Para a política 1, que lida com a classificação em áreas de assunto dos serviços coerente com a classificação das informações, foram identificados cinco casos de falso positivo (analista indica como conforme um serviço que não é conforme) e um caso de falso negativo (analista indica como não conforme um serviço conforme), conforme pode ser visto na seção V.1.

Neste contexto, os analistas reportaram dificuldade em identificar a área a qual o dado pertencia, principalmente nos pontos em que a classificação proposta pela organização divergia de conceitos julgados sob uma perspectiva diferente pelo analista. Um exemplo é a avaliação do analista para o serviço COMPANY. Traduzindo para o Português, o analista julgou que se tratava de um serviço relativo a empresas, igualando este conceito ao conceito de fornecedores e julgando correto o fato de que este serviço estaria manipulando dados da área SUPPLY (suprimentos). Porém, para a organização, COMPANY significa empresa subsidiária, conceito gerido pela área de assunto CORP-ORG, que lida com os dados de organização e estrutura da empresa.

Outro ponto que confundiu os analistas é relativo à classificação dos dados quando o serviço manipula informações de mais de uma área de assunto. Um exemplo é o do serviço STOCKS, classificado na área DWNSTR e que manipula informações das áreas DWNSTR-PRODUCT (entrada) e DWNSTR-LOG (saída). Neste caso, devido aos nomes diferentes, o analista considerou o caso não conforme. Porém, DWNSTR-

PRODUCT e DWNSTR-LOG são subáreas de assunto da área DWNSTR, o que implicaria em um caso conforme.

Com o uso de intelliGOV, estes erros foram evitados utilizando a propriedade transitiva *handlesDataFrom*, que permite que um serviço, associado a uma área de assunto, também esteja associado às subáreas desta área.

No exemplo citado, o serviço STOCKS tem uma propriedade de objeto *handlesDataFrom* associando-o com a área de assunto DWNSTR. Esta área por sua vez, também se relaciona as áreas DWNSTR-PRODUCT e DWNSTR-LOG através da mesma propriedade de objeto. Por transitividade, STOCKS se associa a estas mesmas áreas, tornando-o conforme.

5.1.2 Políticas 2 e 3

Para a política 2, que exige relação do serviço com um nível de segurança de informação válido, foram identificados quatro falsos positivos e quatro falsos negativos, conforme pode ser visto na seção V.2.

Neste cenário, os analistas com menor experiência na empresa não levaram em conta os níveis e classificaram como não conformes serviços que estavam classificados de acordo com o modelo legado. Um exemplo é o serviço BIGE, considerado pelo analista como não conforme por estar classificado como Reservado. Porém, este nível pode ser mapeado diretamente para o nível NP-2 da nova classificação, conforme visto na Tabela 3, levando a um resultado conforme.

Um ponto de discussão ocorreu por conta do serviço CATPL, classificado como *Public*. Uma vez que a empresa atua globalmente. O redator do gabarito informou que mesmo com atuação global, a orientação da empresa é o uso do idioma português na elaboração de documentação, critério este que levou à classificação como não conforme. O autor informou que, caso se julgasse necessário, novas instâncias de *SecurityLevel* representando os níveis em outros idiomas poderiam ser cadastradas, efetuando-se a associação entre estas novas instâncias e as instâncias em Português, possibilitando que este cenário fosse classificado como conforme, caso a organização assim desejasse.

Para a política 3, que trata a aplicação de mecanismos de criptografia de canal para serviços com nível de segurança maior ou igual a NP-2, foram identificados três falsos positivos e nenhum falso negativo, conforme pode ser visto na seção V.3.

Neste caso, os analistas novamente confundiram o mapeamento entre níveis, considerando nível *Reservado* ou sua abreviatura *Res* como níveis que não demandavam criptografia de canal.

Ao aplicar o intelliGOV, os serviços foram classificados corretamente, uma vez que ao serem criadas as instâncias representando os níveis de segurança de informação, foi utilizada uma propriedade do tipo *sameAs* para relacionar os níveis de acordo com a Tabela 3. Deste modo, no exemplo citado para o serviço BIGE, o nível Reservado é considerado como análogo ao nível NP-2, tratando a instância como conforme. Adicionalmente, caso a organização deseje, é possível criar instâncias representando abreviaturas válidas e nomes dos níveis em outros idiomas, visando contornar os problemas identificados pelos analistas.

5.1.3 Políticas 4 a 7

Para as políticas compreendidas entre 4 e 7, os analistas reportaram dificuldade em identificar quando uma mudança de versão deveria originar uma variante ou uma *release*. Tal fato levou a 3 falsos positivos e 1 falso negativo para a política 4, sete falsos positivos e 1 falso negativo para a política 5, um falso positivo e dois falsos negativos para a política 6 e cinco falsos positivos e um falso negativo para a política 6, conforme pode ser visto respectivamente nas seções V.4, V.5, V.6 e V.7.

Alguns exemplos de problemas identificados nestes cenários:

- Alterações somente em campos de documentação dos XMLs que representavam as interfaces dos serviços foram interpretadas incorretamente como mudanças na interface, devido ao uso de ferramentas de verificação de diferenças na plataforma de versionamento utilizada. Ao identificarem que existia uma diferença nos arquivos, os analistas julgaram que estas sempre levavam a uma alteração de interface;
- Em contrapartida, alterações nos nomes dos *namespaces* não foram consideradas por alguns analistas como alterações de interfaces, uma vez que estes julgaram que os elementos que compõem a interface permaneceram iguais. No entanto, alterações de *namespace* configuram alterações de interfaces;

- Alterações na organização das importações dos arquivos XML que compõem as interfaces que não impactavam na interface final também foram consideradas incorretamente como alterações nas interfaces;
- *Releases* duplicadas devido a erros de implantação foram julgadas incorretamente como conformes pelo analista, que julgou apenas uma das *releases* devido a existência de documentação no repositório e código no repositório de versionamento relativos a uma única *release*.

Para a política 7, em especial, vale ressaltar que alguns dos analistas ignoraram que os contratos de serviços na organização possuíam datas de encerramento, que representam o momento em que estes não são mais válidos. Um exemplo é o serviço BIGE, que possui duas *releases*, sendo uma ativa e outra inativa, e dois contratos de consumo: um ativo, relacionado a *release* mais recente; e outro inativo, relacionado a *release* mais antiga, ambos relacionados ao mesmo consumidor do serviço. Por olhar apenas os contratos e não considerar a data de inativação de um deles, o desenvolvedor considerou que existiam dois contratos ativos e julgou que ambas as *releases* deveriam estar ativas. Porém, o analista desconsiderou o fato de que ocorreu uma migração de uso de versões pelo consumidor do serviço, através do encerramento de um contrato e abertura de outro, o que tornava o cenário conforme, uma vez que a *release* antiga pode ser descontinuada.

Com o uso de intelliGOV, os diagnósticos foram novamente corretos. A solução conseguiu separar os pontos que implicavam em interfaces divergentes, como alteração de nomes ou *namespaces* de campos ou inclusão e exclusão de campos, de alterações que não levariam a alteração de dados, como alteração da documentação dos esquemas XML associados. Além disso, a solução conseguiu identificar corretamente os contratos ativos e inativos, considerando todos os elementos necessários para julgar os serviços e contratos.

5.2 Avaliação do esforço

Nesta seção tratamos a questão do esforço gasto na avaliação. A Tabela 6 apresenta o esforço em segundos dispendido para realizar as avaliações, divididos por políticas. A primeira coluna identifica a política, a segunda o tempo total gasto para realizar a avaliação manual considerando todos os 25 serviços. A terceira coluna

apresenta o tempo médio gasto para avaliar um serviço. A quarta coluna apresenta o tempo total gasto para realizar a avaliação usando intelliGOV. A quinta coluna apresenta o tempo médio por serviço para realizar a avaliação com intelliGOV.

Tabela 6. Esforço gasto nas avaliações manuais e com intelliGOV

Política	Avaliação Manual		Avaliação com intelliGOV	
	Tempo de Avaliação (s)	Tempo médio de avaliação por serviço (s)	Tempo de Avaliação (s)	Tempo médio de avaliação por serviço (s)
1	570	23	3,7	0,1
2	405	16	2,7	0,1
3	445	18	3,4	0,1
4	400	16	72,5	2,9
5	365	15	70	2,8
6	305	12	2,8	0,1
7	315	13	2,8	0,1

Como esperado no uso de uma solução automatizada, o esforço de execução da avaliação é reduzido. É importante observar que as políticas 4 e 5 oneraram maior tempo de execução devido a necessidade de executar suas verificações serviço a serviço, ao invés de uma única verificação para todos os serviços envolvidos na arquitetura.

5.3 Discussão dos Resultados

Para debater o resultado das avaliações, cabe iniciar discutindo o resultado das avaliações de acordo com a experiência dos analistas que participaram do estudo de caso. Para tal, os analistas foram classificados em níveis de experiência descritos na Tabela 7, considerando duas classificações: uma baseada em experiência na organização e outra baseada em experiência em SOA.

Tabela 7. Níveis de experiência considerados neste trabalho

Nome	Sigla	Faixa de Experiência
Junior	Jr	Menor que 3 anos
Pleno	Pl	Entre 3 a 5 anos
Sênior	Sr	Maior que 5 anos

Uma vez definidos estes grupos, a quantidade de erros identificados foi distribuída por estes níveis de experiência. O resultado considerando a experiência em SOA é apresentado na Figura 17. É possível ver que Analistas Plenos obtiveram um volume de divergências na avaliação maior que os Analistas Junior, levando a conclusão que um analista mais experiente em SOA não implica em uma maior precisão na análise.

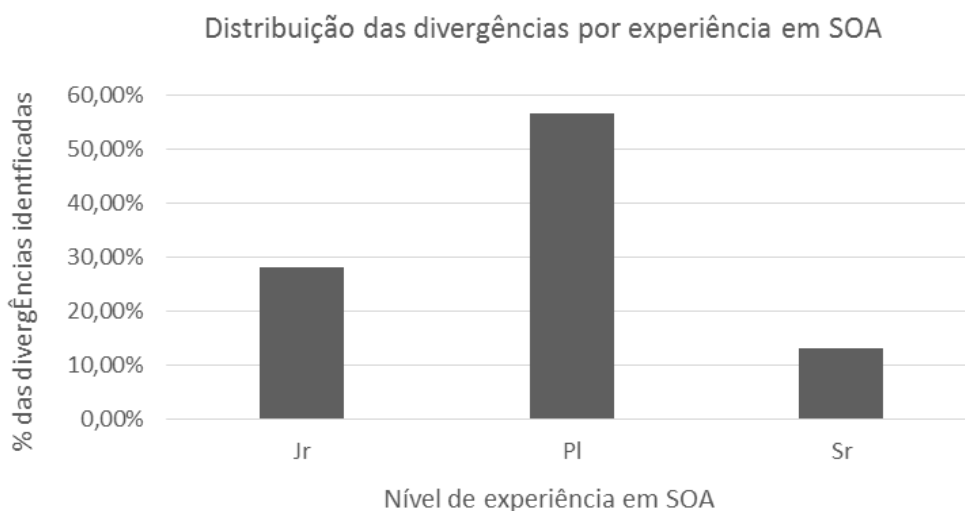


Figura 17. Distribuição das divergências em relação a experiência em SOA

A Figura 18 apresenta o resultado considerando a experiência dos analistas na organização, mostrando que a quantidade de erros decaiu com o aumento de experiência. Tal fato também pode ser derivado analisando a quantidade total de erros de avaliação da política. Políticas que dependiam de maior conhecimento de aspectos da organização, como a classificação de informações (política 1), definição dos níveis de segurança de informação (política 2) e definição do modelo de *releases* de versionamento (políticas 5 e 7) obtiveram um percentual de erros de avaliação maior que as outras políticas, que dependiam de fatores técnicos. Mesmo a política 3, que dependia de fatores técnicos como protocolos e mecanismos de segurança, obteve erros devido a interpretação errada de um fator do negócio relativo a forma como a organização classifica as informações.



Figura 18. Distribuição das divergências em relação a experiência na organização

Deste modo, podemos observar que, independente do perfil profissional, a solução intelliGOV apresenta ganhos em termos de qualidade em relação a qualquer perfil alocado para realizar a atividade de verificação de conformidade.

Mesmo com esta redução, pode se questionar que o percentual de erros originados pelo trabalho de um analista sênior seria baixo o suficiente para tornar a solução sem utilidade no caso do uso exclusivo deste perfil profissional para realizar a avaliação.

Para analisar este caso, foi calculado o tempo médio gasto por cada analista para realizar a avaliação de um serviço em relação a uma política. Os resultados são apresentados na Tabela 8, na qual a primeira coluna indica o analista, a segunda e a terceira os níveis de experiência em SOA e na organização, respectivamente, e a quarta o tempo médio.

Tabela 8. Esforço gasto por cada analista

Analista	Nível de experiência em SOA	Nível de experiência na organização	Tempo Médio de Avaliação por Serviço (s)
1	Jr	Jr	103
2	Pl	Jr	108
3	Pl	Sr	143
4	Jr	Pl	72
5	Sr	Jr	150
6	Pl	Jr	94
7	Pl	Pl	135

Os dois itens marcados em cinza correspondem aos dois analistas classificados como Seniores em SOA ou na organização. Podemos observar que para estes casos, foi gasto mais esforço para realizar a avaliação. Ao serem questionados sobre este ponto, os analistas explicaram o procedimento que utilizaram para realizar suas verificações manualmente e foi percebido que estes buscavam mais informações e detalhes, sendo mais meticolosos em suas avaliações. Deste modo, podemos ver que neste cenário, uma analista sênior consome mais tempo realizando uma avaliação manual que os outros analistas.

Considerando os números obtidos no estudo de caso e empregando um analista sênior para executar as atividades, seriam necessárias em torno de 150 segundos para avaliar cada serviço no contexto de cada política. Considerando apenas o cenário tratado no estudo de caso, estaríamos considerando um esforço total de 25200 segundos ou 7 horas de trabalho.

Se extrapolarmos este valor para o uso de todos os processos descritos no commonGOV, considerando uma média de duas políticas por processo e avaliarmos todo o conjunto de serviços da organização, o esforço necessário para executar uma avaliação completa é ampliado para 502 horas.

Para realizar uma comparação com a abordagem intelliGOV, é necessário considerar não apenas o esforço de execução das verificações de conformidade usando a abordagem, mas também o esforço de implementação da ontologia, regras e cargas.

Para uma estimativa deste esforço, foram considerados os métodos ONTOCOM (SIMPERL, TEMPICH e SURE, 2006) para estimar o esforço para implementação da ontologia e pontos de função (GARMUS e HERRON, 2000) para especificação das regras e implementação dos mecanismos de carga, totalizando 3480 horas. O cálculo detalhado é apresentado no Apêndice VI, considerando uma extrapolação do cenário tratado no estudo de caso para alcançar uma estimativa corporativa.

Deste modo, considerando o esforço indicado anteriormente de 502 horas de trabalho para execução da avaliação manualmente pelos analistas, podemos observar que este esforço se paga em aproximadamente 7 avaliações. Um ponto a ser questionado poderia ser a necessidade de retrabalho devido a mudanças constantes nas políticas, cenário esperado em grandes organizações. Porém, conforme visto no Apêndice VI, apenas 15,2% deste esforço é dispendido em especificação de regras, correspondendo a cerca de 530 horas. Em um cenário pessimista, no qual se tornasse necessário alterar todas as políticas, este custo seria compensado em pouco mais de uma avaliação.

É importante observar que esta é uma estimativa aproximada representando um cenário pessimista, uma vez que foi considerado que o analista sênior seria o responsável por todas as atividades, possuindo baixa experiência no campo de ontologias. Tal número pode ser reduzido devido a divisão destas tarefas entre os especialistas na organização e especialistas em ontologias. Adicionalmente, estes números levam em conta um cenário estimado de conceitos para uma ontologia corporativa, podendo ser reduzido para os elementos tratados no contexto SOA.

Deste modo, é possível observar que a abordagem proposta origina ganhos de qualidade através da redução de erros de avaliação, com indícios de reduções de custos para as organizações, através da redução da necessidade de participação de analistas sêniores na execução das avaliações. O trabalho do analista sênior passa a ter foco na especificação das regras e conceitos a serem utilizados na verificação de conformidade.

Estes ganhos de qualidade são obtidos devido a explicitação e formalização de dois tipos de conhecimento: os que são específicos da organização, como as áreas de assunto e níveis de segurança tratados no estudo de caso; e os que exigem conhecimento técnico mais apurado, como as situações em que é necessário diagnosticar se uma interface foi alterada ou não. Em ambos os casos, a abordagem proposta conseguiu compreender este conhecimento e tomar uma decisão precisa quanto a situação de conformidade dos elementos avaliados, consumindo um tempo de processamento que permite seu uso contínuo, elevando o controle do ambiente SOA das organizações.

5.4 Ameaças à validade

Nesta seção, são apresentados e debatidos fatores e variáveis que poderiam comprometer os resultados desta avaliação. Para tal, foram consideradas três dimensões: tecnologia, método e representatividade dos dados.

5.4.1 Tecnologia

Um dos pontos que poderia comprometer a eficácia dos resultados do intelliGOV seria a seleção inadequada de tecnologias para implementação. Existem diversas tecnologias para implementação dos módulos propostos neste trabalho e deve ser avaliada a possibilidade de uma seleção incorreta invalidar os resultados do trabalho proposto.

Conforme pode ser percebido na análise da seção 5.1, a capacidade de utilizar recursos disponíveis na OWL para tratar transitividade, composições e heranças tornou

a solução eficaz, capturando elementos que seriam de difícil implementação em outras tecnologias. Outras tecnologias para implementação de ontologias poderiam apoiar este trabalho, porém para fins de validação, a OWL mostrou-se eficaz no cenário avaliado. Além disso, cabe ressaltar que a OWL é a linguagem recomendada pela W3C para descrição de ontologias.

Outro ponto é a seleção de tecnologia para escrita de regras e políticas. Também foi possível perceber que a combinação entre SWRL e SQWRL conseguiu atingir os objetivos. O uso de tecnologias que permitam contornar as limitações do SQWRL poderiam apenas melhorar a solução e não inviabilizá-la. Adicionalmente, vale ressaltar que ambas as tecnologias são compatíveis com a OWL, sem necessidade de implementações ou customizações adicionais.

5.4.2 Método

Outro ponto que poderia inviabilizar o estudo seria uma decisão incorreta quanto aos métodos de desenvolvimento e a ontologia utilizada como base ao executar o estudo de caso.

Um ponto a ser considerado foi o uso da metodologia 101 para modelagem da ontologia, ao invés de métodos com maior volume de atividades e maior complexidade de execução, como METHONTOLOGY (FERNÁNDEZ-LÓPEZ, GÓMEZ-PÉREZ E JURISTO, 1997) ou On-To-Knowledge (SURE, STAAB e STUDER, 2004). A seleção de um método mais completo permitiria a constituição de uma ontologia mais detalhada e precisa, o que melhoraria os resultados obtidos, podendo levar a uma melhor descrição do domínio ou redução da complexidade das expressões que representam as políticas.

Outra questão é a seleção da ontologia utilizada como base e o possível uso de outras ontologias para compor o modelo utilizado no estudo de caso. Uma avaliação e seleção mais detalhada de ontologias contribuiriam para melhorar a solução, de maneira semelhante ao uso de uma metodologia mais complexa, sem comprometer os resultados obtidos com este trabalho.

Outro ponto é a validação da ontologia. Neste trabalho, apenas um analista realizou a validação dos elementos que compõem a ontologia utilizada. Porém, poderia se questionar se esta validação seria suficiente. Seguindo esta linha, uma validação mais rigorosa permitiria uma ontologia com maior qualidade, que melhoraria os resultados.

Por fim, pode ser questionada a medida de esforços e custos tratada na seção 5.3. Para estimar o esforço a ser empreendido para a construção do ambiente de execução do

intelliGOV, foi utilizada uma combinação de técnicas fundamentadas, e foram consideradas premissas para extrapolação que consideraram um crescimento linear dos elementos envolvidos. Porém esta combinação carece de uma calibragem mais precisa.

5.4.3 Representatividade dos dados

Por fim, outro ponto que poderia comprometer o resultado deste trabalho seria o uso de um conjunto de dados inadequado, que não representasse um domínio usual de SOA.

Neste contexto, o estudo de caso foi executado em uma empresa que dispõe de um amplo leque de tecnologias para implementação de serviços. Adicionalmente, os envolvidos no estudo de caso possuíam perfil variado, permitindo uma variedade de comportamentos esperada em uma organização de porte mundial.

Adicionalmente, mesmo não tratando todos os serviços disponíveis na organização, o estudo de caso conseguiu lidar com serviços representando os principais grupos de tecnologia existentes na organização – ABAP, Java, .NET, Notes e serviços externos a organização – permitindo assim uma avaliação que minimiza contaminações devido a aspectos exclusivos de alguma das tecnologias indicadas.

