UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UMA ABORDAGEM BASEADA EM SIMILARIDADE SEMÂNTICA PARA PROMOVER BAIXO ACOPLAMENTO DE ESQUEMA DE DADOS ENTRE ASSINANTES E PUBLICADORES EM SOLUÇÕES PUB/SUB

Antônio Fonseca Pimenta Júnior

Orientadores

Leonardo Guerreiro Azevedo

Flávia Maria Santoro

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO de 2017

UMA ABORDAGEM BASEADA EM SIMILARIDADE SEMÂNTICA PARA PROMOVER BAIXO ACOPLAMENTO DE ESQUEMA DE DADOS ENTRE ASSINANTES E PUBLICADORES EM SOLUÇÕES PUB/SUB

Antônio Fonseca Pimenta Júnior

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 EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Leonardo Guerreiro Azevedo, D.Sc., UNIRIO

Flávia Maria Santoro, D.Sc., UNIRIO

Geraldo Zimbrão da Silva, D.Sc., UFRJ

Sean Wolfgand Matsui Siqueira, D.Sc., UNIRIO

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO de 2017

Catalogação informatizada pelo(a) autor(a)

PP644

PIMENTA JÚNIOR, ANTÔNIO FONSECA
UMA ABORDAGEM BASEADA EM SIMILARIDADE SEMÂNTICA PARA PROMOVER BAIXO ACOPLAMENTO DE ESQUEMA DE DADOS ENTRE ASSINANTES E PUBLICADORES EM SOLUÇÕES PUB/SUB / ANTÔNIO FONSECA PIMENTA JÚNIOR. -- Rio de Janeiro, 2017.
84

Orientador: Leonardo Guerreiro Azevedo.
Coorientador: Flávia Maria Santoro.
Dissertação (Mestrado) - Universidade Federal do Estado do Rio de Janeiro, Programa de Pós-Graduação em Informática, 2017.

1. Integração de Aplicações Corporativas. 2. Arquitetura Orientada a Serviços. I. Guerreiro Azevedo, Leonardo, orient. II. Maria Santoro, Flávia, coorient. III. Título.

Àqueles que sempre me dão força para seguir em frente em todos os projetos da minha vida. Este trabalho dedico aos meus pais.

Agradecimentos

O agradecimento é um momento especial em que você percebe quantas pessoas queridas tem em sua vida. Sou muito grato aos amigos e familiares que me incentivaram nesta longa jornada do mestrado.

Agradeço aos meus pais e irmãos pelo apoio incondicional que sempre deram em todos os projetos de minha vida. Não tenho palavras para agradecer à minha noiva, Priscila, pelo carinho e compreensão nos momentos mais difíceis do mestrado.

Devo agradecer imensamente aos meus orientadores, Leonardo Azevedo e Flávia Santoro, pela orientação, dedicação, paciência e toda experiência compartilhada.

Por fim, agradeço aos amigos da Dataprev que acompanharam de perto esta dura jornada e me incentivaram bastante: Vinícius Lopes, Marco Ambrozio, Wagner Amaral, Renata Xavier, Victor Macedo, Cláudio Libanio e todos os outros.

PIMENTA JÚNIOR, ANTÔNIO FONSECA. **Uma abordagem baseada em similaridade semântica para promover baixo acoplamento de esquema de dados entre assinantes e publicadores em soluções Pub/Sub**. UNIRIO, 2017. 84 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

A integração e troca de informações entre sistemas é um problema desafiador em arquitetura de software. A heterogeneidade e a volatilidade das fontes de informação são inerentes ao cenário de sistemas de diferentes fornecedores, construídos sobre diferentes filosofias e prioridades. *Publish/Subscribe* é um padrão de integração que pretende resolver esses problemas promovendo baixo acoplamento entre os sistemas. Este padrão permite que os assinantes expressem através de assinaturas quais eventos desejam receber. As assinaturas baseadas no conteúdo dos eventos se destacam por promover maior flexibilidade e expressividade. Porém, apresentam a desvantagem de exigir que os sistemas participantes acordem quais serão as classes, tipos, propriedades e valores utilizados nas notificações de eventos e nas restrições das assinaturas. O que gera um acoplamento de estrutura de dados entre publicadores e assinantes.

Este trabalho apresenta um modelo arquitetural para integração de sistemas corporativos, baseado no padrão *Publish/Subscribe*, que permite a redução do acoplamento de modelo de dados entre assinantes e publicadores. O modelo proposto permite que o servidor de eventos, através da avaliação de similaridade semântica entre os nomes dos atributos das notificações e assinaturas, seja capaz de determinar se um evento deve ser entregue a um determinado assinante. Nesta proposta, o servidor de eventos é formado por componentes plugáveis que podem ser substituídos por outros que implementem diferentes métodos e estratégias de avaliação de similaridade, a depender do cenário de aplicação do modelo arquitetural.

A viabilidade da arquitetura proposta foi demonstrada através de uma solução de referência. Uma variedade de diferentes métodos de avaliação de similaridade semântica foram plugados à solução e experimentados diante de um *Dataset* representando um cenário real. Os resultados obtidos mostraram que a solução é capaz de determinar com eficácia os assinantes interessados nos eventos publicados.

Palavras-chave: Event-Driven Architecture, Semantic Publish/Subscribe.

ABSTRACT

Integration and exchange of information between systems is a challenging problem in software architecture. Heterogeneity and volatility of information sources are inherent to the scenario of systems of different suppliers, built on different philosophies and priorities. Publish/Subscribe (Pub/Sub) is a system integration pattern that intends to solve these problems by promoting low coupling between systems. This pattern allows subscribers to express which events they wish to receive. The subscriptions based on the content of the events stand out for promoting greater flexibility and expressiveness. However, they have the disadvantage of requiring participating systems to agree on the classes, types, properties, and values used in the event notifications and subscriptions. This generates a data structure coupling between publishers and subscribers.

This work presents an architectural model for systems integration based on Publish/-Subscribe, which allows the reduction of data model coupling between subscribers and publishers. The proposed model allows the event server to be able to determine if an event should be delivered to a given subscriber, through the evaluation of semantic similarity between the names of the attributes of notifications and subscriptions. In this proposal, the event server is composed of pluggable components that can be replaced by others that implement different methods and strategies of similarity evaluation, depending on the scenario.