Mesmo assim, dados o baixo número de avaliações, existem limitações ao se extrapolar o resultado para qualquer organização que utilize SOA, permitindo apenas uma análise qualitativa. Para tornar esta análise mais quantitativa, seria necessário executar esta mesma proposta em outras organizações, com maior diversidade de profissionais.

6. Comparação com outras abordagens

Conforme visto na seção 2.3, dada a criticidade do problema, diversos autores realizaram propostas para facilitar a especificação e verificação de conformidade no contexto SOA. Neste capítulo, serão discutidas as suas diferenças em relação à proposta desta dissertação. Em alguns casos, para tornar mais claras estas diferenças, protótipos utilizando ferramentas que implementam a técnica utilizada pelas soluções foram desenvolvidos para instanciar e mensurar estas diferenças. Ao final do capítulo, esta análise é consolidada mostrando os ganhos e limitações do intelliGOV em relação a estas abordagens.

6.1 Tecnologias propostas nos trabalhos relacionados

Para comparar os trabalhos relacionados listados no Capítulo 2 com a abordagem proposta nesta dissertação, foram elencadas as tecnologias que compõem as abordagens e avaliados os ganhos e limitações do intelliGOV em relação a estas possíveis formas de implementação. Como algumas abordagens utilizam múltiplos métodos, estas são citadas em mais de um grupo. Deste modo, as abordagens foram divididas nos blocos apresentados na Tabela 9, sendo que a primeira coluna representa a tecnologia, a segunda os trabalhos que a utilizam. Em alguns casos, como as limitações em relação ao intelliGOV não eram tão claras, foram implementados protótipos para melhor identificar e avaliar as diferenças.

Tabela 9. Grupos de tecnologias comparados

TECNOLOGIA	TRABALHOS RELACIONADOS
Bases de Dados Relacionais	PENG, LUI e CHEN (2008)
Processamento de Eventos	BIRUKOU <i>et al.</i> (2010) e TRAN <i>et al.</i> (2011)
MDA (Model-Driven Architecture)	TRAN <i>et al.</i> (2011) e TRAN <i>et al.</i> (2012)
Data Warehouse	RODRÍGUEZ <i>et al.</i> (2013) e TRAN <i>et al.</i> , (2011)
Web Semântica	SPIES (2012) e ZHOU <i>et al.</i> (2010)

6.2 Comparação com o uso de base de dados

Uma das possíveis soluções para o problema de conformidade é o uso de uma base de dados para registrar o *status* da arquitetura e a implementação de consultas para verificar o estado de conformidade. Este tipo de abordagem é proposto por PENG, LUI e CHEN (2008).

Para comparar abordagens baseadas em bases relacionais, como a proposta de PENG, LUI e CHEN (2008), com o intelliGOV, foi implementado um protótipo em SGBD relacional no qual foi construído um banco de dados que implementasse uma visão relacional da ontologia representada na Figura 15

Este banco foi implementado utilizando tecnologia ORACLE EXADATA²¹, um SGBD compatível com os padrões SQL e otimizado para manter os dados integralmente em memória visando ganhos de desempenho. Os dados utilizados para o estudo de caso foram carregados neste ambiente e consultas SQL foram redigidas para expressar as políticas. Um desenho do modelo do banco pode ser visto na Figura 19. Os nomes das tabelas foram baseados nos nomes das classes e propriedades da ontologia utilizada no estudo de caso e para derivar relacionamentos e tratar a implementação de especializações, foram utilizadas técnicas recomendadas por HEUSER (2009).

²¹ <http://www.oracle.com/us/products/database/exadata/overview/index.html>

destes elementos, tornando a manutenção deste modelo mais complexa. O mesmo vale para a tabela HANDLES_DATA_FROM, que corresponde à propriedade de objeto transitiva homônima;

- As relações definindo subclasses entre variantes, *releases* e serviços foram consolidadas em uma tabela de serviços, possuindo um campo *SVR_TYPE* para informar a categoria e duas chaves estrangeiras *SVR_PARENT_ID* e *SVR_RELEASE_ID* que permitem referenciar o serviço do qual uma *release* é derivada e a *release* da qual uma variante é derivada respectivamente. A opção pela consolidação visa facilitar a relação destes elementos com as interfaces de serviço;
- Já a especialização de serviço e área de assunto a partir de elemento ficou mantida em uma tabela à parte, visando garantir que as relações destas especializações com outros itens (área de assunto com tipos de informação e serviço com interfaces e restrições de segurança) ficassem devidamente explicitadas.

Tabela 10. Exemplo de preenchimento da relação GREATER_EQUAL_THAN, considerando uma expressão Nível B >= Nível A

Nível A	Nível B
NP-2	NP-2
NP-2	NP-3
NP-2	NP-4

- O mesmo raciocínio se emprega para expressar a relação *sameAs* que representa a equivalência entre os níveis de segurança apresentada na Tabela 3. Foi necessário criar uma tabela SAME_AS para representar a relação entre os níveis.

Com isso, termos que não precisariam ser explicitados em uma regra graças a transacionalidade ou a relações de igualdade na abordagem intelliGOV tiveram que se tornar explícitos nas declarações das consultas SQL. Um exemplo pode ser visto a seguir, com a consulta SQL que representa a política 2.


```

SELECT      S.SRV_NAME, SI.SVRI_NAME, SL.SLVL_NAME
FROM        SERVICE S, SERVICE_INTERFACE SI, SECURITY_LEVEL SL,
           HAS_INTERFACE HI, DEFINES_SEC_CONSTRAINTS SC,
           GREATER_EQUAL_THAN GT, SECURITY_LEVEL GL,
           ACCESIBLE_BY AC, COMMUNICATION_PROTOCOL CP,
           IMPLEMENTS IM, SECURITY_MECHANISM SM
WHERE       S.SRV_ID = HI.HITF_SVR_ID AND
           HI.HITF_SVRI_ID = SI.SVRI_ID AND
           AC.ACC_SVRI_ID = SI.SVRI_ID AND
           AC.ACC_CMPT_ID = CP.CMPT_ID AND
           IM.IMP_CMCT_ID = CP.CMPT_ID AND
           IM.IMP_SMEC_ID = SM.SMEC_ID AND
           SM.SMEC_NAME = 'Channel Criptography' AND
           S.SRV_ID = SC.SCON_SVR_ID AND
           SC.SCON_SLVL_ID = SL.SLVL_ID AND
           SL.SLVL_ID = GT.GET_ID_HIGH AND
           GT.GET_ID_LOW = GL.SLVL_ID AND
           (GL.SLVL_NAME = 'NP-2' OR
           (GL.SLVL_NAME IN (SELECT S1.SLVL_NAME      FROM
           SECURITY_LEVEL S1, SECURITY_LEVEL S2, SAME_AS WHERE
           SMA_SOURCE_ID = S1.SLVL_ID AND SMA_TARGET_ID = S2.SLVL_ID
           AND S2.SLVL_NAME = 'NP-2')))

UNION
SELECT      S.SRV_NAME, SI.SVRI_NAME, SL.SLVL_NAME
FROM        SERVICE S, SERVICE_INTERFACE SI, SECURITY_LEVEL SL,
           HAS_INTERFACE HI, DEFINES_SEC_CONSTRAINTS SC,
           LOWER_THAN LT, SECURITY_LEVEL LL
WHERE       S.SRV_ID = HI.HITF_SVR_ID AND
           HI.HITF_SVRI_ID = SI.SVRI_ID AND
           S.SRV_ID = SC.SCON_SVR_ID AND
           SC.SCON_SLVL_ID = SL.SLVL_ID AND
           SL.SLVL_ID = LT.LWT_LOW_ID AND
           LT.LWT_HIGH_ID = LL.SLVL_ID AND
           (LL.SLVL_NAME = 'NP-2' OR
           (LL.SLVL_NAME IN (SELECT S1.SLVL_NAME      FROM
           SECURITY_LEVEL S1, SECURITY_LEVEL S2, SAME_AS WHERE
           SMA_SOURCE_ID = S1.SLVL_ID AND SMA_TARGET_ID = S2.SLVL_ID
           AND S2.SLVL_NAME = 'NP-2')));

```

É importante observar que os textos marcados em negrito servem para expressar relações que são tratadas pelos mecanismos de inferência do intelliGOV. No primeiro caso, o trecho é reproduzido a seguir:

(...)

```

SL.SLVL_ID = GT.GET_ID_HIGH AND
GT.GET_ID_LOW = GL.SLVL_ID AND
(GL.SLVL_NAME = 'NP-2' OR
(GL.SLVL_NAME IN (SELECT S1.SLVL_NAME      FROM
SECURITY_LEVEL S1, SECURITY_LEVEL S2, SAME_AS WHERE
SMA_SOURCE_ID = S1.SLVL_ID AND SMA_TARGET_ID = S2.SLVL_ID
AND S2.SLVL_NAME = 'NP-2')))

```

A consulta deve procurar todos os níveis que são maiores ou iguais a NP-2. Observe que mesmo considerando poucos termos, este resultado só responde de maneira adequada através da criação de ligações entre todos os níveis maiores que NP-2

na tabela `GREATER_EQUAL_THAN`, e não apenas com o próximo nível através de transitividade. Outro caso diz respeito à igualdade denotada pela relação *sameAs*. Enquanto no intelliGOV esta relação não precisa ser representada nas regras, ao usar a abordagem SQL, torna-se necessário tornar estes termos claros. Raciocínio semelhante é tratado no segundo trecho destacado, reproduzido a seguir:

```
SL.SLVL_ID = LT.LWT_LOW_ID AND  
LT.LWT_HIGH_ID = LL.SLVL_ID AND  
(LL.SLVL_NAME = 'NP-2' OR  
(LL.SLVL_NAME IN (SELECT S1.SLVL_NAME FROM  
SECURITY_LEVEL S1, SECURITY_LEVEL S2, SAME_AS WHERE  
SMA_SOURCE_ID = S1.SLVL_ID AND SMA_TARGET_ID = S2.SLVL_ID  
AND S2.SLVL_NAME = 'NP-2')));
```

Novamente a consulta só pode ser resolvida com explicitação da relação *sameAs* e com o uso de todas as possíveis relações *lowerThan* entre os níveis, sem aproveitar os benefícios da transacionalidade.

Nas maioria das outras políticas foram encontradas variações semelhantes nas consultas. Para quantificar estas diferenças, foram medidas a quantidade de termos distintos e a quantidade de termos totais das consultas SQL e das políticas implementadas no intelliGOV.

O resultado do cálculo de termos distintos é apresentado na

Tabela 11. Considerando este cenário, é possível observar que em todas as políticas, a quantidade de termos utilizados para sua descrição é reduzido com o uso do intelliGOV. Poderia se questionar se esta é uma diferença originária apenas da linguagem utilizada, porém é possível identificar que esta diferença é mais acentuada nas políticas 1, 2 e 3 (diferenças de 51%, 50% e 77% respectivamente em relação à quantidade de termos da consulta SQL), que são exatamente as políticas que aproveitam recursos como transitividade e relações do tipo *sameAs* para reaproveitar conhecimento expresso na ontologia e evitar sua replicação ao descrever a política.

Tabela 11. Quantidade de termos distintos para SQL e intelliGOV

Política	Quantidade de termos distintos		Diferença entre intelliGOV e SQL
	SQL	intelliGOV	
Política 1	63	31	51%
Política 2	16	8	50%
Política 3	69	16	77%
Política 4	53	33	38%
Política 5	72	40	44%
Política 6	23	15	35%
Política 7	27	19	30%

Considerando o cálculo do total de termos, representado na Tabela 12, podemos observar que a tendência se repete, com exceção da política 4. Tal fator é originário da inclusão de diversos operadores de medição e diferença de conjuntos (*sqwrl:difference*, *sqwrl:size* e *swrlb:equal*) em substituição ao operador *sqwrl:equal*. Esta mesma substituição ocasionou na redução dos ganhos para as políticas 1 e 5.

Tabela 12. Quantidade de termos totais para SQL e intelliGOV

Política	Quantidade total de termos		Diferença entre intelliGOV e SQL
	SQL	intelliGOV	
Política 1	116	96	17%
Política 2	16	12	25%
Política 3	145	37	74%
Política 4	101	114	-13%
Política 5	212	131	38%
Política 6	30	27	10%
Política 7	35	34	3%

Considerando estes fatores, podemos listar algumas vantagens do uso da abordagem intelliGOV em relação ao uso de consultas SQL em bases de dados:

- Menor quantidade de termos a serem utilizados devido ao reuso de axiomas existentes na ontologia, que são considerados pelas consultas semânticas através de inferência;
- Uso de uma linguagem menos técnica e menos dependente de tecnologia para expressar as políticas, podendo inclusive atuar como especificação para implementação otimizada em outras tecnologias;

- Possibilidade de validação dos termos que compõem a consulta e de sua estrutura utilizando a própria ontologia como referência, sem necessidade de implementação em banco de dados.

Um ponto em que esta abordagem se tornou superior em relação ao intelliGOV diz respeito ao desempenho. Enquanto que o tempo de avaliação de políticas com intelliGOV foi de 71,25 segundos para as políticas 4 e 5²² e 3,08 segundos para as outras políticas, o uso da abordagem SQL obteve uma média de 223 ms, sem necessidade de realizar alterações nas políticas 4 e 5 para conseguir executar o processamento. Tal fato se explica pelo uso de tecnologias otimizadas para busca na abordagem de banco, principalmente considerando o uso de uma base de dados em memória contra um processamento baseado em manipulação de XMLs.

6.3 Comparação com o processamento de eventos

Para comparar esta abordagem com intelliGOV, foi gerado um conjunto de Redes de Processamento de Eventos (EPNs) representando as políticas originárias do estudo de caso.

A plataforma utilizada para implementação foi a *Oracle Event Processing (OEP)*²³, cujos componentes implementam os elementos necessários para implementação de uma solução de CEP, propostos por ETZION e NIBLETT (2011): (i) produtores de eventos, que coletam as ocorrências dos eventos em seus ambientes de ocorrência, representados na solução como **Adaptadores**; (ii) canais, utilizados para desacoplar os produtores dos outros elementos da solução, representado como **Canal**; (iii) agentes de processamento, que analisam os eventos e tomam decisões, representados na solução Oracle por **Processadores**; (iv) a representação do contexto atual da organização, fornecendo dados para os agentes poderem contextualizar um evento e tomar decisões mais apuradas, representadas na solução por **Contextos**; e (v), consumidores de eventos, que recebem notificações dos agentes informando suas decisões, representados por **Java Beans** que recebem as notificações e realizam ações, como imprimir o resultado em uma tela ou enviar um e-mail.

²² O resultado das políticas 4 e 5 foram destacados devido ao seu alto desvio em relação aos resultados obtidos com as outras políticas.

²³ <http://www.oracle.com/technetwork/middleware/complex-event-processing/overview/complex-event-processing-088095.html>

Um exemplo de uma EPN desenvolvida na plataforma OEP é apresentada na Figura 20, representando a política 3 do estudo de caso

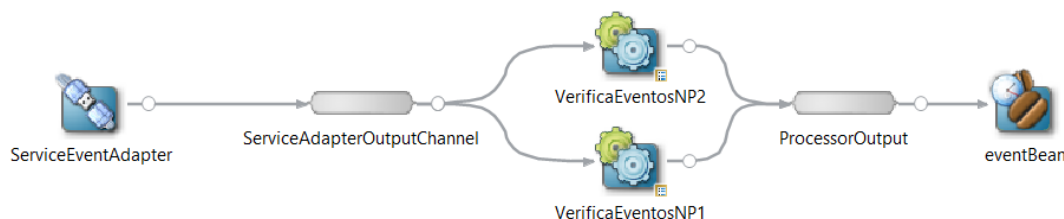


Figura 20. Exemplo de EPN, representando a política 3

Neste caso, o produtor de eventos é representado pelo componente *ServiceEventAdapter*, que aguarda notificações de eventos que podem ser obtidas pela conexão à fila JMS do módulo de carga ou através da leitura de arquivos CSV. Um evento neste caso simboliza cada alteração no ambiente SOA da organização. No protótipo, a cada alteração de um serviço, *release* ou variante, era gerado um evento contendo o nome do serviço, sua interface, entradas, saídas, nível de segurança, área de assunto e protocolo de comunicação utilizado pela interface.

No exemplo apresentado, o evento é inserido em um canal que atua como fila para processamento dos eventos denominado *ServiceAdapterOutputChannel*. Após este canal, foram incluídos dois agentes de avaliação *VerificaEventosNP1* e *VerificaEventosNP2*, denominados na plataforma de processadores. Estes agentes realizam a verificação da política 3 considerando respectivamente os serviços com nível de classificação de segurança menor que NP-2 e com nível de classificação maior ou igual a NP-2.

Cada processador filtra o fluxo de eventos, encaminhando para o canal *ProcessorOutput* apenas os eventos que estão conformes com a política. Estes eventos são listados no componente *eventBean*, que é uma classe Java que imprime no console os eventos julgados conformes.

No caso apresentado, o processador trata dois tipos de elementos:

- A regra de avaliação, descrita através de uma consulta na linguagem Oracle CQL²⁴ (*Continuous Query Language*), baseada na linguagem proposta por ARASU, BABU e WIDOM (2006);
- O contexto que permite obter os dados referentes ao ambiente no qual os eventos ocorrem. Na plataforma OEP, estes dados estão disponíveis em

²⁴ http://docs.oracle.com/cd/E16764_01/doc.1111/e12048/intro.htm

base de dados, sendo este contexto denominado de Contexto JDBC. No caso deste protótipo, foi utilizado o mesmo modelo de dados apresentado na Figura 19.

A regra de verificação é apresentada a seguir.

```
<processor>
  <name>VerificaEventosNP2</name>
  <rules>
    <query id="ConsultaServicos"> <![CDATA[
      select serviceName
      from ServiceAdapterOutputChannel [now],
      TABLE (obtainSecurityMechanism@JDBCCartridgeOne(serviceProtocol) as secMec) as
      securityMechanism,
      TABLE (obtainGreaterThanEqualNP2@JDBCCartridgeOne(serviceLevel) as
      greatEqualThan) as GET
      WHERE securityMechanism.secMec.secMecName = 'Channel Criptography'
    ]]> </query>
  </rules>
</processor>
```

A CQL segue uma sintaxe semelhante ao SQL, podendo especificar janelas de tempo para poder comparar a ocorrência ou alteração de eventos dentro desta janela. Deste modo, a instrução *select serviceName from ServiceAdapterOutputChannel [now]* está buscando os nomes dos serviços a partir do canal *ServiceAdapterOutputChannel* no instante corrente.

É realizada uma operação de *join* destes dados com os dados do contexto utilizando as instruções *TABLE*. No primeiro caso é chamada a função *obtainSecurityMechanism*, para obter os mecanismos de segurança relacionados ao protocolo de comunicação utilizado pela interface do serviço cuja criação ou alteração está sendo analisada. A segunda função, *obtainGreaterThanEqualNP2* verifica se o nível de segurança do serviço é maior ou igual a NP2. Um resultado nulo neste caso anula a consulta como um todo. Por fim, a cláusula *WHERE* verifica se, dentre os mecanismos de comunicação retornados, se encontra a criptografia de canal.

A declaração das funções é realizada no contexto, conforme exibido a seguir.

```

<jdbc:jdbc-ctx>
  <name>JDBCCartridgeOne</name>
  <data-source>BancoServico</data-source>
  <function name="obtainSecurityMechanism">
    <param name="protocolName" type="char"/>
    <return-component-type>
      com.[organizacao].cci.eventdefinitions.SecurityMechanism
    </return-component-type>
    <sql><![CDATA[
      SELECT
        SM.SMEC_ID as secMecId, SM.SMEC_NAME as secMecName
      FROM SECURITY_MECHANISM SM, IMPLEMENTS I,
      COMMUNICATION_PROTOCOL CP
      WHERE
        I.IMP_SMEC_ID = SM.SMEC_ID AND
        I.IMP_CMCT_ID = CP.CMPT_ID AND
        CP.CMPT_NAME = :protocolName
    ]]></sql>
  </function>
  <function name="obtainGreaterThanOrEqualToNP2">
    <param name="securityLevelLow" type="char"/>
    <return-component-type>
      int
    </return-component-type>
    <sql><![CDATA[
      SELECT
        L.SLVL_ID
      FROM
        GREATER_EQUAL_THAN GET, SECURITY_LEVEL L,
        SECURITY_LEVEL H
      WHERE
        L.SLVL_ID = GET.GET_ID_LOW AND
        H.SLVL_ID = GET.GET_ID_HIGH AND
        H.SLVL_NAME = :securityLevelLow AND
        (L.SLVL_NAME = 'NP-2' OR
        L.SLVL_NAME IN ( SELECT S1.SLVL_NAME FROM
        SECURITY_LEVEL S1, SECURITY_LEVEL S2, SAME_AS
        WHERE SMA_SOURCE_ID = S1.SLVL_ID AND
        SMA_TARGET_ID = S2.SLVL_ID AND S2.SLVL_NAME = 'NP-2'))
    ]]></sql>
  </function>
</jdbc:jdbc-ctx>
<processor>

```

No contexto é definida a fonte de dados a ser acessada, delimitada pelas tags *<data-source>* e na sequência são definidas duas funções que permitem obter dados relativos ao contexto SOA da organização. A primeira, *obtainSecurityMechanisms*, realiza a busca dos mecanismos de segurança implementados por um protocolo de comunicação, através de uma consulta SQL na base indicada relacionada à fonte de dados indicada no contexto. A segunda função, *obtainGreaterThanOrEqualToNP2* verifica se um determinado nível passado como parâmetro é maior ou igual ao nível NP-2. Considerando o trecho em negrito no código que descreve o processador, reproduzido a seguir, cabem algumas observações.

(...)

```
L.SLVL_ID = GET.GET_ID_LOW AND  
H.SLVL_ID = GET.GET_ID_HIGH AND  
H.SLVL_NAME = :securityLevelLow AND  
(L.SLVL_NAME = 'NP-2' OR  
L.SLVL_NAME IN ( SELECT S1.SLVL_NAME FROM  
SECURITY_LEVEL S1, SECURITY_LEVEL S2, SAME_AS  
WHERE SMA_SOURCE_ID = S1.SLVL_ID AND  
SMA_TARGET_ID = S2.SLVL_ID AND S2.SLVL_NAME = 'NP-2'))  
(...)
```

- Novamente a ausência de transitividade exigiu a implementação de todas as relações do tipo “maior ou igual que” para possibilitar o retorno correto dos dados;
- Novamente foi necessário explicitar a relação *sameAs* através da tabela *SAME_AS* e verificação de seu conteúdo.

Deste modo, os mesmos problemas de ampliação de quantidade de elementos nas regras e do número de instâncias a serem armazenados em bases de dados também ocorrem.

Adicionalmente, devido a limitações da plataforma, não é possível utilizar nas consultas CQL atributos do evento que sejam representados sob a forma de uma coleção para operações de *join*, iteração ou subtração de conjuntos.

Tomando a política 1 como exemplo, é necessário operar sobre as coleções que representam as entradas e saídas de dados das interfaces de serviço. Para tal, foi necessária a implementação de funções Java que recebem como parâmetro estas listas e executam a verificação de conformidade. Deste modo, a consulta CQL utilizada, representada a seguir, se torna simples:

```
select  eventChannel.getServiceName() as serviceName,  
eventChannel.getServiceRelease() as serviceRelease,  
eventChannel.getServiceVariant() as serviceVariant  
from    eventChannel [now]  
where   validatePolicy(eventChannel.getInterfaceInput(), eventChannel.getInterfaceOutput(),  
eventChannel.getServiceName()) = 1
```

Porém a função *validatePolicy*, cujo código fonte é apresentado no Apêndice VII, teve que contemplar:

- Obtenção da subárea do serviço;

- Obtenção das subáreas hierarquicamente subordinadas a área do serviço;
- Composição destes dois conjuntos, formando um conjunto total de subáreas do serviço;
- Obtenção das subáreas relacionadas às entradas;
- Obtenção das subáreas relacionadas às saídas;
- Composição destes dois conjuntos, formando um conjunto total de subáreas da interface;
- Checagem se o conjunto de subáreas da interface está totalmente contido no conjunto total de subáreas do serviço.

Deste modo, é possível observar que o uso do CEP pode demandar até três linguagens de desenvolvimento para sua implementação. O uso de Java foi necessário em três políticas (1, 4 e 5), enquanto que CQL combinado com SQL foi utilizado nas outras quatro políticas. Esta variedade de tecnologias impediu uma contagem precisa da quantidade de termos, semelhante à realizada para consultas SQL. Porém, a necessidade de expressar os termos que estão implícitos na ontologia se repete, ampliando a complexidade das consultas.

Para avaliar desempenho, a medição de tempo foi obtida através de acompanhamento do tempo de processamento nas ferramentas de monitoração da plataforma CEP. Como o processamento ocorre evento a evento, foi obtido o tempo médio de processamento das políticas por evento, totalizando 152 ms. Se compararmos com a abordagem desta dissertação, na qual temos um tempo da ordem de 100ms considerando as políticas 1, 2, 3, 6 e 7, temos um desempenho semelhante. Porém a abordagem de CEP consegue melhor desempenho na avaliação das políticas 4 e 5, com tempo médio de 497 ms. Novamente, temos uma estrutura otimizada para processamento de altos volumes em memória comparada com o processamento da API Protegé.

Podemos concluir que a abordagem de CEP, além de apresentar as mesmas limitações da abordagem baseada em SQL, também demanda a implementação da solução utilizando múltiplas linguagens de desenvolvimento, tornado a solução mais complexa.

6.4 Comparação com o uso de MDA

Ao analisar o uso de MDA, foi observado que esta abordagem não seria aplicável na organização em que foi executado o estudo de caso. O ponto em questão é a solução ser muito intrusiva.

O modelo proposto por TRAN *et al.* (2012) prevê que tanto as políticas quanto os elementos da arquitetura SOA sejam implementados a partir de modelos, que descrevem seu comportamento e cujo código é gerado automaticamente para executar no ambiente da organização. Este código gerado deve dispor de todos os gatilhos necessários para gerar notificações indicando não conformidades.

Porém, a organização tem como necessidade o consumo de serviços externos, cuja implementação está completamente fora de seu controle. Deste modo, seria inviável realizar a criação dos gatilhos que notificam o estado de conformidade, uma vez que não existe acesso ao código.

Tal ponto se torna mais crítico ao considerarmos que a abstração é um dos princípios de SOA, de acordo com ERL (2005). Este cita que o aspecto de abstração implica no fato de que toda e qualquer lógica de implementação de um serviço deve estar oculta do mundo exterior, sendo permitido o conhecimento apenas de sua interface e contratos. Mesmo se considerarmos abordagens mais modernas, como a proposta por FERRARIO *et al.* (2011), que considera que um serviço deveria ser uma caixa opaca ao invés de caixa preta, na qual pode se compreender parte da forma como um serviço é implementado, mantem-se a ideia de que a implementação não deve ser acessada. Se partirmos para uma abordagem que demanda a interferência de um time de governança para alterar o código de um serviço visando incluir políticas de controle, estamos violando este princípio.

Por conta deste fator, não foi implementado protótipo, porém ao compararmos teoricamente esta abordagem com intelliGOV, observamos que o intelliGov corresponde a uma solução não intrusiva, podendo obter os dados de ferramentas de apoio a SOA ou qualquer outra ferramenta utilizada para monitorar e controlar serviços e ativos em uma organização, demandando a construção de extratores e extensão da ontologia, porém sem demandar intrusões na implementação de cada serviço. Também é importante ressaltar que este mesmo posicionamento é válido para as abordagens baseadas em bancos de dados e eventos.

6.5 Comparação com Data Warehouse

Tanto BIRUKOU *et al.* (2010) quanto RODRÍGUEZ *et al.* (2013) propõem a inclusão de mecanismos de *data warehouse* na arquitetura de suas soluções. Ao analisar a função deste elemento na arquitetura, é possível perceber que o uso desta tecnologia não está focado no diagnóstico de conformidade, mas sim na consolidação dos resultados da análise para geração de indicadores que representem a evolução da conformidade na organização. Como o foco deste trabalho é a detecção de conformidade, esta abordagem pode ser vista como complemento a este trabalho e não como outra forma de lidar com a avaliação de conformidade.

6.6 Comparação com outras abordagens de web semântica

Ao compararmos intelliGOV com a abordagem proposta por ZHOU *et al.* (2010), podem ser observadas algumas similaridades. Ambas se baseiam em um modelo interpretável por agentes computacionais para descrever o contexto (ontologias no intelliGOV e SML na abordagem analisada) e ambas propõem descrever regras usando elementos que manipulem estes modelos (SWRL/SQWRL para intelliGOV e ISO Schematron para a abordagem analisada).

A diferença entre as abordagens se resume na simplicidade da arquitetura proposta nesta dissertação. A abordagem intelliGOV utiliza bibliotecas e ferramentas abertas, sem a necessidade de implementar mecanismos de inferência ou de processamento adicionais de políticas. Já a abordagem de ZHOU *et al.* (2010) depende da implementação de módulos adicionais para tratar as políticas, através de combinação do Schematron com a SML e módulos adicionais para realizar a inferência. Para tal, é demandado desenvolvimento de código especializado para a solução.

Já a abordagem de SPIES (2012) se aproxima bastante do intelliGOV, porém, além de não propor mecanismos de carga e interação com o usuário, a abordagem se limita ao uso de SWRL no contexto de governança de TI. Conforme observado no estudo de caso, SWRL não consegue representar todas as regras necessárias para as políticas tratadas, sendo necessária a inclusão de operadores de agrupamento e comparação de conjuntos. Esta situação só não ocorreu nas política 2 e 3. Todas as outras políticas demandaram o uso de operadores da SQWRL.

6.7 Consolidação da comparação

Com os resultados desta análise, podemos observar que a abordagem proposta reduz a quantidade de termos necessários para expressar uma política em relação a outras abordagens através do uso de inferência, conforme visto para os casos de bancos de dados e processamento de eventos. Também existe a vantagem do uso de uma tecnologia única para especificar e executar a verificação de conformidade.

Também podemos considerar como um ganho o reuso de ontologias existentes para modelar o domínio, de maneira semelhante ao estudo de caso no qual foi reutilizada a ontologia do Open Group para SOA, permite reduzir o tempo de modelagem. Da mesma maneira, é possível ampliar os domínios validados, incorporando novas ontologias.

Outro ponto relevante é a expressividade do modelo descrito sob a forma de uma ontologia, que permite a captura e formalização de diversos conceitos que se perdem ao se otimizar uma implementação de banco de dados, como o exemplo da possível conversão das relações explícitas de ordem e igualdade em identificadores numéricos ordenados.

Adicionalmente, possui uma expressividade maior que abordagens semelhantes que não utilizam SQWRL, sem exigir a implementação de módulos adicionais para processar as políticas.

Porém, um ponto a ser melhorado na solução é o seu desempenho. Apesar de executar em tempo que permita a realização de verificações on-line, o uso de bases de dados em memória permite um tempo de resposta otimizado. Porém cabe ressaltar que o primeiro protótipo do intelliGOV utilizou uma ontologia implementada em arquivos. Uma possível evolução é refinar esta arquitetura, persistindo as informações em um banco de dados.

Mesmo assim, caso se opte pela utilização de uma base em memória, seria possível utilizar intelliGOV para especificar o domínio e as políticas, ampliando a precisão dos requisitos de implementação do banco de dados e suas restrições. Esta mesma camada pode atuar como uma camada de abstração para a execução de consultas em um banco relacional. Um exemplo deste tipo de cenário é proposto em MOTIK, HORROCKS e SATTler (2009).

7. Conclusões

Neste capítulo são apresentadas as conclusões finais, as contribuições deste trabalho e propostas de trabalhos futuros sobre o tema.

7.1 Conclusões sobre o trabalho

Este trabalho apresentou intelliGOV, uma abordagem baseada em ontologias, regras e consultas semânticas para apoiar a verificação de conformidade no contexto de arquiteturas orientadas a serviço (SOA). Neste contexto, a heterogeneidade originária da integração entre aplicações gera um conjunto elevado de domínios a serem considerados ao se analisar a conformidade, exigindo a participação de especialistas para garantir um nível mínimo de qualidade nestas avaliações.

A questão de conformidade é chave para garantir a evolução de SOA nas organizações. Sem conformidade, perde-se a possibilidade de exercer uma efetiva governança que impeça que os múltiplos *stakeholders* envolvidos em ambientes distribuídos desenvolvam e operem serviços partindo de premissas, padrões e regulamentações distintas. Adicionalmente, sem conformidade se torna complexo atender a regulamentações, leis e normas das organizações.

A solução proposta captura os conceitos dos domínios envolvidos em ontologias e expressa as políticas da organização sob a forma de regras e consultas semânticas, baseadas no vocabulário definido na ontologia.

Para validar esta solução, foi executado um estudo de caso em organização de porte global que tem investido fortemente em SOA. Foram selecionadas sete políticas para realizar uma avaliação de 25 serviços, executada tanto por analistas com múltiplos níveis de experiência, quanto pela abordagem intelliGOV.

Foi observado que a abordagem intelliGOV conseguiu gerar resultados com qualidade maior que a obtida pela avaliação manual, através da redução dos erros de diagnóstico de um valor médio de 21% para zero, corroborando a hipótese de que o uso

de ontologias, regras e consultas semânticas reduz a quantidade de erros da atividade de verificação de conformidade. Tal ponto se torna importante no momento em que uma eventual não conformidade pode gerar multas consideráveis devido a falha no atendimento a legislação.

Também foi observado que um nível baixo de erros de diagnóstico (em torno de 10%) são obtidos apenas por analistas com ampla experiência, portanto o valor da solução está em deslocar estes analistas da atividade de verificação manual para apoiarem a definição de políticas e resolução de problemas.

Adicionalmente, foram identificados indícios de reduções de custos no processo. Através da medição do esforço gasto na avaliação, do esforço gasto na avaliação utilizando a abordagem e em uma estimativa do custo necessário para customização do intelliGOV para uma organização, foi possível verificar que uma implementação teria seus custos compensados após sete verificações. Também foi possível identificar que alterações nas políticas são fatores de baixo peso no custo da solução, tornando simples a absorção de mudanças no ambiente corporativo.

Por fim, a solução foi comparada com outras abordagens citadas na literatura. Observou-se que intelliGOV consegue representar as regras com maior simplicidade em relação às outras abordagens, utilizando-se da capacidade de inferência provida pelas tecnologias utilizadas. A arquitetura é menos intrusiva, depende de menos desenvolvimento de novos componentes para sua implementação e demandando o uso de apenas uma linguagem de representação. O ponto a ser melhorado é o desempenho, que mesmo apresentando tempos de resposta aceitáveis, se mostrou aquém do desempenho obtido por soluções de bases de dados em memória.

7.2 Contribuições

Como contribuições diretas deste trabalho, podemos destacar:

- Um conjunto de ferramentas e tecnologias que facilita a atividade de verificação de conformidade em arquiteturas orientadas a serviço, baseadas em ontologias, regras e consultas semânticas;
- A identificação da necessidade do uso de consultas semânticas para representar regras em organizações, ao invés do uso apenas de regras e ontologias. Conforme os resultados obtidos no estudo de caso, o uso isolado de regras não permite realizar comparações de conjuntos e

agregações dos dados. Estas operações se tornaram necessárias para representar a maioria das políticas avaliadas no estudo de caso;

- Uma comparação que explicita os pontos fortes e fracos das abordagens atualmente disponíveis na literatura para tratar o tema de conformidade em SOA.

Adicionalmente, algumas contribuições foram obtidas indiretamente, como resultados obtidos ao longo da pesquisa:

- Identificação e concepção de um modelo comum para processos de governança SOA (TEIXEIRA FILHO e AZEVEDO, 2012; TEIXEIRA FILHO e AZEVEDO, 2014);
- Estudo sobre ontologias aplicáveis ao contexto de governança SOA (TEIXEIRA FILHO e AZEVEDO, 2013)
- Indícios de que o tipo de experiência profissional que é mais relevante para avaliações de conformidade é a experiência na organização, pesando até mais que a experiência técnica;
- Entendimento de mecanismos de extração e carga de ferramentas como barramentos e repositórios de serviços.

7.3 Trabalhos futuros

Com base nos resultados obtidos, diversas possibilidades de novos trabalhos podem ser sugeridas para aprofundamento e evolução desta pesquisa.

Em primeiro lugar, a abordagem *intelliGOV* permite cobrir apenas uma etapa de um ciclo de governança. Diversos autores (HOJAJI e SHIRAZI, 2010; NIEMANN *et al.*, 2010; SCHEPERS, IACOB e VAN ECK, 2008) sugerem que a governança SOA pode ser vista como um ciclo PDCA – *Plan, Do, Check and Act*, baseado na garantia de conformidade. O trabalho realizado nesta dissertação corresponde à etapa *Check* deste ciclo. Ao longo de sua evolução, muito foi discutido sobre como delimitar quais são as políticas efetivamente relevantes à organização (*Plan*), métodos para transformar estas políticas em regras (*Do*) e agir para corrigir desvios (*Act*), como apresentado a seguir.

Considerando a etapa *Plan*, trabalhos nas áreas de arquitetura empresarial, modelagem de processos e levantamento de requisitos poderiam estabelecer técnicas para identificar as políticas corretas a serem empregadas no contexto SOA.

Na etapa *Do*, estudos nas áreas de processamento de linguagem natural e semântica poderiam facilitar a transformação dos textos descritivos das políticas em regras interpretáveis pelo intelliGOV. Uma possibilidade seria o uso de abordagens em maior nível de abstração para fazer um primeiro nível de tradução, que poderia posteriormente ser processado para gerar as consultas em SQWRL. Exemplos seriam a abordagem proposta por (LOPES, 2011), que propõe o uso de ontologias de fundamentação para descrever regras de negócio, reduzindo possíveis erros de interpretação, e a *Semantics of Business Vocabulary and Business Rules (SBVR)* OMG (2004), que descreve assertivas em um formato próximo a linguagem natural, porém tornando a estrutura menos suscetível a erros de interpretação. Outra possibilidade neste contexto seria propor um método para formulação das políticas. Ao analisar as políticas avaliadas no estudo de caso é possível perceber indícios de um padrão para estruturar seu conteúdo, envolvendo etapas para: declaração dos elementos envolvidos, definição do relacionamento entre estes elementos e definição de restrições sobre os mesmos. Uma possibilidade seria a análise e formalização deste tipo de padrão para compor um procedimento que simplificasse a elaboração das políticas e permitisse sua validação.

Na etapa *Check*, ainda existe espaço para tratar a questão de indicadores, endereçada na literatura por soluções de data warehouse. Estudos na área de regras e ontologias poderiam definir técnicas que simplificariam a criação destes indicadores no contexto do intelliGOV. Outro ponto interessante seria o estudo da aplicação do intelliGOV em verificações do tipo *forward*, permitindo a captura de problemas antes de iniciar a operação dos serviços. Neste contexto, uma possibilidade seria incluir as políticas em mecanismos de testes automatizados e de regressão, permitindo capturar falhas em caso de alterações não conformes em serviços existentes.

Por fim em *Act*, trabalhos na área de agentes inteligentes poderiam utilizar o conhecimento existente na solução em conjunto com a informação de não conformidade e tomar ações para regular, de maneira automatizada, o ambiente SOA das organizações.

Ainda nesta linha, todos estes elementos poderiam ser alinhados em uma metodologia para prover uma meta-governança SOA, permitindo criar um processo que orientasse a implementação de governança SOA nas organizações.

Outra linha de trabalho compreenderia verificar se é possível ampliar a abrangência da solução para outros contextos. Esta dissertação teve foco no domínio de governança SOA, tratando apenas conceitos e políticas neste contexto. Porém, seria

interessante verificar a possibilidade de integrar ontologias que tratam outros domínios e verificar se abordagem consegue reproduzir os mesmos ganhos, ampliando assim a sua abrangência. Adicionalmente, poderiam ser tratados conceitos relativos ao negócio integrados ao contexto de SOA, permitindo avaliar como validar políticas de negócio em um ambiente orientado a serviços.

Do ponto de vista técnico, uma das questões identificadas diz respeito ao desempenho da solução. Seria interessante verificar mecanismos que permitissem melhorias de tempo de processamento de consultas e regras semânticas, como o uso de bases de dados para persistir os dados ou o uso do intelliGOV como uma camada de abstração para acesso aos modelos e execução de consultas em bancos de dados.

Outro ponto importante é a melhoria do funcionamento da linguagem de consulta. Apesar de descrita e exemplificada na literatura pelos autores da SQWRL (O'CONNOR e DAS, 2009), a operação *sqwrl:equal* não se comportou de maneira esperada, aumentando desnecessariamente a complexidade das políticas formuladas. Seria interessante contar com o pleno funcionamento deste operador.

Outro fator interessante seria tentar utilizar as descrições contidas nas ontologias para automatizar a geração dos programas de extração. Dispondo dos modelos das ferramentas das quais estão previstas as extrações e utilizando técnicas para alinhar estes modelos com a ontologia, seria possível identificar de maneira automatizada os mapeamentos e gerar os programas de carga.