The viability of the proposed architecture was demonstrated through a reference solution. The results of the experiments with different methods of semantic similarity evaluating and a dataset representing a real scenario, showed that the solution is able to effectively determine subscribers interested in the published events.

Keywords: Event-Driven Architecture, Semantic Publish/Subscribe.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Nomenclaturas

ATOM	Atom Publishing Protocol
EDA	Event-Driven Architecture
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
RDF	Resource Description Framework
RSS	Really Simple Syndication
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
XML	eXtensible Markup Language
XPath	XML Path Language

1. Introdução

Este capítulo apresenta uma visão geral desta pesquisa, incluindo o contexto, o problema tratado, o objetivo da solução proposta, a metodologia de pesquisa científica aplicada, bem como a estrutura do trabalho.

1.1 Motivação e Caracterização do Problema

Eugster [7] define que, no âmbito da integração de sistemas e troca de informações, obter desacoplamento entre sistemas significa remover toda e qualquer dependência explicita entre os sistemas participantes. As soluções orientadas a eventos que utilizam o padrão *Publish/Subscribe* promovem três níveis de desacoplamento: Desacoplamento de Espaço, Tempo e Sincronização.

Além disso, geralmente, nas trocas de informações entre sistemas, os consumidores estão interessados em receber apenas informações específicas. Nesse contexto, uma grande vantagem dos sistemas implementados sob o padrão *Pub/Sub* é a possibilidade dos consumidores, chamados de assinantes, expressarem através de filtros quais eventos, informação originada no sistema produtor, são de seu interesse. Dentre os ganhos dessa abordagem destaca-se a redução do tráfego desnecessário de informações, uma vez que só ocorre transmissão de dados quando um novo evento de interesse do assinante é notificado por um publicador.

Em soluções *Publish/Subscribe* existem diferentes tipos de assinaturas possíveis. Dentre as possibilidades, as implementações que utilizam assinaturas baseadas no conteúdo se destacam por promoverem maior flexibilidade em relação à forma como os assinantes

especificam o tipo de evento que desejam receber.

Os filtros são construídos com base nas estruturas dos dados contidos na notificação dos eventos, levando em consideração os tipos e propriedades que encapsulam os valores. Por exemplo, um sistema publicador que notifica eventos de *Venda na Internet* encapsula os dados na classe *Venda* que contém o nome do *produto* vendido, o *preço*, a data de *ocorrência* da venda, o nome do *comprador* e o *endereço* de entrega. A classe *Venda* poderia ter uma estrutura como essa: *{**String** produto; **Double** preço; **Date** ocorrência; **String** comprador; **String** endereço}*. E uma instância possível da classe *Venda* seria:

Exemplo 1.1 (Evento de venda na Internet)**.** *:*
*{**produto**: Macbook Air,*
***preço**: R$ 6000.00,*
***ocorrência**: 2017-06-15,*
***comprador**: João da Silva,*
***endereço**: Avenida Rio Branco, 1000. Rio de Janeiro - RJ }*

Por outro lado, para filtrar tais eventos, os sistemas consumidores deveriam criar seus filtros obedecendo os nomes dos atributos acordados e os tipos de dados definidos previamente. Um sistema consumidor que queira ser notificado da ocorrência de vendas posteriores ao primeiro dia de 2017 entregues no Rio de Janeiro deveria criar uma assinatura semelhante a esta: *{ocorrência > 2017-01-01; endereço = Rio de Janeiro - RJ}*, nesse caso o filtro seria feito sobre o campo *ocorrência* e *endereço*.

Contudo, como exposto em Hasan *et al.* [11], se os sistemas participantes precisam acordar previamente a estrutura das notificações dos eventos (tipos, propriedades e valores) configura-se então uma explicita dependência entre as duas partes. Tal acoplamento limita a escalabilidade da solução. Outro problema é o alto custo de definir e manter as assinaturas em situações que o assinante deseja receber notificações de eventos que podem ser originados de diferentes publicadores.

Seguindo o exemplo anterior, a Figura 1.1 ilustra um cenário em que dois assinantes criam suas assinaturas para receber notificações do sistema de vendas. O *Assinante A* deseja ser notificado quando ocorrerem vendas do produto "livro"com preço maior que "50". Já o *Assinante B* criou a assinatura utilizando a sentença "preço de venda"para especificar

o filtro sobre o preço do produto. Neste cenário, apesar do *Assinante B* aparentemente estar interessado no mesmo tipo de notificações que o *Assinante A*, apenas o *Assinante A* receberá a *Notificação 1*, pois somente o seu filtro está de acordo com o esquema das notificações do *Publicador*.

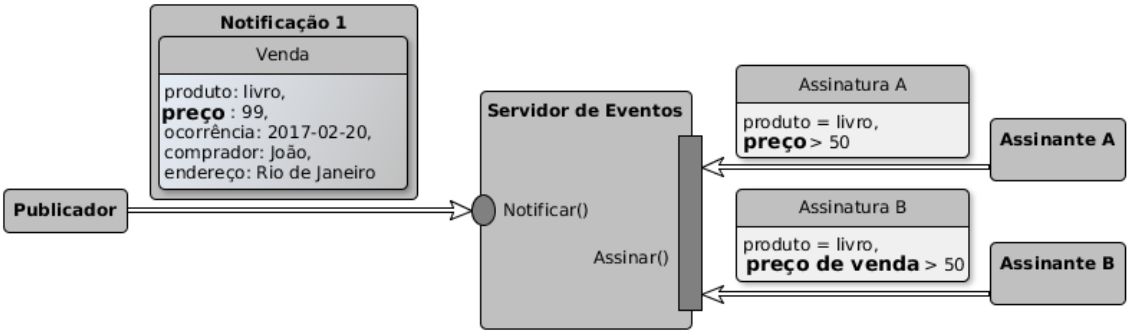

Figura 1.1: Notificação e Assinaturas

Por fim, pode ser afirmado que o acoplamento de estrutura de dados nestes cenários ocorre quando os sistemas participantes, publicadores e assinantes, precisam acordar as classes, tipos, propriedades e valores para as notificações de eventos e, consequentemente, para as assinaturas. Esse acoplamento se torna um problema mais grave em cenários onde ocorre maior heterogeneidade em relação aos dados que precisam ser trafegados, onde, muitas vezes, os sistemas participantes são de fornecedores diferentes ou construídos sobre diferentes paradigmas e prioridades.