Por fim, a mesma semântica poderia ser utilizada nas ferramentas de interação com o usuário. Apesar de ter sido desenvolvido um protótipo que simplifica o acesso a ontologia e a definição de políticas, este apresenta um modelo estático da ontologia para o usuário, que clica sobre este modelo para selecionar conceitos e propriedades. A interpretação da ontologia poderia permitir a geração de visualizações dinâmicas deste modelo, ampliando a flexibilidade da solução.

Referências

ANTONIOU, G.; FRANCONI, E.; VAN HARMELEN, F. Introduction to semantic web ontology languages. In: *Reasoning Web*. [s.l.] Springer, 2005. p. 1–21.

ARASU, A.; BABU, S.; WIDOM, J. The CQL Continuous Query Language: Semantic Foundations and Query Execution. *The VLDB Journal*, v. 15, n. 2, p. 121–142, jun. 2006.

BAJEC, M.; KRISPER, M. A methodology and tool support for managing business rules in organisations. *Information Systems*, v. 30, n. 6, p. 423–443, set. 2005.

BEN HAMIDA, A. et al. An integrated development and runtime environment for the future internet. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 7281 LNCS, p. 81–92, 2012.

BENNETT, S. G. *Oracle Practitioner Guide - A Framework for SOA Governance*. Disponível em: <<http://www.oracle.com/technetwork/topics/entarch/oracle-pg-soa-governance-fmwrk-r3-2-1561703.pdf>>. Acesso em: 3 jul. 2012.

BIRUKOU, A. et al. An integrated solution for runtime compliance governance in SOA. *Service-Oriented Computing*, p. 122–136, 2010.

BRASIL. Lei Nº 12.527 De 18 de Novembro de 2011. Regula o acesso a informações previsto no inciso XXXIII do art. 5º, no inciso II do § 3º do art. 37 e no § 2º do art. 216 da Constituição Federal; altera a Lei nº 8.112, de 11 de dezembro de 1990; revoga a Lei nº 11.111, de 5 de maio de 2005, e dispositivos da Lei nº 8.159, de 8 de janeiro de 1991; e dá outras providências. *Diário Oficial*, Brasília, ed. Extra, pg.1, 18 de Novembro de 2011

BRAZIER, F. et al. Agent-based organisational governance of services. *Multiagent and Grid Systems*, v. 8, n. 1, p. 3–18, 2012.

BROWN, W. A.; MOORE, G.; TEGAN, W. *SOA Governance-IBM's Approach*. Disponível em: <ftp://170.225.15.40/software/soa/pdf/SOA_Gov_Process_Overview.pdf>.

IT Governance Institute, *Cobit 4.1*. EUA: IT Governance Institute, 2007.

CYGANIAK, R.; WOOD, D.; LANTHALER, M. *RDF 1.1 Concepts and Abstract Syntax*. Disponível em: <<http://www.w3.org/TR/rdf11-concepts/>>. Acesso em: 10 abr. 2014.

DIIRR, T.; AZEVEDO, L.; SANTORO, F. SOA Governance from an Enterprise Architecture Viewpoint. In: *Anais do X SIMPOSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO - SBSI*. Londrina: 2014

ERL, T. *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall, 2005.

ERL, T. *SOA governance governing shared services on-premise and in the cloud*. Upper Saddle River, NJ: Prentice Hall, 2011.

ETZION, O.; NIBLETT, P. *Event processing in action*. Greenwich: Manning, 2011.

EUZENAT, J. *Ontology matching*. New York: Springer, 2007.

FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ, A.; JURISTO, N. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In: *Proceedings of the Ontological Engineering AAAI-97*, EUA, 1997

FERRARIO, R. et al. Towards an Ontological Foundation of Services Science: The General Service Model. In: *Proceedings of the 10th International Conference on Wirtschaftsinformatik*, 16th-18th February 2011, Zurich, Switzerland, 2011

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. Dissertation. University of California, Irvine, CA, EUA. 2000.

FORTIŞ, T.-F.; MUNTEANU, V. I.; NEGRU, V. "Towards an ontology for cloud services". In: *Proceedings - 2012 6th International Conference on Complex, Intelligent, and Software Intensive Systems*, 2012.

GARMUS, D.; HERRON, D. *Function Point Analysis: Measurement Practices for Successful Software Projects*. 1 ed. Boston: Addison-Wesley Professional, 2000.

GORTON, S. et al. "StPowla: SOA, policies and workflows". In: *Service-Oriented Computing-ICSOA 2007 Workshops*. Springer Berlin Heidelberg, 2009. p. 351-362.

GRUBER, T. R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *Knowledge acquisition*, v. 5, n. 2, p. 199–220, 1993.

GUARINO, N. "Formal Ontologies and Information Systems". In: *Proceedings of FOIS 98*, Trento, Italia, 2008.

HANNA, A. et al. *ITIL V3 foundation handbook*. Gloucester, UK: The Stationery Office, 2009.

HEUSER, C. A. *Projeto de Banco de Dados*. 6. ed. [s.l.] Bookman, 2009. v. 4

HEWITT, E. *Java SOA Cookbook*. [s.l.] O'Reilly Media, Inc., 2009.

HITZLER, P. et al. *OWL 2 Web Ontology Language Primer (Second Edition)*. Disponível em: <http://www.w3.org/TR/2012/REC-owl2-primer-20121211/#What_is_OWL_2.3F>. Acesso em: 9 fev. 2014.

HOJAJI, F.; SHIRAZI, M. R. . A Comprehensive SOA Governance Framework Based on COBIT. In: *Proceedings of 6th World Congress on Services (SERVICES-1)*. Miami, FL: 2010

HORROCKS, I. et al. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, v. 21, p. 79, 2004.

HSIUNG, A.; RIVELLI, G.; HUTTENEGGER, G. How to design a global SOA infrastructure: Coping with challenges in a global context. In: *Proceedings of the 2012 IEEE 19th International Conference on Web Services, ICWS*. 2012

JANIESCH, C.; KORTHAUS, A.; ROSEMAN, M. Conceptualisation and facilitation of SOA governance. In: *Proceedings of: 20th Australasian Conference on Information Systems*. Melbourne: 2009

JANIESCH, C.; NIEMANN, M.; REPP, N. Towards a service governance framework for the internet of services. In: *Proceedings of the 17th European Conference on Information Systems*. Verona: 2009

JELLIFFE, R. *The Schematron Assertion Language 1.6*. Disponível em: <<http://xml.ascc.net/resource/schematron/Schematron2000.html>>. Acesso em: 7 dez. 2013.

JOACHIM, N.; BEIMBORN, D.; WEITZEL, T. The influence of SOA governance mechanisms on IT flexibility and service reuse. *Journal of Strategic Information Systems*, 2013.

JOSUTTIS, N. *Soa in practice*. First ed. [s.l.] O'Reilly, 2007.

KARAN, D. V. *Ferramenta para Verificação de Conformidades utilizando Ontologias e Regras Semânticas em Governança SOA*. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação). Departamento de Informática Aplicada, UNIRIO, Rio de Janeiro, 2013.

LESBEGUERIES, J. et al. Multilevel event-based monitoring framework for the Petals Enterprise Service Bus (industry article). In: *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, DEBS'12*. 2012

LOPES, M. M. G. *MODELAGEM CONCEITUAL DE REGRAS DE NEGÓCIO BASEADA EM ONTOLOGIA DE FUNDAMENTAÇÃO*. Dissertação de Mestrado. Departamento de Informática Aplicada. UNIRIO, Rio de Janeiro, set. 2011.

LUCKHAM, D. C. *The power of events*. [s.l.] Addison-Wesley Reading, 2002.

MARKS, E. A.; BELL, M. *Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology*. 1. ed. [s.l.] Wiley, 2006.

- MELLOR, S. J. et al. Model-driven architecture. *Advances in Object-Oriented Information Systems*, p. 290–297, 2002.
- MOTIK, B.; HORROCKS, I.; SATTLER, U. Bridging the gap between OWL and relational databases. *Web Semantics: Science, Services and Agents on the World Wide Web*, v. 7, n. 2, p. 74–89, abr. 2009.
- NIEMANN, M. et al. Challenges of Governance Approaches for Service-Oriented Architectures. In: *Proceedings of the Thrid IEE International Conference on Digital Ecosystems and Technologies*. Istanbul: 2009
- NIEMANN, M. et al. Structuring SOA Governance. *International Journal of IT/Business Alignment and Governance*, v. 1, n. 1, p. 58–75, 2010.
- NOY, N. F.; MCGUINNESS, D. L. Ontology development 101: A guide to creating your first ontology. [s.l.] *Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880*, 2001.
- O’CONNOR, M. J.; DAS, A. K. SQWRL: A Query Language for OWL. In: *Proceedings of OWLED*. 2009
- OMG. *Semantics of Business Vocabulary and Business Rules (SBVR)*. Disponível em: <<http://www.omg.org/spec/SBVR/>>. Acesso em: 05 Maio 2014
- PANDIT, B.; POPESCU, V.; SMITH, V. *Service Modeling Language, Version 1.1*. Disponível em: <<http://www.w3.org/TR/sml/>>. Acesso em: 6 dez. 2013.
- PAPAZOGLU, M. P. Service -Oriented Computing: Concepts, Characteristics and Directions. In: *Proceedings of the Fourth International Conference on Web Information Systems Engineering*. Washington, DC, USA, 2003
- PAPAZOGLU, M. P. et al. Service-oriented computing: State of the art and research challenges. *Computer*, v. 40, n. 11, p. 38–45, 2007.
- PENG, K.-Y.; LUI, S.-C.; CHEN, M.-T. A study of design and implementation on SOA Governance: A Service Oriented Monitoring and alarming perspective. In: *Proceedings of the 4th IEEE International Symposium on Service-Oriented System Engineering*, 2008.
- RAMEZANI, E.; FAHLAND, D.; VAN DER AALST, W. M. Where did i misbehave? diagnostic information in compliance checking. In: *Business Process Management*. [s.l.] Springer, 2012. p. 262–278.
- RODRÍGUEZ, C. et al. SOA-enabled compliance management: instrumenting, assessing, and analyzing service-based business processes. *Service Oriented Computing and Applications*, p. 1–18, 2013.
- RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering - Springer. *Empirical Software Engineering*, 2008.

SCHEPERS, T. G. J.; IACOB, M. E.; VAN ECK, P. A. T. A lifecycle approach to SOA governance. In: *Proceedings of the 2008 ACM symposium on Applied computing*. Fortaleza, CE: 2008

SIMPERL, E. P. B.; TEMPICH, C.; SURE, Y. ONTOCOM: A Cost Estimation Model for Ontology Engineering. In: CRUZ, I. et al. (Eds.). *The Semantic Web - ISWC 2006*. Lecture Notes in Computer Science. [s.l.] Springer Berlin Heidelberg, 2006. p. 625–639.

SIRIN, E.; PARSIA, B. SPARQL-DL: SPARQL Query for OWL-DL. In: *Proceedings of OWLED*. 2007.

SOUZA, J. et al. 0002/2008 - Gestão de Ontologias. *RelaTe-DIA*, v. 2, n. 1, 9 fev. 2009.

SPIES, M. Continuous Monitoring for IT Governance with Domain Ontologies. In: *23rd International Workshop on Database and Expert Systems Applications*. 2012

STEIN, S.; LAUER, J.; IVANOV, K. ARIS method extension for business-driven SOA. *Wirtschaftsinformatik*, v. 50, n. 6, p. 436–444, 2008.

SU, X.; ILEBREKKE, L. Using a Semiotic Framework for a Comparative Study of Ontology Languages and Tools. *Information modeling methods and methodologies*, n. 1537-9299, p. 278–299, 2005.

SURE, Y.; STAAB, S.; STUDER, R. On-To-Knowledge Methodology (OTKM). In: STAAB, P. D. S.; STUDER, P. D. R. (Eds.). *Handbook on Ontologies*. International Handbooks on Information Systems. [s.l.] Springer Berlin Heidelberg, 2004. p. 117–132.

TEIXEIRA FILHO, H. M.; AZEVEDO, L. G. CommonGOV: A Consolidate Approach for Governance of Service-Oriented Architectures. In: *8TH INTERNATIONAL CONFERENCE ON WEB SERVICES PRACTICES*. São Carlos, Brasil: 2012

TEIXEIRA FILHO, H. M.; AZEVEDO, L. G. Uso de Ontologias para Governança de Arquiteturas Orientadas a Serviço. *RelaTe-DIA*, v. 7, n. 1, 15 nov. 2013.

TEIXEIRA FILHO, H. M.; AZEVEDO, L. G. Governance of Service-Oriented Architecture through the CommonGov Approach. *International Journal of Computer Information Systems and Industrial Management Applications*, v. 6, p. 505–514, 2014.

THE OPEN GROUP. *SOA Governance Framework*. Disponível em: <<https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=c093>>. Acesso em: 14 abr. 2012.

THE OPEN GROUP, *Service-Oriented Architecture Ontology*. Disponível em: <<https://www2.opengroup.org/ogsys/protected/publications/viewDocument.html?publicationid=12245&documentid=11637>>. Acesso em: 15 nov. 2012.

THOMAS MANES, A. *A Guidance Framework for SOA Governance: Burton Research*. [s.l.] Gartner Group, 2009

TRAN, H. et al. An end-to-end framework for business compliance in process-driven SOAs In: *Proceedings of the 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2010

TRAN, H. et al. Compliance in service-oriented architectures: A model-driven and view-based approach. *Information and Software Technology*, v. 54, n. 6, p. 531–552, 2012.

VIERING, G.; LEGNER, C.; AHLEMANN, F. The (Lacking) Business Perspective on SOA-Critical Themes in SOA Research. In: *Proceedings of the 9TH INTERNATIONAL CONFERENCE ON BUSINESS INFORMATICS*. Wien: 2009

YANG, Y. et al. Phase distribution of software development effort. In: *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. 2008

YIN, R. K. *Case study research: Design and methods*. [s.l.] Sage, 2009. v. 5

ZHOU, Y. C. et al. Context model based SOA policy framework. In: *Proceedings of the IEEE 8th International Conference on Web Services*. 2010

Apêndice I. **commonGOV**

Para desenvolver o trabalho descrito nesta dissertação, foi necessário realizar um estudo sobre modelos de governança propostos pela literatura para identificar o conjunto de processos necessário para implementar uma efetiva governança SOA e quais seriam os processos a serem verificados em uma estratégia de verificação de conformidade. Neste estudo foram identificadas divergências e similaridades entre os modelos e proposto um modelo comum, resultante da composição das propostas analisadas. A partir deste trabalho foi possível identificar a diversidade de domínios de conhecimento necessários para implementação de governança, tornando complexa a atividade de verificação de conformidade. Neste apêndice, este modelo é descrito.

I.1 Modelos de governança

O primeiro passo deste estudo foi o levantamento de propostas de modelos de governança originárias da academia e do mercado. Inicialmente, o foco deste estudo era identificar eventuais problemas a serem tratados ao longo da pesquisa, com foco em governança SOA. Para tal foi realizado um levantamento de abordagens propostas pela academia e pelo mercado. Os trabalhos identificados são listados a seguir.

NIEMANN et al. (2010) propõem um modelo com os componentes necessários para implementar um mecanismo de governança SOA nas organizações, apresentado na seção 2.1.4 desta dissertação. Trata-se de um modelo que considera os elementos em alto nível de abstração – como visto, políticas, estruturas organizacionais, boas práticas, observação de conformidade e medição de maturidade – e não se aprofunda nos processos necessários para implementar uma estrutura de governança SOA.

JANIESCH, NIEMANN e REPP (2009) propõem um modelo baseado nos frameworks COBIT (IT Governance Institute, 2007) e ITIL (HANNA *et al.*, 2009), dividindo os processos em cinco possíveis estágios: projeto, implantação, entrega, monitoração e mudança, e verificando se estes processos existem, precisam de evolução

ou não existem nos frameworks COBiT e ITIL. Porém este trabalho não descreve o objetivo da cada processo, cabendo ao leitor intuir o seu significado.

O OPEN GROUP (2009) propõe um modelo que divide os processos em duas grandes categorias: processos de governança, que tem como objetivo manter a governança operacional; e processos governados, que são utilizados para implementar e operar SOA e devem ser regidos pelos processos de governança. Adicionalmente, propõe quatro categorias de processos governados: processos para gestão de portfólio de serviços, com o objetivo de definir e manter qual é o conjunto de serviços da organização; gestão de ciclo de vida de serviços, com o objetivo de controlar o desenvolvimento e operação de serviços; gestão do portfólio de aplicações SOA, com foco no controle do conjunto de aplicações SOA da organização, que são as aplicações construídas através de composição ou uso de serviços; e gestão do ciclo de vida de aplicações SOA, determinando com estas devem ser desenvolvidas e operadas. Apesar de conter uma descrição detalhada de cada processo, o modelo não considera aspectos relacionados a gestão de mudança e não considera aspectos relacionados a segurança da informação.

HOJAJI e SHIRAZI (2010) propõem um modelo baseado no COBiT, contendo atividades que consideram quatro domínios: planejar, definir, implementar e medir. Porém este modelo considera apenas a governança de serviços, sem considerar a governança das aplicações que utilizam os serviços.

DIIRR, AZEVEDO e SANTORO (2014) também apresentam um conjunto de processos para gerenciamento de SOA que suportam a governança de serviços, tratando de diversas perspectivas como impacto na organização e definição de novos papéis, prospecção tecnológica, estratégia de implantação, projeto e implementação de infraestrutura e comunicação de resultados na organização. Porém este modelo não trata a governança das aplicações baseadas em SOA, segurança e gestão de mudança.

A proposta da Oracle (BENNETT, 2012) se baseia na proposta do Open Group para propor um modelo de governança, incluindo aspectos para tratar a mudança cultural originária da implantação de SOA. Porém não considera a gestão da infraestrutura relacionada aos serviços e não inclui processos para identificação dos serviços.

Por fim, o Gartner Group (THOMAS MANES, 2009) propõe um conjunto de itens a serem endereçados por governança SOA, como gestão de portfólio de serviços, definição de arquitetura técnica e gestão de mudanças. Este modelo é apresentado de

maneira detalhada por ERL (2011) e apenas cita a governança das aplicações consumidoras, sem considerar atividades específicas para este fim e não considera atividades relacionadas à segurança e à identificação e priorização de serviços.

Diante destas divergências, foi identificada a necessidade de compor um modelo comum que represente de maneira consolidada o universo de processos para governança SOA. Este modelo, denominado commonGOV, é apresentado na seção a seguir.

I.2 commonGOV

O resultado da avaliação dos trabalhos listados na seção anterior levou a proposta do commonGOV, um modelo comum representando o conjunto de processos necessário para implementar governança SOA. Para tal, estes processos foram organizados em grupos e subgrupos de acordo com suas similaridades. A Figura 21 apresenta os processos que compõe o commonGOV, devidamente organizados em grupos e subgrupos.

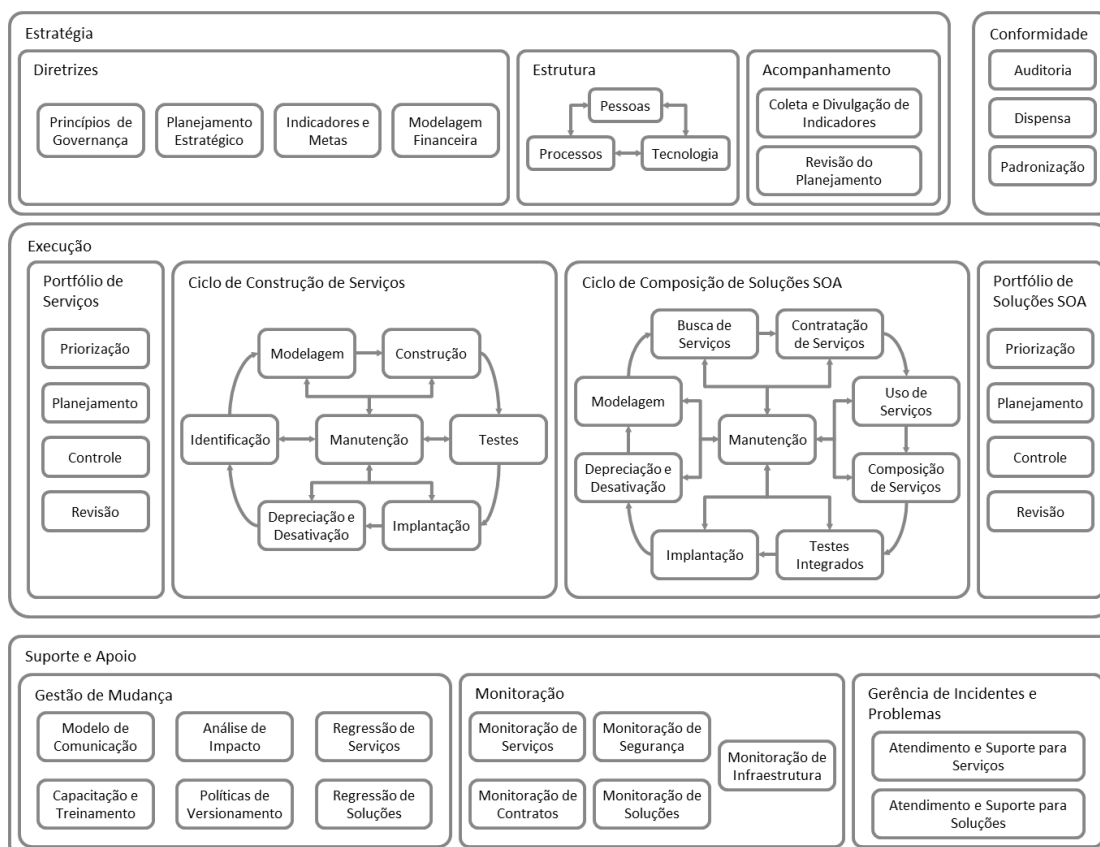


Figura 21. Estrutura do commonGOV

Em termos de grupos, o modelo foi dividido em quatro categorias: Estratégia, que tem foco na definição dos objetivos e premissas que direcionam o alinhamento entre a arquitetura e o negócio; Conformidade, que trata mecanismos para verificar e

garantir que a arquitetura desenvolvida está alinhada com os objetivos e premissas definidos pelo grupo de estratégia; Execução, que estabelece regras para a constituição de SOA; e Suporte e Apoio, que definem os processos necessários para suportar e manter SOA operacional na organização.