1.2 Questão de pesquisa e hipótese

Este trabalho investiga a seguinte questão de pesquisa: "Como garantir o desacoplamento de estrutura de dados das notificações de eventos em situações em que ocorre heterogeneidade entre os sistemas produtores e o consumidores?".

A hipótese da pesquisa é: "Uma arquitetura Publish/Subscribe, onde o servidor de eventos utilize métodos de similaridade semântica como base para a tomada de decisão de definir se um evento deve ser notificado ou não a um assinante, pode garantir o desacoplamento de estrutura de dados mesmo em situações onde ocorra heterogeneidade".

1.3 Objetivo da Pesquisa

O objetivo geral deste trabalho é apresentar uma proposta arquitetural que possibilite a redução do acoplamento de estrutura de dados entre publicadores e assinantes.

Para alcançar este objetivo, a abordagem proposta utiliza métodos de avaliação de similaridade semântica para apoiar a tomada de decisão de definir se um evento deve ser notificado ou não a um assinante.

1.4 Metodologia de Pesquisa

O método de pesquisa aplicado neste trabalho foi o quantitativo. Este método é centrado na coleta de dados com o objetivo de avaliar o estado de alguma variável de um determinado domínio no mundo real. O método aplicado segue o processo proposto por Recker [22], o qual compreende as atividades de: (i) geração da teoria e hipótese; (ii) desenvolvimento de instrumentos para medição; (iii) coleta de dados; e (iv) análise dos dados e avaliação dos resultados.

Para permitir a avaliação da proposta, foi implementada uma solução de referência para o modelo arquitetural proposto. Esta solução foi submetida a experimentos simulando cenários reais. Foram coletados dados sobre sua eficácia em relação a sua capacidade de inferir se um evento deve ser entregue a um assinante de acordo com a especificação do conteúdo do evento e do interesse do assinante. Nesta análise foram avaliadas *Precisão*, *Cobertura* e *Medida-F*.

Além da avaliação de eficácia, também foram utilizadas métricas para avaliar o desempenho da solução. Para isso foi verificada a *Taxa de Notificação* (*Throughput*)[1] e o *Tempo de Notificação*[2].

[1] Número de notificações processadas por segundo.
[2] Tempo médio de processamento das notificações.

1.5 Organização da Dissertação

Para uma melhor compreensão do leitor, esta dissertação está divida em 5 capítulos incluindo esta introdução. A fundamentação teórica necessária à compreensão dos aspectos tratados neste trabalho é apresentada no capítulo 2. No capítulo 3 é apresentada a abordagem baseada em semântica para arquiteturas *Publish/Subscribe* proposta por este trabalho. Esta abordagem é avaliada através de um experimento, onde o planejamento e resultados são apresentados no capítulo 4. Finalizando, no capítulo 5 são apresentadas as conclusões, limitações e os trabalhos futuros.

2. Fundamentação Teórica

Neste capítulo será apresentada a fundamentação teórica necessária para uma melhor compreensão do presente trabalho. Serão apresentados os principais conceitos relacionados a arquiteturas orientadas a eventos, sistemas *Publish*/*Subscribe*, medidas de similaridade semântica e *Wordnet*.

2.1 Arquiteturas Orientadas a Eventos

Em Engenharia de Software, um *Evento* é usado para denominar um acontecimento que provoca mudança em alguma informação relevante para o contexto do negócio. Eventos podem ocorrer, por exemplo, devido à interação de um usuário com um sistema ou a algum processamento lógico que produziu ou alterou alguma informação [3]. Como exemplo, podem ser citados o cadastro de uma nova informação, a ocorrência de uma compra com cartão de crédito, um saque, um sinal de um sensor, ou o atingimento de uma cotação de moeda.

Dizer que um animal, pessoa, ou companhia é orientada a eventos significa dizer que este age em resposta ao acontecimento de eventos. Da mesma forma, um sistema orientado a eventos segue este princípio, e sobre eles dizemos que implementam o paradigma de orientação a eventos.

Os *softwares* não podem lidar diretamente com os eventos do mundo real, por isso nesse paradigma os eventos são abstraídos na forma de *Objetos Evento*. Um objeto evento é a abstração dos dados do evento em formato que o computador pode entender, geralmente documentos XML, JSON, e outros. Quando um objeto evento é transmitido entre um

software e outro em uma mensagem, recebe o nome de notificação [3, 17].

O componente de *software* sensível à ocorrência dos eventos de negócio e responsável por notificar os demais componentes é chamado de produtor de evento ou *producer*. O componente que recebe as notificações e reage a elas é chamado de consumidor de eventos ou *consumer*.

Arquitetura orientada a eventos (*Event-driven Architecture* ou EDA) é o estilo arquitetural que busca, baseado no paradigma de orientação a eventos, permitir a integração de sistemas através da disseminação de informações. Segundo Chandy *et al.* [3], existem cinco princípios fundamentais que um sistema EDA deve seguir:

- **Individualidade**: cada evento deve ser transmitido individualmente no momento da sua ocorrência. O produtor de eventos não deve acumular a ocorrência de vários eventos antes de começar a enviá-los.
- ***Push***: as notificações são enviadas pelo produtor, e não buscadas pelo consumidor. O que significa que o produtor decide qual o melhor momento para enviar a notificação. Este é um princípio fundamental, pois o consumidor não sabe quando os eventos ocorrem, e evita que ele precise ficar buscando novas notificações de tempos em tempos sem saber se elas de fato ocorreram.
- **Imediatismo**: o consumidor de eventos deve agir/responder imediatamente depois de receber uma notificação.
- **Mão única**: após notificar o evento, o produtor terminou seu trabalho na interação; portanto, ele não precisa aguardar resposta do consumidor.
- **Livre de comandos**: uma notificação não deve ser um comando, ou seja, ela não deve prescrever o que o consumidor da notificação irá fazer. O consumidor deve implementar a lógica para tratar a notificação.

2.2 Publish/Subscribe

No contexto da arquitetura de softwares e da integração de sistemas, *Publish/Subscribe*, ou simplesmente *Pub/Sub*, é um padrão, baseado nos princípios de *EDA*, de troca

de mensagens que propõe baixo acoplamento e eficiência no quesito tráfego de informações. Ao contrário das propostas de integração mais tradicionais, em que a comunicação é ponto-a-ponto e síncrona, com *Pub/Sub* os principais objetivos são garantir a flexibilidade de comunicação e o baixo acoplamento entre os participantes, o que representa uma vantagem em relação às propostas tradicionais [3, 17, 26].

A Figura 2.1 apresenta um comparativo entre as propostas de integração tradicionais e as baseadas em *Publish/Subscribe*. Nas propostas tradicionais (*Request-Response*), a disseminação de informação ocorre somente após a solicitação dos sistemas consumidores (à direita) por novas informações do sistema produtor (à esquerda). A cada nova solicitação é necessário verificar se ocorreram atualizações ou se novas informações estão disponíveis no produtor. Nas propostas *Publish/Subscribe* ocorre uma mudança de paradigma, as informações são disseminadas (notificadas) no momento em que elas são geradas ou atualizadas, de maneira que o sistema consumidor sempre terá disponível as informações mais atualizadas, sem a necessidade de checar constantemente se ocorreram modificações. Essa abordagem promove maior eficiência no trafego de dados e desacoplamento entre consumidor e provedor, que se quer se conhecem.

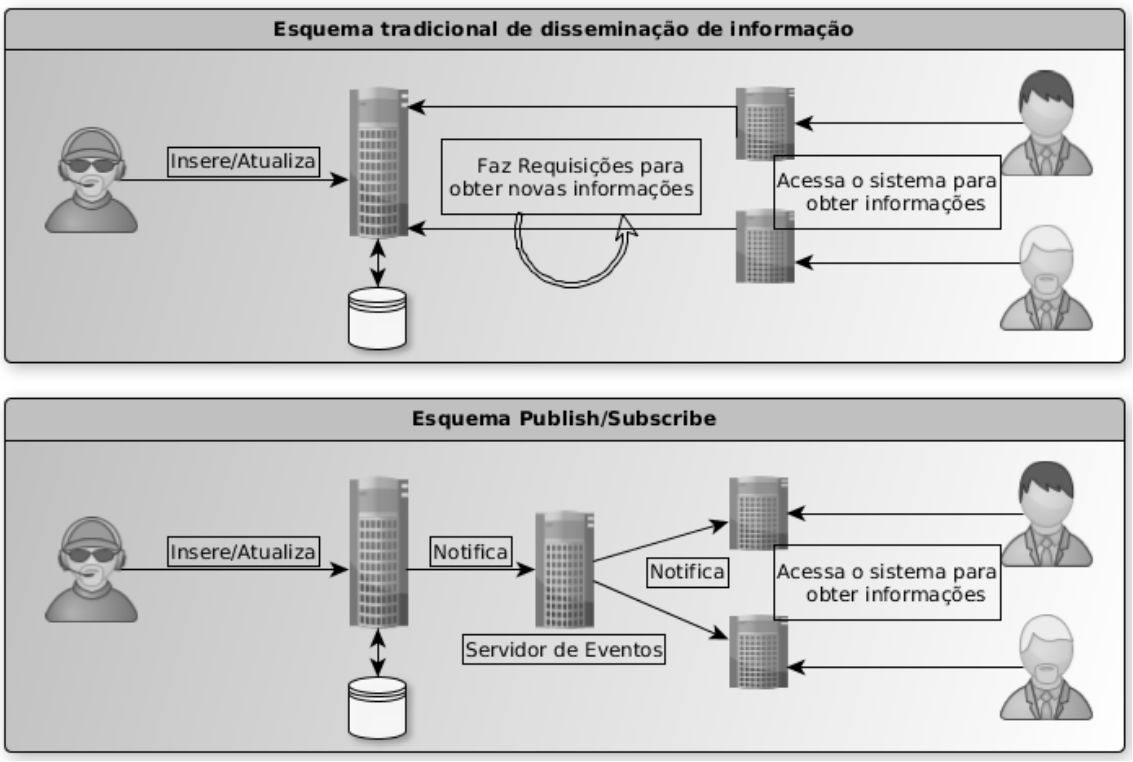

Figura 2.1: Comparação entre o modelo tradicional e *Publish/Subscribe*

Outras vantagens de *Pub/Sub* são relacionadas aos princípios de *EDA* na Figura 2.2.