O grupo estratégia é definido dividido em três subgrupos: diretrizes, que focam na definição e comunicação dos objetivos da arquitetura; estrutura, que tem o foco de alinhar os elementos que compõe a organização (pessoas, processos e tecnologia) com os objetivos definidos pelas diretrizes; e acompanhamento, que mede e divulga os resultados de SOA.

O grupo de conformidade não possui subgrupos, mas trata três processos que visam assegurar o alinhamento entre SOA e os objetivos: auditoria, para verificar o alinhamento; dispensa, para tratar situações que estão não conformes mas que a organização aceita devido as suas condições no momento de ocorrência de uma não conformidade; e padronização, que tem foco em definir regras para estabelecer padrões e políticas de maneira uniforme na organização.

O grupo de execução contempla quatro subgrupos, baseados no modelo proposto pelo Open Group: portfólio de serviços, ciclo de construção de serviços, ciclo de construção de soluções SOA e portfólio de aplicações.

Por fim, o grupo de suporte e apoio é dividido em quatro subgrupos: gestão de mudança, com foco nas atividades necessárias para comunicar, treinar e tratar a mudança cultural originária em SOA e as alterações nos elementos da arquitetura; monitoração, com o objetivo de diagnosticar problemas; e gerência de incidentes e problemas, com foco no tratamento de problemas que sejam identificados por usuários e pela monitoração.

Uma descrição detalhada dos processos que compõe o commonGOV pode ser vista em TEIXEIRA FILHO e AZEVEDO (2012, 2014).

I.3 Comparação commonGOV com as outras propostas

Para comparar o commonGOV com outras abordagens, foi composta uma tabela na qual as linhas correspondiam a cada processo do modelo. Nas colunas foram destacadas as abordagens analisadas. Para cada cruzamento (processo x abordagem), foi verificado se o processo é descrito, apenas citado ou não consta da abordagem, através

das respectivas indicações T, P ou N. Um exemplo é apresentado na Tabela 13, demonstrando a análise do grupo de Estratégia.

Tabela 13. Exemplo de comparação do commonGOV

Processo	Open Group	Gartner	Oracle	Janiesch, Niemann Repp	Hojaji Shirazi	Dirr Azevedo Santoro
Princípios de Governança	T	P	T	T	T	T

Neste caso, observamos que o modelo proposto pelo Gartner Group cita a necessidade de estabelecer princípios de governança, enquanto que os outros modelos indicam a necessidade de existirem processos específicos para esta tarefa.

Para quantificar estas diferenças, foram propostos dois indicadores: o consenso relativo ao processo, que corresponde ao percentual de modelos que cita ou descreve o processo, indicando o quanto este é relevante para governança SOA; e a cobertura de um determinado modelo, que indica o percentual de processos do commonGOV descritos ou citados pelo modelo analisado.

No exemplo apresentado na Tabela 13, existe um consenso de 100% entre os modelos quanto a necessidade de implementação de um processo para definição dos princípios de governança.

A avaliação completa pode ser visualizada no trabalho publicado por TEIXEIRA FILHO e AZEVEDO (2014). É importante observar que devido a seu alto nível de abstração, o modelo proposto por NIEMANN et al. (2010) não foi considerado nesta análise.

I.4 Conclusões da análise

Após ser realizada a análise, alguns pontos relevantes puderam ser identificados.

- Todos os modelos indicam a necessidade de processos para gestão do ciclo de vida de serviços, gestão de portfólio de serviços e gestão de estrutura organizacional;
- Não existe consenso sobre a governança das aplicações que consomem serviços e aplicações compostas. Se consideramos a média do consenso dentro do subgrupo Ciclo de Composição de Soluções SOA, obtemos um valor de 52%, deixando clara esta divergência entre os modelos;

- Monitoração é uma preocupação existente em boa parte dos modelos, levando a um consenso médio neste subgrupo de 72%. Porém, existe divergência sobre que elementos devem ser monitorados. Apesar de existir consenso quanto a monitoração de serviços e de contratos de serviços, existem divergências quanto a necessidade de monitoração de segurança, soluções baseadas em SOA e infraestrutura;
- Quanto à conformidade, quase todos os modelos consideram a necessidade de atividades de auditoria e padronização. A única exceção é o modelo da Oracle, que apesar de afirmar que um dos grandes objetivos da governança SOA é obter conformidade, não considera uma atividade específica para este fim.

Apêndice II. Procedimento de revisão de literatura

Neste capítulo será apresentado o procedimento realizado para revisar a literatura para estudo do tema, obtenção de áreas com possíveis problemas e identificação de trabalhos relacionados. O foco inicial do estudo era governança de arquiteturas orientadas a serviço, sendo identificado o problema de conformidade após a análise dos resultados da revisão.

O levantamento foi realizado em três etapas:

- Busca de trabalhos relacionados a área de pesquisa, através de definição de uma *string* de busca, sua calibragem e seu uso em mecanismos de busca;
- Filtragem dos trabalhos, através de eliminação de duplicidades, leitura do resumo do trabalho, leitura das introduções e conclusões dos artigos para selecionar trabalhos com relação ao assunto pesquisado;
- Leitura e análise dos trabalhos filtrados, visando a identificação de problemas e questões de pesquisa.

Tal trabalho foi realizado sobre as bases SCOPUS, IEEEExplore e ACM Digital Library. Cada seção subsequente deste capítulo trata uma destas etapas.

II.1 Busca de Trabalhos

Para realização da busca, o primeiro passo foi definir uma *string* de busca que permitisse a obtenção de artigos relacionados ao tema de governança SOA, visando obter modelos, frameworks, ferramentas e ontologias nesta área

Deste modo, a *string* relativa as bases SCOPUS e IEEEExplore foi formatada como:

```
TITLE-ABS-KEY((soa OR soc OR (service AND oriented AND (architecture OR computing))) AND governance AND (model OR framework OR metamodel OR ontology OR conceptualization OR tool OR environment))
```

Para a base ACM Digital Library, a *string* foi formatada como:

```
((TITLE((SOA OR SOC OR "service oriented computing" OR "service oriented architecture" OR "service-oriented computing" OR "service-oriented architecture") AND (governance) AND (model OR framework OR metamodel OR ontology OR conceptualization OR tool OR environment))OR ABS((SOA OR SOC OR "service oriented computing" OR "service oriented architecture" OR "service-oriented computing" OR "service-oriented architecture") AND (governance) AND (model OR framework OR metamodel OR ontology OR conceptualization OR tool OR environment)) OR KEYWORDS((SOA OR SOC OR "service oriented computing" OR "service oriented architecture" OR "service-oriented computing" OR "service-oriented architecture") AND (governance) AND (model OR framework OR metamodel OR ontology OR conceptualization OR tool OR environment))))
```

Para verificar a eficácia das *Strings*, foram selecionados artigos de conhecimento do autor e/ou dos orientadores e foram realizados testes, visando obter no retorno estes artigos. São estes:

- NIEMANN *et al.* (2009) - artigo sobre desafios e oportunidades no contexto de governança SOA;
- JANIESCH, KORTHAUS e ROSEMANN, 2009 - artigo que versa sobre a definição dos conceitos que devem ser considerados em um contexto de governança SOA;
- FORTIŞ, MUNTEANU e NEGRU (2012) - artigo que versa sobre a definição de mecanismos de governança de soluções de computação em nuvem associadas a orientação a serviços;
- TEIXEIRA FILHO e AZEVEDO (2012) - artigo do autor contemplando uma avaliação dos modelos de governança SOA e a proposta de um modelo comum;
- HOJAJI e SHIRAZI (2010) - artigo que descreve um modelo de governança SOA baseado no COBiT.

Foram realizados ajustes até que os artigos de controle fossem retornados por buscas em pelo menos uma das bases utilizadas.

II.2 Filtragem

Após execução da busca, realizada em Fevereiro de 2013, foram obtidas as quantidades de artigos descritas na Tabela 14.

Tabela 14. Resultados da busca inicial de artigos

Mecanismo de Busca	Quantidade de Artigos retornados
Scopus	192
IEEEExplore	110
ACM DL	252

Dado o número elevado de artigos, foram realizadas etapas de filtragem. O primeiro passo foi a verificação de duplicidade, buscando artigos que constassem do resultado de duas ou mais bases. Com esta etapa, foram excluídos 18 resultados da consulta da IEEEExplore e 11 resultados da consulta da ACM Digital Library.

O passo seguinte foi a leitura do título e da seção de resumo dos artigos remanescentes, visando eliminar artigos que não retornaram na consulta devido apenas a combinação de palavras desejadas, porém sem conteúdo alinhado com o objetivo do estudo. Como critério de inclusão, foi considerado que o artigo deveria tratar pelo menos um dos seguintes assuntos:

- Modelos ou frameworks para governança SOA;
- Modelos conceituais ou ontologias para SOA e governança;
- Aplicação de conceituações ou ontologias para governança;
- Ferramentas ou ambientes para governança SOA.

Com esta etapa, foram mantidos 94 artigos da base SCOPUS, 20 artigos da base IEEEExplore e 11 artigos da base ACM.

Por fim, foi realizada a leitura das introduções e conclusões de cada artigo remanescente, aplicando os mesmos critérios utilizados para avaliação de título e resumo. Com esta atividade, foi obtido um resultado final de 42 artigos na base SCOPUS, 12 da base IEEEExplore e 7 da base ACM. É importante ressaltar que a diferença de quantidades entre as bases é originária de dois fatores:

- Ao eliminar duplicidade entre bases, os artigos duplicados foram mantidos no resultado da pesquisa SCOPUS e eliminados do resultado das outras bases;
- Como o trabalho foi realizado inicialmente na base SCOPUS e depois aplicado as outras bases, ocorreu evolução da avaliação do autor ao realizar a triagem das duas outras bases.

II.3 Análise dos resultados

Para avaliar os resultados e buscar possíveis linhas de pesquisas, foram identificados grupos de assuntos presentes nos artigos e estes foram classificados de acordo com estes grupos. É importante observar que um artigo pode versar sobre mais de um grupo de assunto. O resultado é apresentado na Tabela 15.

Tabela 15. Distribuição por áreas de assunto dos artigos obtidos na revisão

Grupo	Percentual de Artigos
Modelos de Governança	20%
Ontologias para SOA e governança	5%
Aplicações para governança SOA	5%
Ferramentas para governança SOA	27%
Conformidade em governança SOA	12%
Alinhamento entre negócio e TI com SOA	14%
Modelos de maturidade para SOA	9%
Outros assuntos	24%

O grupo com maior volume de publicações é relativo a artigos que propõem ferramentas que facilitem algum processo relacionado a governança de serviços (27%). Neste tema, são propostas ferramentas com funções variando desde o apoio a verificação de conformidade, como explorado no capítulo 2 desta dissertação, até ferramentas de monitoração LESBEGUERIES *et al.* (2012) e modelagem de soluções compostas STEIN, LAUER e IVANOV (2008). Um ponto relevante é que deste universo de trabalhos lidando com ferramentas, a maioria dos artigos (44%) trata de ferramentas para tratar aspectos de conformidade, propondo abordagens diversas, como visto na seção 2 deste trabalho. Tal ponto também é corroborado pelo fato de 12% dos artigos lidarem com conformidade, mesmo sem considerar este termo nas *strings* de busca. Este ponto indica a importância do assunto de conformidade no contexto de SOA.

Outro tema com volume expressivo de publicações diz respeito a modelos de governança, respondendo por 20% dos artigos. Como visto no Apêndice I, existem divergências entre propostas de modelos, indicando um campo de estudo em aberto nesta área.

Outros assuntos que se destacaram foram publicações sobre alinhamento entre negócio e TI alavancados por SOA (14%) e o estudo de modelos de maturidade no contexto de governança SOA (9%).

Como conclusão deste levantamento, é possível verificar que existe um volume expressivo de trabalhos na área de conformidade em SOA e, como apresentado nos capítulos 2 e 6 desta dissertação, existem múltiplas abordagens para este tema, cada qual com seus ganhos e limitações. Esta dissertação foca em uma abordagem nesta área denominada intelliGOV.

Apêndice III. OWL, SWRL e SQWRL

Neste apêndice são descritos os fundamentos das tecnologias utilizadas na implementação da abordagem proposta nesta tese.

III.1 Implementação da ontologia – OWL

Ontologias são descritas através de linguagens de descrição. Para ANTONIOU, FRANCONI e VAN HARMELEN (2005), uma linguagem para descrição de ontologias é um meio de descrever o que é necessariamente verdadeiro em um domínio de interesse, explicitando as restrições que são obrigatoriamente válidas em qualquer cenário deste domínio. Neste trabalho, é utilizada a *Web Ontology Language* (OWL) (HITZLER et al., 2012, p. 2), que é uma linguagem recomendada pela W3C para descrever ontologias. Para seu uso, a ontologia é descrita em um ou mais arquivos padronizados, que podem ser escritos utilizando vários padrões de sintaxe, como uma notação funcional ou o uso de RDF/XML.

A linguagem opera sobre três elementos básicos: classes, que representam os conceitos existentes no domínio; propriedades, que simbolizam as relações entre classes e as relações com elementos de dados; e os indivíduos, que correspondem as instâncias das classes. Além de definir os elementos que compõe a ontologia, é possível incluir axiomas que enriquecem a descrição do domínio.

No caso de classes, é possível declarar diversos tipos de relações entre classes: especialização, utilizando a assertiva *SubClassOf*; equivalência, utilizando a assertiva *EquivalentClasses*; e disjunção, utilizando a assertiva *DisjointClasses*. Um exemplo de aplicação destes axiomas pode ser visto na Figura 22.

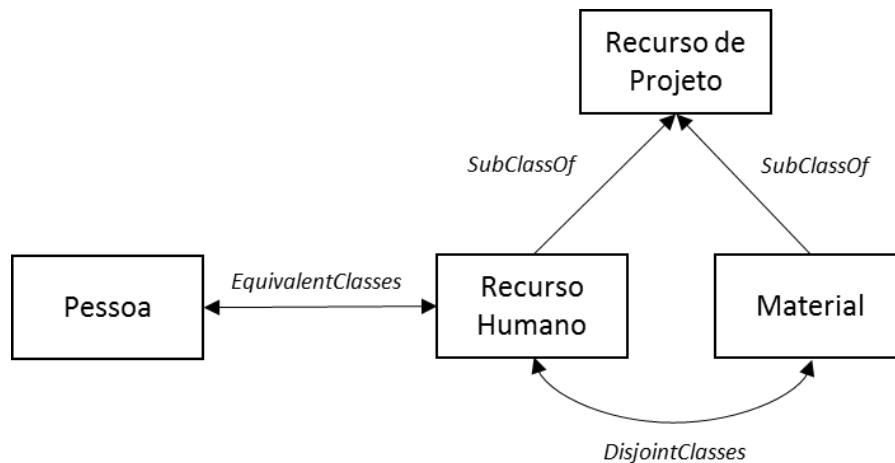


Figura 22. Exemplo de assertivas utilizadas com classes em OWL

Neste caso, tratando o domínio de recursos de projetos, as classes *Recurso Humano* e *Material* são subclasses de *Recurso de Projeto*, indicando que toda instância de recurso humano e material é também um recurso de projeto. Adicionalmente, devido a afirmação de que Pessoas equivalem a Recursos Humanos, todo membro da classe Pessoa neste contexto passa também a ser classificados como Recurso de Projeto. Devido a afirmação que Recursos Humanos e Materiais são disjuntos, um recurso de projeto só pode pertencer a classe de recursos humanos ou a classe de materiais, nunca podendo pertencer a ambas as classes.

Para as propriedades, é possível classifica-las em duas categorias distintas: propriedades de objeto e propriedades de dados. A primeira identifica a ligação entre duas classes ou indivíduos. A segunda identifica a ligação entre uma classe ou indivíduo e uma informação associada a um tipo de dado. Um exemplo pode ser visto na Figura 23.

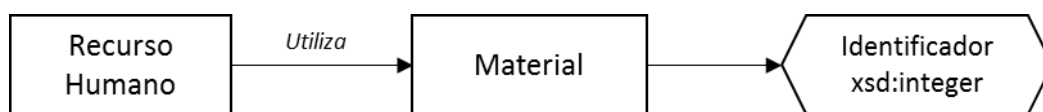


Figura 23. Exemplos de propriedades na OWL

Neste caso, estamos afirmando que um recurso humano pode estar relacionado com um material através de uma propriedade de objeto *Utiliza*, que atesta o uso de um material por uma pessoa. O material, por sua vez pode possuir um identificador, representada por um número do tipo *xsd:integer*. É importante ressaltar que as propriedades de dados podem permitir apenas ligações com valores de dados e não com indivíduos.

No caso de propriedades, assertivas permitem definir semântica das relações. Alguns exemplos destes casos são apresentados na Figura 24.

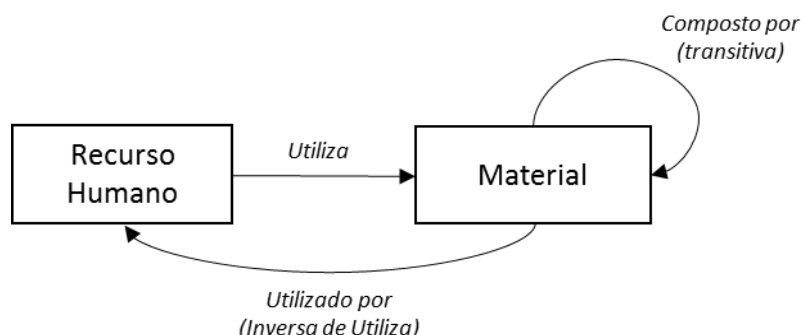


Figura 24. Exemplos de assertivas aplicadas às propriedades em OWL

Neste exemplo, Uma nova propriedade de objeto denominada *utilizado por* é declarada como inversa da propriedade *Utiliza*. Isso significa que caso um recurso humano *Fulano* utilize o material *Engrenagem*, é correto afirmar que *Engrenagem é utilizada por Fulano*. Outra propriedade de objeto utilizada é *Composto por*, declarada como transitiva. Com isso, caso existam três indivíduos da classe material, denominados *Turbina*, *Motor e Cilindro*, e *Turbina* seja composta de *Motor e Motor* seja composta por *Cilindro*, isso implica em *Turbina é composta por Cilindro*.

Outro tipo de assertiva importante relacionada às propriedades é a que descreve o domínio e a faixa de valores permitidos em uma associação. No exemplo apresentado, é possível descrever que *Utiliza* tem como domínio a classe recursos humanos, permitindo como faixa de valores indivíduos da classe Material.

Por fim, considerando os indivíduos, mais semântica pode ser incluída no modelo utilizando assertivas do tipo *sameIndividual* e *differentIndividuals*. Deste modo, dois indivíduos denominados *Fulano* e *Ciclano*, associados através da relação *sameIndividual* são interpretadas como a mesma pessoa.

Um ponto importante é que existem ferramentas abertas para edição e manipulação de arquivos OWL, além de bibliotecas compatíveis de inferência que permitem classificar os elementos que compõe uma ontologia ou verificar sua integridade. Baseado no exemplo apresentado nesta seção, são listadas a seguir algumas possíveis inferências que poderiam ser realizadas:

- Os indivíduos *Fulano*, *Turbina e Motor* são recursos de projeto;
- *Fulano* é uma pessoa;
- *Ciclano*, que equivale a *Fulano*, é uma pessoa e um recurso humano;
- *Ciclano* utilizou *Turbina* em seu projeto.