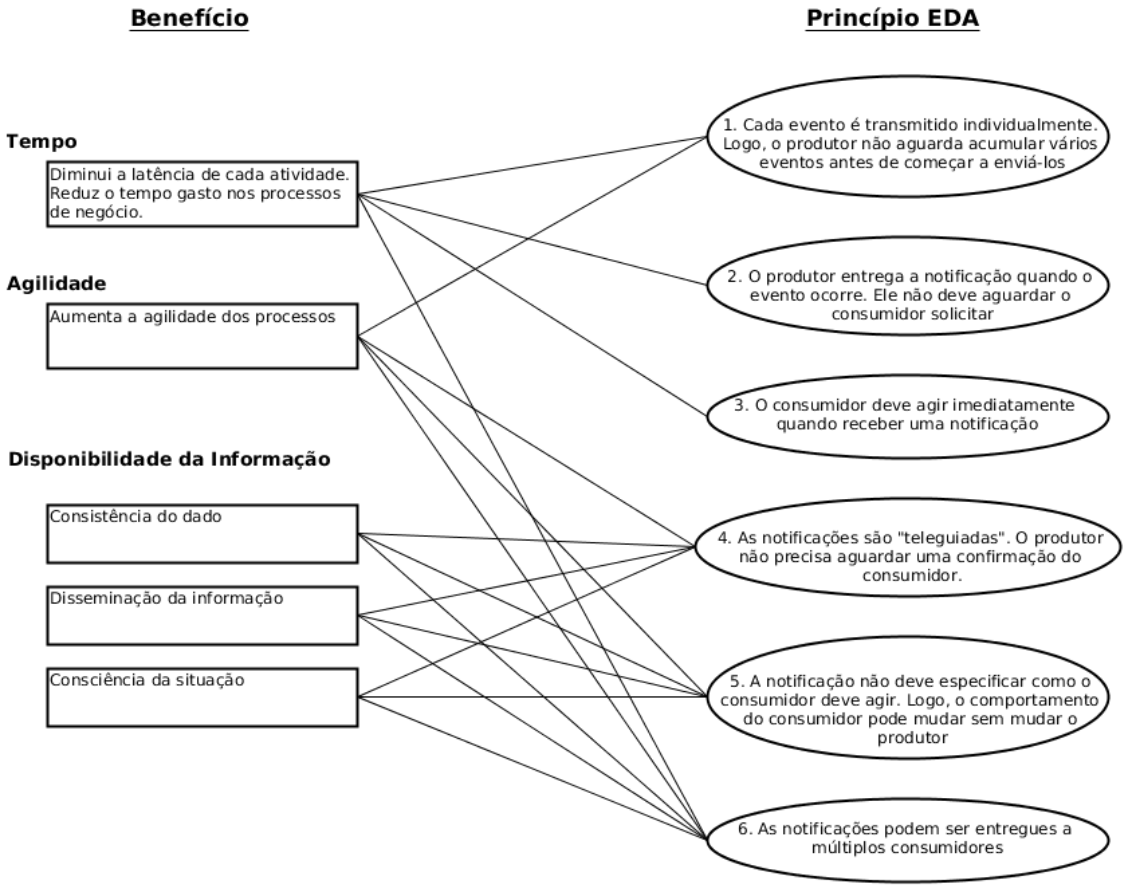

Figura 2.2: Vantagens de *Pub/Sub* e princípios de *EDA* (adaptado de Chandy *et al.* [3])

A Figura 2.3 ilustra o funcionamento básico das interações dos participantes de uma solução de integração construída com o padrão *Pub/Sub*. Os sistemas *Publishers* ou Publicadores são os produtores de eventos, ou seja, aqueles em que ocorre algum processamento lógico que produz ou altera alguma informação e que a disponibiliza como uma notificação de eventos nesse padrão. Os sistemas *Subscribers* ou Assinantes são os consumidores de eventos que, ao assinar um padrão de notificações, informam estar interessados em ser notificados da ocorrência de um evento no contexto do negócio. O padrão *Pub/Sub* inclui ainda um terceiro ator, o Servidor de Eventos, o qual é um mediador da comunicação entre os publicadores e os assinantes. Ele é o responsável por receber as notificações dos publicadores, manter as assinaturas dos assinantes, armazenar temporariamente as notificações de eventos e os rotear para entregar aos assinantes.

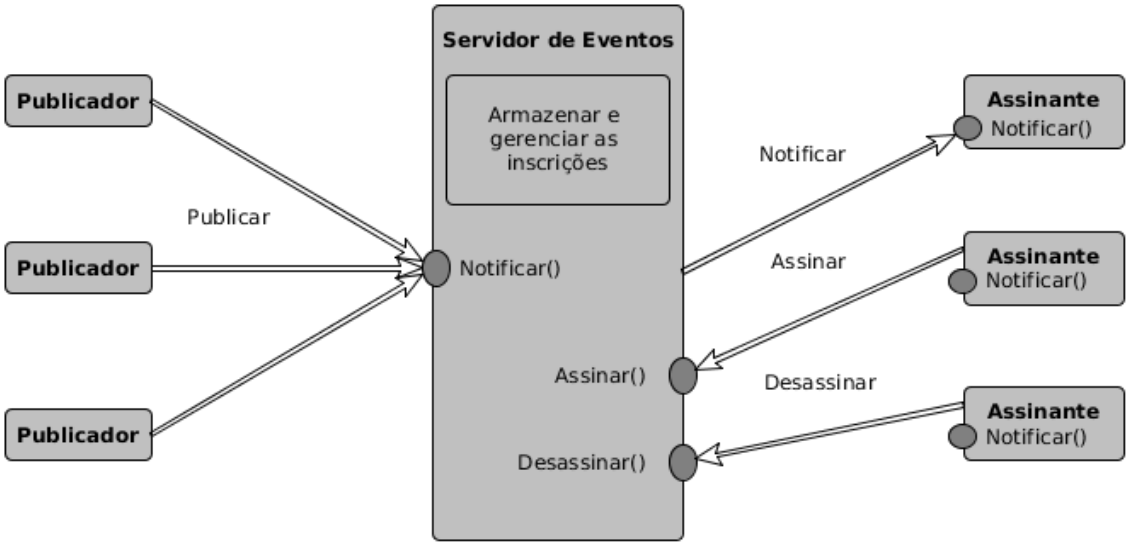

Figura 2.3: Interações do padrão *Pub/Sub* (adaptado de Eugster *et al.* [7])

2.2.1 Desacoplamento

Como já foi mencionado, o desacoplamento é uma grande vantagem do padrão *Pub/-Sub*. Ele é uma propriedade desejável porque permite escalabilidade, possibilitando que os participantes operem independentemente uns dos outros [7]. *Pub/Sub* permite três tipos de desacoplamento entre produtores e consumidores, são eles (Figura 2.4):

- **Desacoplamento de Espaço**: As partes que interagem não precisam se conhecer. Toda a comunicação fica a cargo do intermediador (*i.e* o servidor de eventos). Os publicadores publicam eventos, através do servidor de eventos, e os assinantes obtêm esses eventos indiretamente, através do servidor de eventos. Os publicadores não mantêm referências aos assinantes, nem sabem como os assinantes estão participando da interação. Da mesma forma, os assinantes não conhecem os publicadores, nem sabem quantos desses publicadores estão participando da interação. Além disso, um evento de um publicador pode ter como destino mais de um assinante.