É importante observar que não foi necessário declarar nenhuma destas assertivas. Todas foram derivadas a partir dos axiomas descritos na ontologia.

III.2 Implementação das regras e consultas – SWRL e SQWRL

Para implementação das políticas, foram selecionadas as linguagens *Semantic Web Rule Language (SWRL)* (HORROCKS et al., 2004) e *Semantic Query-enhanced Web Rule Language (SQWRL)* (O’CONNOR; DAS, 2009), dada sua compatibilidade com a OWL e com as bibliotecas utilizadas para sua manipulação.

A linguagem SWRL foi criada com o objetivo de complementar os axiomas de uma ontologia com regras no formato *antecedente* \rightarrow *consequente*, permitindo desta forma enriquecer o conjunto de assertivas da ontologia. Os termos antecedente e consequente são compostos por zero ou mais átomos, sendo que um antecedente representa uma condição sempre verdadeira e um consequente nulo representa uma condição sempre falsa.

Os átomos podem representar classes da ontologia, propriedades de objetos e dados, assertivas do tipo *sameAs* ou *differentFrom* e podem utilizar funções denominadas *built-ins* para realizar operações matemáticas, comparações manipulações de Strings, datas e listas.

Um exemplo de regra em SWRL é exibido no exemplo a seguir, baseado na ontologia apresentada como exemplo na seção anterior:

$$\text{RecursoHumano}(?r) \wedge \text{Material}(?m) \wedge \text{Utiliza}(?r, ?m) \rightarrow \text{MaterialEmUso}(?m)$$

Neste caso, estamos afirmando que sempre que um *recurso humano* for relacionado a um *material* através da propriedade *utiliza*, este *material* poderá ser classificado como um *material em uso*. Observe que esta assertiva depende da criação de uma nova classe na ontologia, representando os *materiais em uso* e que o uso desta regra permite determinar que indivíduos da classe *material* pertencem a esta classe.

Outro exemplo, considerando *built-ins* é apresentado a seguir.

$$\text{Pessoa}(?p) \wedge \text{temIdade}(?p, ?i) \wedge \text{swrlb:lessThanOrEqual}(?i, 18) \rightarrow \text{Menor}(?p)$$

Neste caso, se uma *pessoa* possuir uma propriedade de dado *idade* e esta for *menor ou igual a 18 anos*, esta *pessoa* será classificada como *menor de idade*.

Porém é importante observar que o SWRL permite classificar elementos, mas não é uma linguagem de consulta. Para permitir consultas neste cenário, seria necessário executar uma regra, classificar os elementos em uma nova classe e listar os elementos nesta classe. Adicionalmente, o SWRL não dispõe de operações para agregar dados ou para manipular conjuntos. Considerando os exemplos apresentados, se fosse necessário saber quantos *materiais* se relacionam a um determinado *recurso humano*, esta operação não seria possível. Outro caso seria comparar se um mesmo *material* está associado a mais de uma *pessoa*.

Em termos de linguagens para consulta em ontologias, uma opção é o uso de SPARQL. Porém, como pode ser verificado em O'CONNOR e DAS (2009), esta não contempla todos os elementos disponíveis na OWL, operando sobre a camada RDF.

Para contornar estas limitações, O'CONNOR e DAS (2009) propuseram a inclusão de *built-ins* no SWRL para possibilitar consultas, ampliando a linguagem, criando a SQWRL. A estrutura básica de uma consulta SQWRL pode ser vista a seguir:

$$\text{Pessoa}(?p)^{\wedge}\text{temIdade}(?p,?i)^{\wedge}\text{swrlb:lessThanEqual}(?i,18) \rightarrow \text{sqwrl:select}(?p)$$

Neste caso, ao invés de criar uma nova classe *menor* para listar as *pessoas* com *idade menor ou igual a dezoito anos*, foi criada uma consulta, utilizando como consequente o *built-in* `sqwrl:select`. Caso se desejasse aproveitar a regra SWRL apresentada anteriormente, esta consulta poderia ser reescrita como:

$$\text{Menor}(?p) \rightarrow \text{sqwrl:select}(?p)$$

Outro ponto importante é a possibilidade de realizar operações de contagem de elementos utilizando a linguagem de consulta. Um exemplo é apresentado a seguir, permitindo obter a quantidade de *materiais utilizados* por uma *pessoa*.

$$\text{Pessoa}(?p) \wedge \text{Material}(?m) \wedge \text{utiliza}(?p, ?m) \rightarrow \text{sqwrl:select}(?p) \wedge \text{sqwrl:count}(?m)$$

Outra possibilidade na SQWRL é a possibilidade de compor, manipular e comparar conjuntos. Para tal, é utilizado o *built-in* `sqwrl:makeSet` e operações para agrupamento e comparação de conjuntos. Um exemplo é apresentado a seguir.

$$Pessoa(?p) \wedge Material(?m) \wedge utiliza(?p, ?m) \wedge sqwrl:makeSet(?s, ?m) \wedge sqwrl:groupBy(?s, ?p) \wedge sqwrl:size(?t, ?s) \wedge swrlb:greatThan(?t, 2) \rightarrow sqwrl:select(?p)$$

Os três primeiros elementos da expressão permitem restringir o conjunto analisado a *peessoas* que *utilizam materiais*. O termo `sqwrl:makeSet` cria um conjunto *s*, contendo todos os *materiais* que são *utilizados* por *peessoas*. O termo `sqwrl:groupBy` agrupa os elementos deste conjunto por *peessoas*. O termo `sqwrl:size` calcula o tamanho *t* do conjunto, levando em conta o agrupamento indicado anteriormente. Por fim, o tamanho do conjunto deve ser maior que 2 para satisfazer a consulta, condição verificada pelo *built in* da SWRL `swrlb:greatThan`. Deste modo, a consulta estará retornando *peessoas* que utilizem mais do que dois *materiais*. Tal cálculo realizado por *peessoa* é permitido pelo agrupamento `sqwrl:groupBy`.

Por fim, é possível utilizar *built-ins* para realizar operações de diferença e adição de conjuntos, consolidação e contagem de elementos. Os operadores utilizados nesta dissertação estão listados na Tabela 16, descrevendo na primeira coluna a sua sintaxe e na segunda coluna o seu funcionamento.

Tabela 16. *Built-ins* da SQWRL utilizados nesta dissertação

Operador	Significado
<code>sqwrl:makeSet(?s, ?e1)</code>	Cria um conjunto <i>s</i> de elementos do tipo <i>e1</i>
<code>sqwrl:groupBy(?s, ?e2)</code>	Agrupa um conjunto <i>s</i> por elementos do tipo <i>e2</i> . É importante ressaltar que para agrupamento, pode ser considerado qualquer elemento que se relacione através de uma propriedade com os elementos que constituem o conjunto
<code>sqwrl:size(?t, ?s)</code>	Calcula o tamanho <i>t</i> de um conjunto <i>s</i>
<code>sqwrl:difference(?sd, ?s1, ?s2)</code>	Calcula a diferença entre os conjuntos <i>s1</i> e <i>s2</i> e preenche o resultado em <i>sd</i> . É importante ressaltar que elementos existentes em <i>s2</i> que não existam em <i>s1</i> serão desconsiderados nesta operação.
<code>sqwrl:equal(?s1, ?s2)</code>	Retorna verdadeiro se o conjunto <i>s2</i> conter todos os elementos do conjunto <i>s1</i> .

Apêndice IV. Ontologia utilizada no estudo de caso

Conforme citado na seção 4.4.2.7, a ontologia utilizada no estudo de caso foi composta a partir da ontologia do Open Group. Neste apêndice são apresentadas a ontologia do Open Group e os refinamentos aplicados para atender aos requisitos do estudo de caso.

A ontologia proposta pelo Open Group (2010) tem como objetivo descrever os conceitos que constituem o domínio de arquiteturas orientadas a serviço. Os conceitos propostos, suas propriedades de objeto e de dados são apresentadas na Figura 25.

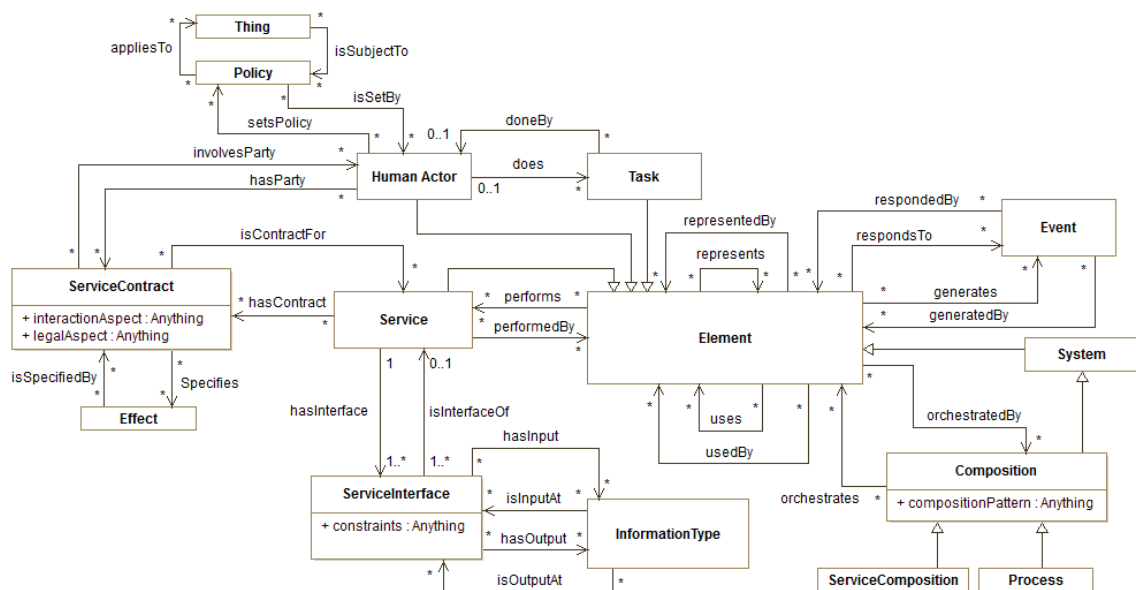


Figura 25. Ontologia do Open Group, adaptada de OpenGroup (2010)

Os conceitos que compõe a ontologia são listados a seguir.

- Elementos (*Element*): Representam todos os elementos que podem compor uma arquitetura orientada a serviços. Um elemento está relacionado a outro elemento através de propriedades que representam seu uso (*uses/isUsedBy*). Eles também podem estar relacionados através

de relações de representação que permitem afirmar que um determinado elemento corresponde a uma abstração de outro elemento (*Represents/representedBy*). Elementos são *opacos* uma vez que apresentam-se como caixa preta para seus usuários e expõem características internas para fins de gestão, descoberta e identificação;

- Sistemas (*System*): Coleções organizadas de elementos. Como é uma subclasse de elemento, um sistema pode usar outros sistemas. Neste contexto, a propriedade *uses/usedBy* simboliza que um sistema pode ser composto por outros elementos, utilizando-os;
- Atores Humanos (*Human Actor*): Representam pessoas ou organizações. A relação de representação neste caso simboliza a possibilidade de uma organização (instância de *Human Actor*) representar diversas pessoas distintas (outras instâncias de *Human Actor*);
- Tarefas (*Tasks*): Um elemento Tarefa consiste de uma atividade atômica que leva à um resultado. Tarefas são realizadas por instâncias de atores humanos. Por conta deste fator, se associam através de propriedades *does/doneBy* com estes elementos;
- Serviço (*Service*): Um serviço corresponde à representação lógica de uma atividade repetível que gera um resultado. Um serviço é autocontido e seus detalhes de implementação são ocultos de seus usuários. Um serviço é uma representação, ou seja, o serviço é efetivamente executado por algum outro elemento. Esta relação é representada pela propriedade *performs/performedBy*. As relações de consumo e provimento de serviço são representadas respectivamente por propriedades *uses* e *performs*, não existindo explicitamente conceitos que representem provedores e consumidores;
- Contrato de Serviço (*Service Contract*): O contrato de serviço representa os termos, condições e regras de interação a serem seguidas por seus participantes para utilizarem o serviço. Este conceito possui propriedades de objeto para referenciar o serviço que é regulado por este contrato (*isContractFor/hasContract*) e para indicar quais *Human Actors* são partes do contrato (*involvesParty/isPartyTo*);

- Efeito (*Effect*): Representa o resultado da execução de um serviço, de acordo com as condições expressas no contrato, simbolizando o que se espera após a execução do serviço. Possui propriedades de objeto que indicam que estes são especificados por contratos (*isSpecifiedBy/Specifies*);
- Interface do Serviço (*Service Interface*): Define como as partes podem interagir com o serviço e como estas enviam e recebem informações. Possui uma propriedade de dados denominada *Constraints*, que simboliza eventuais restrições nos dados a serem trocados, e se relaciona com o conceito serviço através de propriedades de objeto *isInterfaceOf/hasInterface*;
- Tipo de Informação (*Information Type*): Representam os tipos de informações a serem trocadas pelo serviço e seus consumidores. São relacionadas com as interfaces de serviços através de propriedades de objeto do tipo *IsInputAt/hasInput* para dados de entrada e *isOutputAt/hasOutput* para dados de saída;
- Composição (*Composition*): Representa o resultado da composição de elementos. Associa-se a estes através de propriedades de objeto *orchestrates/orchestratedBy*, que representam o conjunto de elementos que faz parte da composição;
- Composição de Serviços (*Service Composition*): Subclasse de Composição que representa composições de serviços. Representa um *performer* da composição;
- Processo (*Process*): Processos que compõe funções a partir de todos os possíveis elementos – atores humanos, tarefas, sistemas e serviços;
- Política (*Policy*): Uma política é uma diretriz ou modo de comportamento definido por algum elemento do tipo *Human Actor* e esperado de algum elemento que seja parte da arquitetura. É relacionado com a classe *Thing* através de propriedades de objeto do tipo *appliesTo/isSubjectTo* e com *HumanActor* através de propriedades de objeto *setsPolicy/isSetBy*;
- Evento (*Event*): Representa algum acontecimento que pode ser respondido ou gerado por algum elemento. Permitem conhecer como

elementos se comportam através de suas reações e eventos gerados. Relacionam-se com elementos através de propriedades de objeto *respondsTo/respondedBy*, que simbolizam a lista de eventos aos quais um elemento reage; e *generates/generatedBy*, que correspondem à determinação de que elemento gera um determinado evento.

Para alguns conceitos foram definidas propriedades de dados, visando capturar características associadas a estes conceitos. Estas propriedades são listadas a seguir.

- Para os contratos de serviços, foram definidas duas propriedades de dados: *interactionAspect*, que contém informações a respeito das interações previstas; e *legalAspects*, que contém informações relativas às restrições legais aplicáveis a serem consideradas;
- Para as composições, é considerada a propriedade *compositionPattern*, que indica a forma como os elementos são agregados à coleção. Para contexto de SOA são considerados três padrões básicos: orquestração, no qual existe uma entidade controladora e o fluxo é pré-definido; coreografia, no qual não existe uma entidade controladora, porém existe um fluxo pré-definido do qual todas as partes têm ciência; e colaboração, no qual não existe uma entidade controladora e não existe um fluxo pré-definido, sendo este gerado dinamicamente de acordo com o comportamento dos envolvidos.

Para descrever os conceitos do intelliGOV, a ontologia do Open Group foi complementada com os conceitos descritos na seção 4.4.2.7.

Apêndice V. **Problemas identificados no estudo de caso**

Neste apêndice, será detalhada a análise do estudo de caso, demonstrando os dados coletados, divergências identificadas e resultado da discussão relativa a cada divergência.

Para tal, cada seção a seguir apresenta os resultados obtidos para avaliação de cada política, considerando a avaliação manual, a avaliação utilizando a abordagem proposta e o resultado das discussões com o grupo. Para analisar as divergências encontradas, foram analisadas as justificativas apresentadas pelos participantes e esclarecimentos ocorridos ao longo da etapa de discussão junto aos participantes do estudo de caso.

V.1 Política 1

A política trata a questão referente a classificação correta de informações por áreas de assunto. O resultado obtido nas avaliações é apresentado na Tabela 17, organizada em cinco colunas: a primeira representa o nome do serviço, que foi codificado neste trabalho para garantir a segurança dos dados da organização; a segunda o analista responsável pela avaliação manual; a terceira contendo o status de conformidade identificado pelo avaliador, sendo OK para conforme e X para não conforme; a quarta o resultado obtido com a abordagem deste trabalho; e a sexta o resultado esperado, descrito no gabarito de avaliação. Os itens que apresentam divergências foram grifados em cinza.

Tabela 17. Resultado da verificação da política 1

Serviço	Avaliador	Avaliação Manual	Avaliação intelliGOV	Resultado Gabarito
BIGE	Analista 4	OK	OK	OK
BUPLA	Analista 4	X	X	X
WFORCE	Analista 5	OK	OK	OK
COMPANY	Analista 5	OK	X	X
VENDOR	Analista 5	OK	OK	OK
ORGUNI	Analista 5	OK	OK	OK
DIP	Analista 6	OK	X	X
MENTOR	Analista 6	OK	OK	OK
VACPLN	Analista 6	OK	OK	OK
SDORDER	Analista 7	OK	X	X
REMCAR	Analista 7	X	X	X
SDINVOIC	Analista 7	X	X	X
NOTESKEY	Analista 7	OK	OK	OK
CATAPL	Analista 7	OK	OK	OK
SALDO	Analista 8	OK	OK	OK
STOCKS	Analista 8	X	OK	OK
REMBEN	Analista 8	X	X	X
ESTREF	Analista 8	OK	OK	OK
PLATA	Analista 9	OK	X	X
SONDA	Analista 9	OK	X	X
BUNKER	Analista 9	OK	OK	OK
CONTA	Analista 9	OK	OK	OK
EMAIL	Analista 10	OK	OK	OK
CALEND	Analista 10	OK	OK	OK
VENPAY	Analista 10	OK	OK	OK

Como pode ser observado na tabela acima, ocorreram erros na avaliação manual para os serviços COMPANY, DIP, SDORDER, STOCKS, PLATA e SONDA. Para identificar a causa destas ocorrências, foram tabuladas as áreas de assunto do serviço, entradas e saídas. O resultado é apresentado na Tabela 18.

Tabela 18. Dados dos serviços divergentes para a política 1

Serviço	Área de Assunto Serviço	Área de Assunto Entrada	Área de Assunto Saída
COMPANY	SUPPLY	CORP-ORG	CORP-ORG
DIP	TI	CORP-PROC	CORP-PROC
SDORDER	DWNSTR-SALES	DWNSTR-SALES	DWNSTR-SALES E DWNSTR-PRODUCT
STOCKS	DWNSTR	DWNSTR-PRODUCT	DWNSTR-LOG
PLATA	UPSTR-ASSET	UPSTR	UPSTR
SONDA	UPSTR-ASSET	UPSTR	UPSTR

Para o serviço COMPANY, o analista julgou que este tratava de informações relativas a fornecedores e julgou que a classificação na área SUPPLY, referente a suprimentos, estava correta. Porém, o serviço trata de empresas subsidiárias e deveria estar classificado na área CORP, que trata dos dados de organização da empresa. Tal ponto já era indicado pela classificação dos dados envolvidos, porém o analista considerou apenas a classificação do serviço.

Para o serviço DIP, o erro foi semelhante, porém considerando áreas diferentes. O analista julgou a classificação TI correta, porém o serviço, responsável por emitir documentos formais na empresa, manipula informações da área CORP-PROC.

Para o serviço SDORDER, o problema diz respeito a ausência de áreas na classificação do serviço. Este foi classificado na área de assunto DWNSTR-SALES porém manipula dados das áreas DWNSTR-SALES e DWNSTR-PRODUCT. O analista julgou que a classificação levando em conta apenas uma das áreas estava correta.

Para o serviço STOCKS, o erro foi ocasionado por não ter sido considerada a hierarquia de áreas de assunto. A área DWNSTR é uma área que consolida as áreas DWNSTR-PRODUCT, DWNSTR-LOG e DWNSTR-SALES. Pela política definida pela empresa, caso um serviço esteja associado a área “pai” e os dados pertençam a subáreas, o serviço estaria conforme. Deste modo, como o serviço STOCKS está associado a área consolidadora (DWNSTR) e os dados a duas subáreas desta área (DWNSTR-LOG e DWNSTR-PRODUCT), o serviço deveria estar conforme. O analista porém considerou os nomes das áreas, ignorando a hierarquia existente.

Os dois últimos serviços, PLATAF e SONDA, também foram classificados como conformes erroneamente devido a questões de hierarquia das áreas de assunto. Porém, neste caso, os dados estavam classificados de maneira mais abrangente que o serviço, tornando a classificação deste mais restrita do que as informações que este manipula, tornando o serviço não conforme com a política. O analista considerou que os dados pertenciam a mesma área e classificou ambos como conformes.

Ao aplicar o intelliGOV, todos os serviços foram corretamente classificados. Os dois primeiros serviços foram classificados como não conformes devido a simples comparação entre as áreas envolvidas. Os quatro outros serviços levaram em conta a hierarquia das áreas de assunto, através do uso da propriedade de objeto *handlesDataFrom*, que foi declarada na ontologia como transitiva, permitindo assim que a

avaliação levasse em conta não apenas a área associada ao serviço, mas todas as subáreas relacionadas.

Para tal, no caso do serviço SDORDER, aplicando intelliGOV, este foi classificado apenas na área DWNSTR-SALES, sem levar em conta a área DWNSTR-PRODUCT, o que levou o serviço a ser considerado como não conforme, de acordo com o gabarito.

Já para o serviço STOCKS, o uso da abordagem identificou que o serviço estava relacionado não apenas a área DWNSTR, mas também a todas as subáreas de assunto relacionadas, incluindo DWNSTR-PRODUCT e DWNSTR-LOG, o que permitiu classificar o serviço como conforme, uma vez que este está associado a área de assunto relacionada com as informações que manipula. Neste caso, a transitividade da relação foi fundamental para identificar a área de assunto, uma vez que através desta, foi possível associar o serviço não apenas a área “pai” da hierarquia, mas também a todas as áreas filhas.

Para os serviços PLATAF e SONDA, a abordagem considerou que o serviço manipulava dados da subárea UPSTR-ASSET e que os dados compreendem uma área maior, denominada UPSTR, que contém, além da área UPSTR-ASSET, outras áreas de assunto, tornando a abrangência dos dados maior que a abrangência do serviço.

Deste modo, podemos ver que o uso de transitividade na relação *handlesDataFrom* tornou o diagnóstico mais preciso.

V.2 Política 2

Para a política 2, na qual se verificava se todo o serviço estava classificado do ponto de vista de segurança de informação, foi realizada a mesma análise. Os resultados são apresentados na Tabela 19, com estrutura análoga a Tabela 17, apresentada na seção IV. Novamente, os pontos de divergência foram marcados em cinza.

Tabela 19. Resultados da verificação da política 2

Serviço	Avaliador	Avaliação Manual	Avaliação intelliGOV	Resultado Gabarito
BIGE	Analista 4	X	OK	OK
BUPLA	Analista 4	X	X	X
WFORCE	Analista 5	OK	X	X
COMPANY	Analista 5	X	X	X
VENDOR	Analista 5	OK	OK	OK
ORGUNI	Analista 5	OK	X	X
DIP	Analista 6	OK	OK	OK
MENTOR	Analista 6	OK	OK	OK
VACPLN	Analista 6	X	X	X
SDORDER	Analista 7	OK	OK	OK
REMCAR	Analista 7	X	X	X
SDINVOIC	Analista 7	OK	OK	OK
NOTESKEY	Analista 7	OK	OK	OK
CATAPL	Analista 7	OK	X	X
SALDO	Analista 8	X	X	X
STOCKS	Analista 8	OK	X	X
REMBEN	Analista 8	X	X	X
ESTREF	Analista 8	OK	OK	OK
PLATA	Analista 9	X	X	X
SONDA	Analista 9	X	X	X
BUNKER	Analista 9	X	X	X
CONTA	Analista 9	X	X	X
EMAIL	Analista 10	X	OK	OK
CALEND	Analista 10	X	OK	OK
VENPAY	Analista 10	X	OK	OK

Como pode ser visto acima, ocorreram erros de avaliação para os serviços BIGE, WFORCE, ORGUNI, CATPL, STOCKS, EMAIL, CALEND e VENPAY. Para analisar as causas dos erros, foram levantados os níveis de segurança associados a cada um dos serviços envolvidos, apresentados na Tabela 20.

Tabela 20. Níveis de segurança dos serviços com divergências

Serviço	Nível de Segurança
BIGE	Reservado
WFORCE	Res
ORGUNI	Res
CATPL	Public
STOCKS	2
EMAIL	Corporativo
CALEND	Corporativo
VENPAY	Reservado

Com base nestes dados, foi debatido com a equipe as razões dos erros.

Em primeiro lugar, foi percebida uma falha de padronização na descrição dos níveis de segurança, que levou a abreviaturas, traduções e codificações não institucionalizadas dos níveis, cadastradas pelas pessoas que redigiram a documentação dos serviços. Tal fato surge nos termos Res (abreviatura de Reservado) nos serviços WFORCE e ORGUNI, no termo Public (do inglês, Público) no serviço CATPL e o número 2 (representando NP-2) para o serviço STOCKS. Nestes casos, os avaliadores consideraram as abreviaturas corretas e marcaram os serviços como Conformes erroneamente.

A segunda categoria de erro dizia respeito a serviços que estavam utilizando a classificação antiga: BIGE, EMAIL, CALEND e VENPAY. Nestes casos, os dois analistas, com baixa experiência na empresa, consideraram os serviços não conformes, uma vez que haviam sido orientados apenas no modelo novo de classificação e esperavam encontrar valores NP-2 e NP-3 para os serviços, ao invés de Corporativo e Reservado. Porém, conforme discutido na seção 4.4.2.2.2, este mapeamento deveria ser considerado na avaliação, tornando os serviços avaliados conformes.

Com o uso do intelliGOV, foram incluídas relações do tipo *sameAs* entre as instâncias equivalentes da classe *SecurityLevel*. Adicionalmente, a interface de carga gerava o relacionamento *definesSecurityConstraints* apenas para os serviços com classificações de informação anteriormente cadastradas, não permitindo a geração de novas instâncias de *SecurityLevel* para níveis inadequados. Deste modo, os serviços WFORCE, ORGUNI, CATAPL e STOCKS foram classificados como não conformes de acordo com o gabarito e os serviços BIGE, EMAIL, CALEND e VENPAY como conformes.

Ocorreu discussão no grupo quanto ao uso do termo em inglês *Public* para classificar um nível, uma vez que a empresa atua globalmente. O redator do gabarito informou que mesmo com atuação global, a orientação da empresa é o uso do idioma português na elaboração de documentação, critério este que levou a classificação como não conforme. O autor informou que, caso se julgasse necessário, novas instâncias de *SecurityLevel* representando os níveis em outros idiomas poderiam ser cadastradas, efetuando-se a associação entre estas novas instâncias e as instâncias em Português, facilitando o trabalho de verificação de conformidade.

V.3 Política 3

Para a política 3, que tratava a aplicação de mecanismos de criptografia de canal para serviços com nível de segurança maior ou igual a NP-2, foram obtidos os resultados apresentados na Tabela 21, organizada de maneira semelhante a Tabela 17, apresentada no item IV. Os itens divergentes foram marcados em cinza.

Tabela 21. Resultados da verificação da política 3

Serviço	Avaliador	Avaliação Manual	Avaliação intelliGOV	Resultado Gabarito
BIGE	Analista 4	X	X	X
BUPLA	Analista 4	X	X	X
WFORCE	Analista 5	OK	X	X
COMPANY	Analista 5	X	X	X
VENDOR	Analista 5	OK	OK	OK
ORGUNI	Analista 5	OK	X	X
DIP	Analista 6	OK	OK	OK
MENTOR	Analista 6	X	X	X
VACPLN	Analista 6	X	X	X
SDORDER	Analista 7	OK	OK	OK
REMCAR	Analista 7	X	X	X
SDINVOIC	Analista 7	X	X	X
NOTESKEY	Analista 7	OK	OK	OK
CATAPL	Analista 7	X	X	X
SALDO	Analista 8	X	X	X
STOCKS	Analista 8	X	X	X
REMBEN	Analista 8	X	X	X
ESTREF	Analista 8	OK	X	X
PLATA	Analista 9	X	X	X
SONDA	Analista 9	X	X	X
BUNKER	Analista 9	X	X	X
CONTA	Analista 9	X	X	X
EMAIL	Analista 10	OK	OK	OK
CALEND	Analista 10	OK	OK	OK
VENPAY	Analista 10	X	X	X

Neste caso, apenas dois serviços apresentaram divergências: ORGUNI e ESTREF. Para compreender a causa das divergências, foram levantados os níveis de segurança, protocolo e mecanismos de criptografia empregados para cada serviço. Os resultados são apresentados na Tabela 22.

Tabela 22. Dados dos serviços com divergências de avaliação quanto a política 3

Serviço	Nível de Segurança	Protocolo	Mecanismo de Criptografia
WFORCE	Res	HTTP	Nenhum
ORGUNI	Res	HTTP	Nenhum
ESTREF	Reservado	HTTP	Nenhum

Nos dois primeiros serviços existem dois erros: primeiro a especificação do nível de segurança, descrito em forma abreviada como *Res* e não *Reservado*, conforme especificado na organização; e segundo, mesmo que a abreviatura fosse válida, o protocolo utilizado, HTTP, não implementa criptografia de canal. Como o nível Reservado equivale ao nível NP-2 na nova classificação, o serviço deveria utilizar protocolo que implementasse criptografia de canal. Em consulta ao analista responsável pelas avaliações, o mesmo disse que se confundiu no mapeamento dos níveis, devido a desconhecimento do modelo antigo de classificação.

Erro semelhante ocorreu no serviço ESTREF. Apesar de usar terminologia correta de classificação, o analista não interpretou *Reservado* como semelhante a NP-2, levando ao parecer de que o protocolo HTTP seria adequado.

Ao aplicar intelliGOV, os três serviços foram classificados corretamente como não conformes. Para os serviços WFORCE e ORGUNI, esta conclusão foi originária da abreviatura inexistente *Res*, que impediu a identificação correta do nível de segurança do serviço. Já para o serviço ESTREF, a associação do nível *Reservado* ao nível NP-2 através da propriedade *sameAs* aplicada as instâncias permitiu inferir que se tratavam do mesmo nível e foi verificado, corretamente, que o serviço utilizava um protocolo inadequado.

V.4 Política 4

Para a política 4, que trata a questão da garantia da coerência das interfaces entre variantes de uma mesma release de serviços, foram obtidos os resultados apresentados na Tabela 23, seguindo a mesma estrutura da Tabela 17, apresentada na seção IV. As divergências são novamente sinalizadas em cinza.

Tabela 23. Resultados da verificação da política 4

Serviço	Avaliador	Avaliação Manual	Avaliação intelliGOV	Resultado Gabarito
BIGE	Analista 4	OK	X	X
BUPLA	Analista 4	OK	OK	OK
WFORCE	Analista 5	OK	OK	OK
COMPANY	Analista 5	OK	X	X
VENDOR	Analista 5	OK	OK	OK
ORGUNI	Analista 5	OK	OK	OK
DIP	Analista 6	OK	OK	OK
MENTOR	Analista 6	OK	OK	OK
VACPLN	Analista 6	OK	OK	OK
SDORDER	Analista 7	OK	OK	OK
REMCAR	Analista 7	OK	OK	OK
SDINVOIC	Analista 7	X	OK	OK
NOTESKEY	Analista 7	OK	OK	OK
CATAPL	Analista 7	OK	OK	OK
SALDO	Analista 8	OK	OK	OK
STOCKS	Analista 8	OK	OK	OK
REMBEN	Analista 8	OK	OK	OK
ESTREF	Analista 8	OK	OK	OK
PLATA	Analista 9	OK	OK	OK
SONDA	Analista 9	OK	OK	OK
BUNKER	Analista 9	OK	OK	OK
CONTA	Analista 9	OK	OK	OK
EMAIL	Analista 10	OK	OK	OK
CALEND	Analista 10	OK	OK	OK
VENPAY	Analista 10	OK	OK	OK

Foram identificados três serviços com divergências: BIGE, COMPANY e SDINVOIC. Para avaliar as razões das divergências, foi analisada a mudança que ocorreu entre as variantes dos serviços e foi solicitado aos analistas responsáveis porque julgavam conformes ou não conformes os serviços. O resultado é apresentado na Tabela 24.

Tabela 24. Alterações relativas as divergências na avaliação da política 4

Serviço	Mudança ocorrida	Justificativa
BIGE	Os <i>namespaces</i> dos arquivos XSD utilizados pelos serviços variaram de http://schemata.organizacao.br/GE/1.0 ²⁵ para http://schemata.organizacao.br/GE/1.1	O analista julgou que as alterações aplicadas não alteravam a interface de serviço
COMPANY	Os <i>namespaces</i> dos arquivos XSD utilizados pelos serviços variaram de http://schemata.organizacao.br/CORP/1.0 para http://schemata.organizacao.br/CORP/1.2	O analista julgou que as alterações aplicadas não alteravam a interface de serviço
SDINVOIC	Foram alterados campos de documentação do arquivo XSD utilizado pelo serviço	Como a ferramenta de verificação de diferenças do ClearCASE indicou alteração no arquivo XSD, o analista julgou que esta alteração implicaria em mudança de assinatura do serviço.

Para avaliação com o intelliGOV, os nomes dos *namespaces* foram considerados, uma vez que estes nomes são considerados ao ser realizada a carga, gerando instâncias diferentes de *information type* para cada elemento de dados presente em um *namespace* diferente. Deste modo, os serviços BIGE e COMPANY foram corretamente classificados pela abordagem.

Esta mesma carga não levou em conta os campos de documentação, uma vez que estes não impactam em mudança nos consumidores dos serviços no momento em que estes realizam chamadas no serviço. Com isso, o serviço SDINVOIC foi classificado corretamente.

V.5 Política 5

A quinta política verifica se ocorrem mudanças efetivas na interface de entrada e saída dos serviços ao ser gerada uma nova release. Os resultados da verificação de conformidade são apresentados na Tabela 25, com estrutura semelhante à utilizada pela Tabela 17, apresentada na seção IV. As divergências são destacadas na cor cinza.

²⁵ Neste texto, todos os nomes de *namespaces* citados serão alterados para conterem a palavra *organização* ao invés do nome da empresa para manter o seu anonimato.

Tabela 25. Resultados da verificação da política 5

Serviço	Avaliador	Avaliação Manual	Avaliação intelliGOV	Resultado Gabarito
BIGE	Analista 4	OK	X	X
BUPLA	Analista 4	OK	OK	OK
WFORCE	Analista 5	OK	X	X
COMPANY	Analista 5	OK	OK	OK
VENDOR	Analista 5	OK	X	X
ORGUNI	Analista 5	OK	OK	OK
DIP	Analista 6	X	OK	OK
MENTOR	Analista 6	OK	OK	OK
VACPLN	Analista 6	OK	OK	OK
SDORDER	Analista 7	OK	X	X
REMCAR	Analista 7	OK	OK	OK
SDINVOIC	Analista 7	OK	OK	OK
NOTESKEY	Analista 7	OK	X	X
CATAPL	Analista 7	OK	OK	OK
SALDO	Analista 8	OK	OK	OK
STOCKS	Analista 8	OK	OK	OK
REMBEN	Analista 8	OK	OK	OK
ESTREF	Analista 8	OK	X	X
PLATA	Analista 9	OK	OK	OK
SONDA	Analista 9	OK	OK	OK
BUNKER	Analista 9	OK	OK	OK
CONTA	Analista 9	OK	OK	OK
EMAIL	Analista 10	OK	OK	OK
CALEND	Analista 10	OK	X	X
VENPAY	Analista 10	OK	OK	OK

Foram identificadas divergências nos serviços BIGE, WFORCE, VENDOR, DIP, SDORDER, NOTESKEY, ESTREF e CALEND. Para analisar as causas destas divergências, foram identificadas as alterações realizadas entre as releases dos serviços e a justificativa dos analistas para suas avaliações. Estes resultados são apresentados na Tabela 26.

Tabela 26. Alterações relativas as divergências na avaliação da política 5

Serviço	Alteração	Justificativa
BIGE	Alteração de namespaces ocorreu na variante e não na release	Analista interpretou a variante como se fosse a <i>release</i> do serviço
WFORCE	Alteração em organização de esquemas, sem alteração da estrutura de dados existente, visando uma compatibilidade com plataforma .NET.	Analista interpretou a mudança como nova release devido ao uso de ferramenta de identificação de diferenças no clearcase
VENDOR	Otimização da rotina de busca, sem alteração de interface de entrada e saída	Analista julgou que esta mudança demandaria a criação de uma nova release
DIP	Inclusão de novo campo opcional na assinatura do serviço	Analista julgou que esta mudança não impactava os consumidores existentes e que estaria conforme com a política
SDORDER	Sem alterações. Duas releases existentes com a mesma assinatura devido a um erro de implantação no ambiente.	Analista analisou apenas uma <i>release</i> .
NOTESKEY	Exclusão de campos opcionais	Analista julgou que se tratava de uma variante ao invés de uma <i>release</i>
ESTREF	Otimização do código, sem alteração de interface.	Analista interpretou a mudança como nova release devido ao uso de ferramenta de identificação de diferenças no clearcase.
CALEND	Inclusão de novo campo opcional na interface de saída	Analista julgou que esta mudança não impactava os consumidores existentes e que estaria conforme com a política

É possível observar que ocorreram erros em três categorias: diferenciação entre variante e *release*; identificação do tipo de mudança que demandaria a criação e uma nova *release* e erros no procedimento de verificação.

Quanto a diferenciação entre variante e *release* (serviços BIGE e NOTESKEY), observou-se que os analistas não tinham clara a diferença entre os dois conceitos, o que os levou a classificar erroneamente as versões dos serviços.

Quanto ao conceito do que seria uma mudança necessária para geração de uma nova *release* (WFORCE, VENDOR, DIP, ESTREF e CALEND), este era outro ponto confuso para os analistas. Apesar de explícito na política que estava sendo considerada apenas a questão de interface de entrada e saída, os analistas responsáveis pela avaliação, todos com experiência em SOA em outras organizações, utilizaram conceitos que traziam de outras organizações ou da própria teoria, como impacto em consumidores, alterações de tecnologias e efeitos. Tal fato levou a novas discussões na equipe e foi sugerida uma evolução da política para considerar a questão de inclusão de campos opcionais no critério de variante e não de *release*, uma vez que esta operação não compromete o uso dos serviços por parte dos consumidores existentes.

Quanto ao procedimento de verificação (SDORDER), foi verificado que o analista tinha pouca experiência no ambiente de operação do barramento de serviços, não conseguindo identificar com precisão as duas releases disponíveis do serviço.

Em todos os casos, o uso do intelliGOV conseguiu identificar corretamente as conformidades e não conformidades através da comparação entre os campos de entrada e saída entre as *releases* distintas. É importante observar que caso se opte pelo tratamento de campos opcionais nas políticas, tal ponto pode ser tratado através da inclusão de um propriedade de dado ou de uma nova classe referente a tipos de elementos de dados na ontologia.

V.6 Política 6

Para a política 6, que verifica a existência de apenas uma variante de cada *release* ativa no ambiente produtivo, foram obtidos os resultados na verificação de conformidade apresentados na tab. A estrutura é a mesma da Tabela 17, apresentada na seção IV e as divergências são apresentadas em cinza.

Tabela 27. Resultados da verificação da política 6

Serviço	Avaliador	Avaliação Manual	Avaliação intelliGOV	Resultado Gabarito
BIGE	Analista 4	OK	OK	OK
BUPLA	Analista 4	OK	OK	OK
WFORCE	Analista 5	OK	X	X
COMPANY	Analista 5	X	X	X
VENDOR	Analista 5	OK	OK	OK
ORGUNI	Analista 5	X	OK	OK
DIP	Analista 6	OK	OK	OK
MENTOR	Analista 6	OK	OK	OK
VACPLN	Analista 6	OK	OK	OK
SDORDER	Analista 7	OK	OK	OK
REMCAR	Analista 7	OK	OK	OK
SDINVOIC	Analista 7	OK	OK	OK
NOTESKEY	Analista 7	X	OK	OK
CATAPL	Analista 7	OK	OK	OK
SALDO	Analista 8	OK	OK	OK
STOCKS	Analista 8	OK	OK	OK
REMBEN	Analista 8	OK	OK	OK
ESTREF	Analista 8	OK	OK	OK
PLATA	Analista 9	OK	OK	OK
SONDA	Analista 9	OK	OK	OK
BUNKER	Analista 9	OK	OK	OK
CONTA	Analista 9	OK	OK	OK
EMAIL	Analista 10	OK	OK	OK
CALEND	Analista 10	OK	OK	OK
VENPAY	Analista 10	OK	OK	OK

Foram identificadas divergências em quatro serviços: WFORCE, COMPANY, ORGUNI e NOTESKEY. Para verificar o motivo dos erros, foram levantadas as variantes ativas de cada serviço em produção. Os resultados são apresentados na Tabela 28.