- **Desacoplamento de Tempo**: As partes que interagem não precisam estar participando ativamente da interação ao mesmo tempo. O publicador pode publicar eventos enquanto o assinante está desconectado. Da mesma forma que o assinante pode ser notificado sobre a ocorrência de algum evento enquanto o publicador original está desconectado.

- **Desacoplamento de Sincronização**: Os publicadores não são bloqueados durante a produção de eventos e os assinantes podem receber notificação assíncrona (por meio de uma chamada de retorno) da ocorrência de uma atividade concorrente simultânea. A produção e o consumo de eventos não ocorrem no fluxo principal de controle dos publicadores e assinantes, e portanto não ocorrem de maneira síncrona.

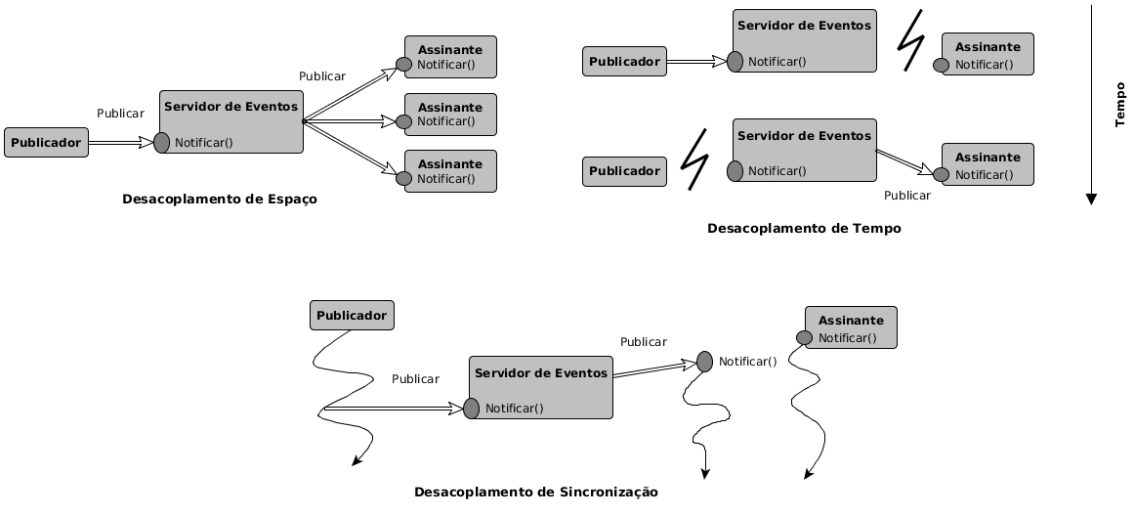

Figura 2.4: Desacoplamento (adaptado de Eugster *et al.* [7])

2.2.2 Tipos de assinatura

Os assinantes geralmente estão interessados em eventos ou padrões de eventos específicos e não em todos os eventos [7]. O padrão *Pub/Sub* permite que o assinante expresse quais eventos são de seu interesse através da sua assinatura. Existem três formas de assinatura:

- **Assinatura baseada em tópicos ou canais**: Os participantes podem publicar eventos e assinar tópicos individuais, que são identificados por palavras-chave. Os consumidores assinam as publicações que trafegam no canal específico do seu tópico de interesse -- funcionamento similar ao dos padrões RSS [1] e ATOM [2].

- **Assinaturas baseadas no conteúdo dos eventos**: esta alternativa é mais dinâmica e expressiva do que a assinatura baseada em tópicos ou canais, pois não se baseia

[1] **RSS** (*Really Simple Syndication*) permite aos usuários se inscrevem em sites que fornecem atualizações RSS e, então, passar a receber seu conteúdo sem precisar visitá-los um a um.

[2] O protocolo **ATOM** (*Atom Publishing Protocol*) foi desenvolvido como alternativa ao RSS e também permite que usuários recebam atualizações de conteúdos de seu interesse.

em um critério externo e predefinido como o nome de um tópico. Ela se baseia nas propriedades dos próprios eventos, que podem ser atributos internos das estruturas de dados que transportam eventos ou metadados associados aos eventos. Os consumidores se inscrevem em eventos especificando filtros usando uma linguagem de subscrição. As linguagens de subscrição mais comuns são baseadas na utilização de ``*strings*'' nos padrões SQL, XPath, SPARQL[3] ou outros padrões proprietários [17, 29].

- **Assinaturas baseadas no tipo**: este tipo de assinatura se baseia no tipo de dado do evento, ou seja, na sua estrutura externa ou classe que encapsula os dados do evento. O assinante deve informar o tipo de classe de evento que deseja receber notificações. Por isso é fortemente dependente de linguagem de programação e, apesar de garantir segurança na conversão dos dados (*Type safety*), exige que os componentes envolvidos na comunicação utilizem a mesma linguagem.

2.3 WordNet

A *WordNet* pode ser definida como uma base de dados léxica compreensível por computadores, organizada como uma taxonomia de conceitos. Este banco de dados liga substantivos, verbos, adjetivos e advérbios da língua inglesa a conjuntos de sinônimos que, por sua vez, são ligados através de relações semânticas que determinam as definições de palavras [18].

Embora a *WordNet* leve em consideração a morfologia das palavras, ela tenta priorizar os significados. Por basear-se em teorias psicolinguísticas para definir os significados das palavras, leva em consideração não somente as associações entre as palavras mas também as associações entre significados [15].

A *WordNet* agrupa as palavras em conjuntos de sinônimos chamados de *synsets*, que proveem curtas definições e guardam as várias relações semânticas destes conjuntos de sinônimos. Os *synsets* são interligados por meio de relações conceituais-semânticas e léxicas de maneira hierárquica. Os conceitos são relacionados a outros conceitos mais altos

[3]**SPARQL** é uma linguagem de consultas para documentos em RDF. RDF é um formato de dados em grafo rotulado e direcionado empregado para representar documentos na Web (`https://www.w3.org/TR/rdf-sparql-query/`).

ou baixos na hierarquia através de diferentes tipos de relacionamentos em que os mais comuns são os de generalização/especialização (*Hypernym/Hyponym*) e todo/parte de (*Meronym/Holonym*). Dessa forma, na *WordNet* qualquer palavra pode ser definida em termos de outras palavras a ela relacionadas na hierarquia.

A Figura 2.5 ilustra um exemplo de relacionamento de generalização/especialização para os conceitos *cat* (c) e *dog* (c'). Neste exemplo, o conceito comum mais específico considerando estes dois conceitos é *carnivore* (c'').

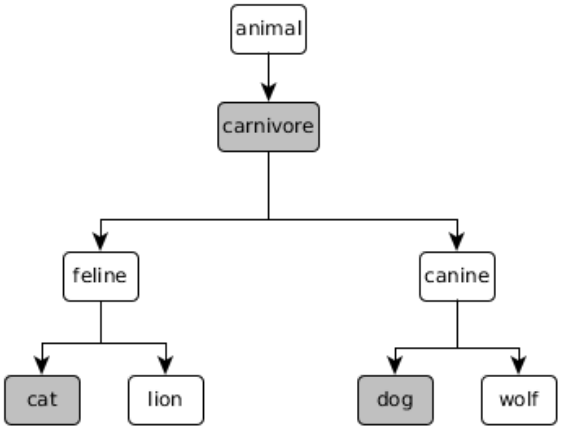

Figura 2.5: Exemplo de relacionamento de generalização/especialização na *Wordnet*

Além da *WordNet*, existem outras bases de conhecimento linguístico em outras línguas. Em português, podemos citar UfesWN.BR [9], CONTO.PT [10] e OpenWN-PT [5].

2.4 Medidas de similaridade semântica entre termos

Similaridade semântica, muitas vezes chamada apenas de similaridade, pode ser aplicada ao nível de palavra, sentença ou texto. As medidas de similaridade semântica entre palavras são usadas para detectar o nível de relacionamento entre os conceitos das palavras ou termos que tenham características em comum. Elas são capazes de determinar a similaridade de termos que muitas vezes não são lexicalmente similares. Para isto, estas medidas exploram os relacionamentos linguísticos entre conceitos utilizando recursos externos, geralmente ontologias como a *WordNet* [27] [15] [20].

Várias abordagens para determinar as similaridades semânticas foram propostas nas últimas décadas. Elas podem ser classificadas em três tipos de estratégias [15]:

- **Baseadas em arestas (*edge-based*)**: a similaridade entre dois conceitos é determinada pela distância (caminho) entre os conceitos e a posição do conceito na taxonomia;
- **Baseadas em informação (*information-based*)**: a similaridade é determinada considerando a quantidade de informação existente entre os conceitos em função das suas probabilidades de ocorrência em um Corpus [4].
- **Híbridas**: a similaridade é determinada combinando as duas abordagens anteriores.

Na construção da solução proposta por este trabalho foram utilizadas quatro diferentes abordagens que implementam os três tipos de estratégias citados acima. Essas abordagens foram utilizadas para permitir uma análise de viabilidade da proposta. As abordagens utilizadas são apresentadas nas próximas Seções.

2.4.1 Proposta de Wu & Palmer

Wu e Palmer [28] propuseram uma abordagem para determinar a similaridade semântica baseada em arestas. De maneira que, dados dois conceitos *c1* e *c2*, a similaridade semântica *s(c1, c2)* é dada em função das suas profundidades na taxonomia e da profundidade de seu conceito mais específico *c3*. Entende-se como profundidade a quantidade de nós entre um conceito e outro, como ilustra a Figura 2.6. Se *N1* e *N2* são as profundidades de *c1* e *c2*, e *N3* a profundidade de seu conceito mais específico, *c3*, na *WordNet*, a similaridade semântica entre *c1* e *c2* pode ser definida como:

$$sim_{wup}(c1, c2) = \frac{2 * N3}{N1 + N2 + (2 * N3)} \tag{2.1}$$

2.4.2 Propostas de Lin e Resnik

Os trabalhos de Lin [14] e Resnik [23] propõem medidas de similaridade baseadas em informação. Elas se baseiam nos links hierárquicos entre conceitos em uma ontologia e em um Corpus. A similaridade leva em conta as informações compartilhadas entre os dois

[4] **Corpus** linguístico é um conjunto de textos em uma determinada língua usado como base para análises linguísticas.

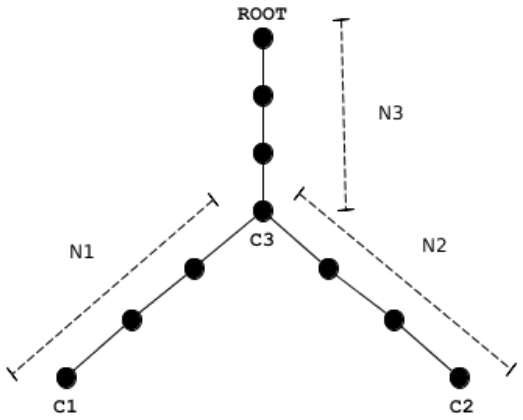

Figura 2.6: Medida de similaridade entre conceitos (Adaptado de Wu e Palmer [28])

conceitos avaliados, de maneira que a similaridade é determinada a partir da quantidade de informação (IC - *Information Content*) necessária para indicar o quão comum são os dois conceitos.

A medida de Resnik é calculada com a seguinte expressão:

$$\mathrm{sim}_{\mathrm{resnik}}(c1, c2) = \max_{c \in S(c1,c2)}[-\log(p(c)] \tag{2.2}$$

onde *p(c)* é a probabilidade de ocorrência do conceito *c* em um Corpus e *S(c1, c2)* é o subconjunto de conceitos subordinados a ambos os conceitos *c1* e *c2*.

A medida de Lin, por sua vez, possui os mesmos componentes da medida de Resnik, porem considera a proporção das probabilidades dos conceitos e por isso, segundo avaliação de Slimani [27], obtêm melhores resultados. A medida é calculada com a seguinte expressão:

$$\mathrm{sim}_{\mathrm{lin}}(c1, c2) = \frac{2 * \log(p(c3))}{\log(p(c1)) + \log(p(c2))} \tag{2.3}$$

onde *c3* é o conceito comum mais específico de *c1* e *c2*.