Tabela 28. Variantes ativas em produção dos serviços divergentes para a política 6

Serviço	Variantes ativas
WFORCE	WFORCE_2.0, WFORCE_2.1, WFORCE_2.1.1, WFORCE_2.3
ORGUNI	ORGUNI_1.0, ORGUNI_2.1
NOTESKEY	NOTESKEY_1.0, NOTESKEY_2.0

Para o serviço WFORCE, foram geradas quatro variantes da release 2 e as quatro permaneceram ativas em produção. As variantes 2.0, 2.1 e 2.3 tem diferenças quanto a melhorias na implementação do serviço, porém foram mantidas ativas devido

ao receio do analista responsável de comprometer as variantes anteriores, uma vez que as variantes possuem elevado número de consumidores dependentes do serviço. A variante 2.1.1 foi gerada para tratar o problema de organização de esquemas da plataforma .NET, sem alteração na interface de entrada e saída, conforme visto na seção V.5. É importante observar que o analista, ao avaliar a política 6 considerou a versão como release, e ao avaliar a política 7, considerou como uma release. O uso de intelliGOV tratou este elemento como variante e não como *release*, levando aos resultados esperados.

Para o serviço ORGUNI, o analista atentou apenas para os números das variantes, sem considerar a release na análise, tratando ambas as variantes como originárias da mesma release. O uso de intelliGOV organizou corretamente as variantes em relação as releases.

Por fim, para o serviço NOTESKEY, como o analista julgou que as duas releases eram duas variantes e ambas se encontravam ativas em produção, ele considerou os serviços não conformes. O uso de intelliGOV tratou as duas versões como releases, assegurando o diagnóstico correto de conformidade.

V.7 Política 7

Para a política 7, que tratava da manutenção de releases ativas em produção com contratos ativos, foram obtidos os resultados apresentados na Tabela 29, com estrutura semelhante a utilizada pela Tabela 17, apresentada na seção IV. As divergências são destacadas na cor cinza.

Tabela 29. Resultados da verificação da política 7

Serviço	Avaliador	Avaliação Manual	Avaliação intelliGOV	Resultado Gabarito
BIGE	Analista 4	OK	X	X
BUPLA	Analista 4	OK	OK	OK
WFORCE	Analista 5	OK	OK	OK
COMPANY	Analista 5	OK	OK	OK
VENDOR	Analista 5	OK	X	X
ORGUNI	Analista 5	OK	OK	OK
DIP	Analista 6	X	OK	OK
MENTOR	Analista 6	OK	OK	OK
VACPLN	Analista 6	OK	OK	OK
SDORDER	Analista 7	OK	X	X
REMCAR	Analista 7	OK	OK	OK
SDINVOIC	Analista 7	OK	OK	OK
NOTESKEY	Analista 7	OK	OK	OK
CATAPL	Analista 7	OK	OK	OK
SALDO	Analista 8	OK	OK	OK
STOCKS	Analista 8	OK	OK	OK
REMBEN	Analista 8	OK	OK	OK
ESTREF	Analista 8	OK	X	X
PLATA	Analista 9	OK	OK	OK
SONDA	Analista 9	OK	OK	OK
BUNKER	Analista 9	OK	OK	OK
CONTA	Analista 9	OK	OK	OK
EMAIL	Analista 10	OK	OK	OK
CALEND	Analista 10	OK	X	X
VENPAY	Analista 10	OK	OK	OK

Foram identificadas divergências nos seguintes serviços: BIGE, VENDOR, DIP, SDORDER, ESTREF e CALEND. Para diagnosticar a origem dos problemas, os dados referentes às releases ativas e os seus contratos ativos (entre parênteses, após a *release*) são apresentados na Tabela 30.

Tabela 30. Releases ativas e seus contratos ativos

Serviço	Releases ativas em produção e contratos
BIGE	BIGE_1.1 (0), BIGE_2.0 (1)
VENDOR	VENDOR_1.0 (0), VENDOR_2.0 (1)
DIP	DIP_1.0 (1), DIP_2.0 (2)
SDORDER	SDORDER_1.0 (1), SDORDER_2.0 (0)
ESTREF	ESTREF_1.2 (0), ESTREF_2.1 (2)
CALEND	CALEND_1.0 (0), CALEND_2.0 (2)

No caso dos serviços BIGE, VENDOR, ESTREF e CALEND, o analista não considerou a migração realizada de consumidores das releases 1 para 2, registrada no repositório de serviços da organização. Deste modo, julgaram que existiam contratos ativos para ambas as releases do serviço, classificando-os erroneamente como conformes.

Para o serviço DIP, ocorreu uma peculiaridade. Foi identificado que o mesmo consumidor de serviços utiliza as duas *releases* do serviço, em módulos distintos gerados em fases diferentes do projeto da aplicação. O analista responsável pela avaliação, ao visualizar a mesma aplicação consumindo as duas *releases* no repositório, intuiu que esta havia realizado migração da versão e, como não existiam outros consumidores do serviço, classificou a *release* como não conforme, mesmo existindo um contrato ativo para a *release* 1.0.

Por fim, para o serviço SDORDER, o erro de implantação descrito na seção V.5 gerou duas *releases*, sendo a primeira com um contrato registrado e ativo e a segunda sem nenhum contrato. Como tal erro não gerou nova entrada no catálogo de serviços, o analista considerou apenas uma *release* em sua análise, classificando como conforme, no momento em que existia uma versão posterior, sem contratos ativos, ativada em produção.

Em todos os casos o uso da abordagem intelliGOV diagnosticou corretamente o estado de conformidade, tendo atuado neste caso como uma base integradora de informações e conceitos, tornando mais simples a avaliação.

Apêndice VI. Estimativa de esforço para implementação do intelliGOV

Neste capítulo é apresentada a técnica utilizada para estimar o esforço de implementação do intelliGOV, visando subsidiar uma análise dos custos da solução intelliGOV em comparação com avaliações manuais.

Por ser uma solução baseada em ontologias, regras, consultas e depender da implementação de mecanismos de carga, para estimar o custo foi necessário o uso de duas técnicas de estimativa: para ontologia, foi utilizada a técnica ONTOCOM (SIMPERL, TEMPICH e SURE, 2006); para as regras e mecanismos de carga, a técnica de pontos de função (GARMUS e HERRON, 2000). Nas próximas seções são apresentadas as técnicas e o cálculo de esforço.

VI.1 Estimativa da ontologia

Para estimar a ontologia, foi utilizada a técnica ONTOCOM, proposta por (SIMPERL, TEMPICH e SURE, 2006). Esta técnica considera que o esforço para implementação de uma ontologia é diretamente proporcional a quantidade de conceitos envolvidos e é multiplicada por fatores de ajuste que permitem equalizar o resultado com condições do ambiente no qual a ontologia será desenvolvida, como experiência da equipe, ferramentas disponíveis e abrangência da ontologia. A equação para cálculo é apresentada a seguir.

$$Esforço = A * Tamanho^{\alpha} * \prod FC_i$$

Onde:

- A é uma constante de ajuste;
- Tamanho é a quantidade de conceitos da ontologia, incluindo classes, propriedades e axiomas;

- α é um fator que permite ajustar o esforço de maneira não linear com o tamanho;
- FC_i representa fatores de custo, que permitem ajustar o resultado.

Os fatores de custo são obtidos através da complexidade de variáveis que podem contribuir para ampliar ou reduzir o esforço. São estas:

- Características do produto, que definem a complexidade da ontologia em questão, dividida nas variáveis complexidade do domínio, complexidade de implementação, necessidade de reuso, complexidade da conceitualização, complexidade das instâncias e necessidade de documentação;
- Características de reuso, que definem a influência do desenvolvimento para reuso no custo, considerando as variáveis de complexidade de avaliação da ontologia, dificuldade de tradução da ontologia, complexidade de alteração da ontologia e facilidade de compreensão da ontologia;
- Características da equipe, definindo a influência das pessoas que participarão do desenvolvimento, considerando as variáveis disponibilidade do especialista no domínio, continuidade da equipe envolvida ao longo do tempo, experiência do especialista de domínio e experiência nas linguagens utilizadas;
- Características de projeto, definindo a influência de condições do projeto no desenvolvimento, considerando como variáveis a disponibilidade de ferramentas e o qual o grau de interação entre os envolvidos no trabalho.

Cada uma das variáveis é classificada em níveis (Muito Baixa, Baixa, Nominal, Alta ou Muito Alta), que implicam em multiplicadores numéricos a serem utilizados para modular o valor do esforço.

Os valores de A , α e os multiplicadores FC_i são calibrados continuamente pela equipe responsável pelo método, disponibilizando a possibilidade de participar da calibragem através do site <http://ontocom.sti-innsbruck.at/index.htm>. Neste site, também se encontra disponível uma calculadora para estimativa utilizando o método.

Considerando o cenário desta dissertação, esta calculadora foi utilizada considerando um cenário corporativo. Para chegar aos parâmetros, foi considerado um crescimento do total de entidades trabalhada no estudo de caso (28, incluindo classes e propriedades) proporcional a razão entre os processos do commonGOV (51) e os processos tratados no estudo de caso (3). Com isso, alcançamos um universo de 476 entidades. Também foi considerado que a disponibilidade de analistas seria baixa, requisitos de documentação seriam altos e a experiência dos envolvidos em qualquer tecnologia ou método ligado a ontologias seria baixa. Com estas premissas, foram considerados os parâmetros apresentados na Tabela 31.

Tabela 31. Parâmetros para estimativa com ONTOCOM

Variável	Valores cenário corporativo
Tamanho	476
Produto/complexidade de domínio	Muito Alta
Produto/complexidade de implementação	Normal
Produto/necessidade de reuso	Normal
Produto/complexidade da conceitualização	Alta
Produto/complexidade das instâncias	Normal
Produto/necessidade de documentação	Muito Alta
Reuso/complexidade de avaliação	Normal
Reuso/complexidade de tradução	Normal
Reuso/complexidade de alteração	Normal
Reuso/facilidade de compreensão	Normal
Equipe/disponibilidade do especialista de domínio	Baixa
Equipe/continuidade da equipe	Alta
Equipe/experiência do especialista de domínio	Normal
Equipe/experiência na linguagem de modelagem	Muito Baixa
Projeto/ferramentas	Normal
Projeto/distribuição do trabalho	Alta
Resultado da Estimativa – Pessoas/mês	2,68 pessoas/mês

VI.2 Estimativa das cargas e das regras

Para estimar o esforço das regras e das cargas, foi utilizada a técnica de pontos de função (GARMUS e HERRON, 2000). Esta técnica consiste em isolar o sistema e contar fluxos de dados entre o usuário, meios de armazenamento e sistemas externos, considerando os seguintes fatores:

- Arquivos lógicos de entrada (ALI): representam quaisquer arquivos ou repositório de dados utilizados pela aplicação. Para cada arquivo, são

contados os diferentes tipos de registro (TR) e os diferentes atributos armazenados (TD);

- Arquivos de Interface Externa (AIE): representam quaisquer arquivos ou interfaces utilizadas para comunicação com outras aplicações. Para tal, devem ser contados os tipos de registros (TR) e os tipos de dados distintos (TD) considerados em cada registro enviado;
- Entradas Externas (EE): Quaisquer interações com o usuário que acarretem em entradas de dados no sistema. Também devem ser medidos os tipos de registro (TR) e os tipos de dados (TD);
- Saídas Externas (SE): Compreende todas as alterações que acarretam em alterações dos dados da aplicação. Consideram também o número de tipos de registro (TR) e tipos de dados (TD);
- Consultas Externas (CE): Consideram interações com o foco em consultas pontuais dos dados existentes no sistema, sem alterações de estado. Também devem ser considerados a quantidade de tipos de registro e a quantidade de tipos de dados distintos.

Para cada tipo de elemento contado, os valores de TR e TD implicam em um nível de complexidade do elemento. Um exemplo é apresentado na Tabela 32, considerando TDs e TRs para os elementos do tipo Entradas Externas.

Tabela 32. Exemplo de atribuição de complexidade em pontos de função

TR	TD		
	< 20	20-50	> 50
1	BAIXA	BAIXA	MEDIA
2-5	BAIXA	MEDIA	ALTA
> 5	MEDIA	ALTA	ALTA

Uma vez determinados os elementos e calculada a sua complexidade, estes são agrupados e contados. A quantidade de cada tipo de elemento e sua respectiva complexidade é multiplicada por um peso, e este total é somado, gerando uma contagem de pontos de função não ajustada. Um exemplo é apresentado na Tabela 33.

Tabela 33. Exemplo de contagem de pontos de função

Categorias De Funções		BAIXA		MEDIA		ALTA		Total
		Qtd	Peso	Qtd	Peso	Qtd	Peso	
Arquivos Lógicos Internos	ALI	3	7	1	10	0	15	31
Arquivos de Interface Externa	AIE	1	5	0	7	0	10	5
Entrada Externa	EE	3	3	0	4	1	6	15
Saída Externa	SE	0	4	2	5	1	7	17
Consulta Externa	CE	0	3	1	4	1	6	10
Total Não Ajustado								78

Neste exemplo, existem três ALIs de complexidade baixa, 1 ALI de complexidade média e nenhum de complexidade alta. O número de elementos de complexidade baixa é multiplicado por um peso, definido pelo método como 7 para este tipo de elemento, e o de complexidade média é multiplicado por peso de valor 10. Estes valores são somados, gerando o total de pontos de função gerados por Arquivos Lógicos Internos. De maneira semelhante, são calculados os pontos de função originários dos outros tipos de elemento. O total fornece uma contagem de pontos de função não ajustada para uma aplicação.

Para ajustar o valor, são empregados fatores de ajuste, que permitem modular o valor de acordo com fatores como complexidade de processamento, volume de transações e desempenho. Com base no valor final, este é convertido para esforço através da multiplicação da quantidade de pontos pela produtividade da equipe que irá desenvolver o sistema em uma determinada tecnologia.

No cenário do estudo de caso, a técnica foi utilizada para estimar o esforço da especificação das regras e das cargas necessárias para alimentar o intelliGOV. Como a base utilizada é a própria ontologia, estimada na seção anterior, esta não foi considerada nesta estimativa.

Para realizar esta estimativa cada regra foi considerada como uma consulta externa (CE) e foram consideradas duas consultas por processo do commonGOV, totalizando 102 consultas. Cada consulta considerou a média de 14 tipos de registro com 3 tipos de dado cada. Deste modo, obtemos sete elementos do tipo consulta externa de complexidade média, que multiplicados pelo peso deste tipo de elemento (4), equivalem a 408 pontos de função. Porém, como a solução demanda apenas a especificação da regra, não sendo necessária implementação adicional, deve ser considerada apenas esta etapa para fins de estimativa. Em estudo realizado por (YANG

et al., 2008), estima-se que em um ciclo de desenvolvimento em cascata, cerca de 16% do esforço é dedicado a etapa de especificação de requisitos. Considerando este fator, temos um total de 65,28 pontos de função.

Para as cargas, foram implementados dois módulos de carga, um trafegando quatro registros, o outro carregando três registros, conforme descrito na seção 4.4.2.8. Cada módulo foi considerado como um arquivo de interface externa (AIE), com 3 e 4 tipos de registro e três tipos de dados distintos, gerando dois elementos de complexidade baixa. Se considerarmos o cenário corporativo, foi considerado o mesmo fator de crescimento da quantidade de conceitos da ontologia, levando a 34 interfaces de complexidade baixa. Considerando o peso, temos mais 340 pontos de função. Com este número, alcançamos um total de 405,28 pontos de função não ajustados.

Para fins de ajuste, foram considerados os fatores apresentados na Tabela 34, gerando um multiplicador no valor de 0,94, totalizando 380,96 pontos de função. Se consideramos a produtividade de homem-hora por ponto de função na tecnologia Java da organização que participou no estudo de caso como 8 horas / ponto de função, temos um esforço total aproximado de 3050 horas para implementação das regras e mecanismos de carga.

Tabela 34. Cálculo do fator de ajuste para o cenário do estudo de caso

Característica Geral do Sistema	Nível de Influência	Peso
Comunicação de Dados	Influência Significativa	4
Processamento Distribuído	Nenhuma Influência	0
Performance	Influência Moderada	2
Configuração Altamente Utilizada	Nenhuma Influência	0
Volume de Transações	Influência Média	3
Entrada de dados On-Line	Influência Moderada	2
Eficiência do Usuário Final	Influência Moderada	2
Atualizações On-Line	Influência Mínima	1
Complexidade de Processamento	Influência Significativa	4
Reutilização	Influência Moderada	2
Facilidade de Instalação	Influência Moderada	2
Facilidade de Operação	Influência Significativa	4
Múltiplos Locais	Influência Mínima	1
Facilidade de Mudança	Influência Moderada	2
TDI (Somatório Níveis de Influência)		29
VAF - (TDI x 0,01) + 0,65)		0,94

VI.3 Estimativa Total

Considerando a estimativa da ontologia (2,68 pessoas/ mês, equivalentes a aproximadamente 430 horas) e a estimativa da implementação das cagas e das regras (3050 horas), temos um esforço total de 3480 horas para customização da solução intelliGOV. Vale ressaltar que, deste total, apenas 528 horas (em torno de 15,2%) correspondem a especificação das regras.

Apêndice VII. Função *validatePolicy*

Neste apêndice é apresentada a listagem do método descrito a seção 6.3, utilizado para verificar conformidade em Java na plataforma OEP.

```
public class PolicyJavaHandler {

    /**
     * @param args
     */

    private Connection con;

    public PolicyJavaHandler() {

        try{
            // Connects on Database
            Class.forName("oracle.jdbc.OracleDriver");
            String url =
                "jdbc:oracle:thin:@192.168.0.13:1521:XE";
            con = DriverManager.getConnection(url, "service",
                "service");
            System.out.println("Conectado");
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    public int validatePolicy1 ( HashSet<String> input,
                                HashSet<String> output,
                                String serviceName) {
        // TODO Auto-generated method stub

        HashSet<String> intData = new HashSet<String>();
        HashSet<String> saData = new HashSet<String>();
        HashSet<String> saServ = new HashSet<String>();

        // combines input and output in one collection
        intData.addAll(input);
        intData.addAll(output);

        try {

            Iterator<String> intIterator = intData.iterator();
```

```

// Obtain subject area list for data elements
while(intIterator.hasNext()) {
    String infName = (String) intIterator.next();
    String query = "SELECT SUB_NAME FROM
                    SUBJECT_AREA SA,
                    RESPONSIBLE_FOR RF,
                    INFORMATION_TYPE INF ";
    query = query + "WHERE SA.SUB_ID =
                    RF.RFOR_SUB_ID AND
                    RF.RFOR_INF_ID =
                    INF.INF_ID AND
                    INF.INF_NAME = '"";
    query = query + infName + "'";

    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery(query);

    while(rs.next()) {
        String subArea =
            rs.getString("SUB_NAME");
        saData.add(subArea);
    }
}

// Obtain Service Subject Area List
String queryServ ="SELECT SUB_NAME,SUB_ID
                    FROM SUBJECT_AREA SA,
                    HANDLES_DATA_FROM
                    HDF, ELEMENT E,
                    SERVICE S ";
queryServ = queryServ + "WHERE SA.SUB_ELEMENT_ID =
                    HDF.HDF_ELEMENT_FROM_ID
                    AND E.ELEMENT_ID =
                    HDF.HDF_ELEMENT AND ";
queryServ = queryServ + "E.ELEMENT_ID = S.ELMT_ID
                    AND S.SRV_NAME = '" +
                    serviceName + "'";

//System.out.println(queryServ);

Statement st = con.createStatement();
ResultSet rs = st.executeQuery(queryServ);

while (rs.next()) {
    // Checks for child subject areas
    String saName = rs.getString("SUB_NAME");
    int saID = rs.getInt("SUB_ID");
    saServ.add(saName);

    if (saName != null)
    {
        String childQuery = "SELECT SUB_NAME FROM
                            SUBJECT_AREA ";
        childQuery = childQuery + "START WITH
                            SUB_ID = " + saID + " ";
        childQuery = childQuery + "CONNECT BY
                            PRIOR SUB_ID = SUB_PARENT_ID ";
    }
}

```

```

        //System.out.println(childQuery);

        Statement chldStmt =
            con.createStatement();
        ResultSet chlsRes =
            hldStmt.executeQuery(childQuery);

        while (chlsRes.next()) {
            String saChildName =
                chlsRes.getString("SUB_NAME");
            //System.out.println(saChildName);
            saServ.add(saChildName);
        }
    }

    // Verify if all data subject areas are included in
    // service subject area
    boolean compliant = saServ.containsAll(saData);
    if (compliant) {
        return 1;
    } else {
        return 0;
    }

} catch (Exception e) {
    e.printStackTrace();
    return 0;
}

}

}

```