2.4.3 Proposta de Jiang & Conrath

Em seu trabalho Jiang e Conrath [13] propuseram uma medida de similaridade híbrida. Ou seja, uma proposta derivada das ideias das abordagens baseadas em arestas, mas que também considera a quantidade de informação (IC - *Information Content*) necessária para indicar o quão comum são os dois conceitos como fator determinante. Nessa abordagem a similaridade entre dois conceitos *c1* e *c2* é determinada pelas seguintes expressões:

$$\mathrm{ic}(c) = -\log(\mathrm{p}(c)) \tag{2.4}$$

$$\mathrm{d}(c1, c2) = \mathrm{ic}(c1) + \mathrm{ic}(c2) - 2 * \mathrm{ic}(\mathrm{ls}(c1, c2)) \tag{2.5}$$

$$\mathrm{sim}_{\mathrm{JiangConrath}}(c1, c2) = \frac{1}{\mathrm{d}(c1, c2)} \tag{2.6}$$

onde *p(c)* é a probabilidade de ocorrência do conceito *c* em um Corpus, *d(c1, c2)* é a distancia entre os conceitos *c1* e *c2*, e *ls(c1, c2)*, *Link Strength*, é expressão da diferença de quantidade de informação (IC) entre *c1* e *c2* e seu conceito comum mais específico *c3*.

2.5 Similaridade Semântica entre sentenças

Determinar similaridade semântica entre sentenças é uma tarefa crucial para a área de pesquisa de processamento de linguagem natural, pois são usadas em tarefas como a recuperação de informações, resposta automática a perguntas e resumo de textos. Porém, não é uma tarefa trivial. As medidas de similaridade semântica de termos ou conceitos são a base das técnicas de avaliação de similaridade entre sentenças ou frases. A similaridade é computada através das medidas semânticas entre grupos de termos e são classificadas em três tipos: as baseadas em Corpus, as baseadas no conhecimento e as híbridas [1].

- As **baseadas em Corpus** dependem das informações extraídas de um Corpus, como a frequência de ocorrência das palavras e sua ordem de ocorrência em frases.

- As **baseadas em conhecimento** exploram informações de dicionários, como o relacionamento entre as palavras, seus significados, classe gramatical e seus sinônimos.
- As **híbridas** combinam as duas técnicas anteriores e, por isso, geralmente apresentam melhores resultados.

2.5.1 Proposta de Feng

Os humanos avaliam a similaridade entre sentenças considerando suas relevâncias diretas e indiretas. A relevância direta diz respeito à percepção de coerência entre as sentenças. Enquanto a relevância indireta diz respeito à percepção da potencial relação entre as duas sentenças [8].

Feng *et al.* [8] basearam-se nesse conceito para propôr um método para estimar a similaridade entre sentenças. A abordagem proposta avalia a similaridade considerando a relevância direta e a relevância indireta entre frases.

A quantidade de informação contida em uma sentença é sensível ao tamanho da sentença e impacta na relevância direta. Além disso, segundo os autores, de maneira geral, as sentenças tendem a ser muito curtas e, por isso, propõem o enriquecimento das sentenças. O processo de enriquecimento consiste da identificação do principal significado (*Sense*) de cada palavra da sentença, com auxílio de um dicionário (WordNet), e da adição dos sinônimos desse primeiro significado ao conjunto de palavras a serem submetidas à analise.

Sejam *s1* e *s2* duas sentenças. A relevância direta entre elas é computada através da seguinte expressão:

$$\mathrm{f_1(dr)} = \frac{|\mathrm{s'1}|}{|\mathrm{s'1}| + |\mathrm{s'2}|}\mathrm{sig(s1, s2)} + \frac{|\mathrm{s'2}|}{|\mathrm{s'1}| + |\mathrm{s'2}|}\mathrm{sig(s2, s1)} \tag{2.7}$$

onde *s'1* e *s'2* são as respectivas versões enriquecidas, $|s'1|$ e $|s'2|$ a quantidade de palavras, e *sig(s1, s2)* a similaridade condicional que as palavras da sentença *s1* exercem sobre a sentença *s2*, e vice-versa.

Para computar a relevância indireta é utilizada a seguinte expressão:

$$\mathbf{f}_2(\mathrm{indr}) = \frac{\mathrm{F}_{|s1|,|s2|}}{\max(|s1|,|s2|)} \tag{2.8}$$

onde F é o resultado computado pelo algoritmo *Needleman-Wunsch*[5] adaptado para sequências de palavras.

Por fim, a similaridade entre as sentenças é computada com a seguinte expressão:

$$\mathbf{f}(\mathrm{dr},\mathrm{indr}) = \lambda \mathbf{f}_1(\mathrm{dr}) + (1-\lambda)\mathbf{f}_2(\mathrm{indr}) \tag{2.9}$$

onde λ representa o nível de contribuição da relevância direta e indireta no cálculo da similaridade. Os autores sugerem que este valor seja configurado para *0.85*, devido ao papel subordinado que a relevância indireta exerce na similaridade.

2.6 Considerações Finais

A Arquitetura Orientada a Eventos é um estilo arquitetural que visa permitir a integração de sistemas através da disseminação de informações. Essa disseminação de informação ocorre em decorrência da mudança de estado de alguma informação relevante para o contexto do negócio, o que é chamado de Evento. Os sistemas implementados sob esse estilo, como os que seguem o padrão *Publish*/*Subscribe*, possuem a vantagem de promover baixo acoplamento entre os componentes participantes da troca de informações no que diz respeito a Espaço, Tempo e Sincronização.

As implementações de *Publish*/*Subscribe* que utilizam assinaturas baseadas no conteúdo permitem também maior flexibilidade, permitindo que o filtro das informações de interesse dos assinantes seja feito com base no conteúdo das informações notificadas. Contudo, possuem a desvantagem de provocarem um forte acoplamento estrutural entre os dados das Notificações de eventos e das Assinaturas, por exigir homogeneidade nos nomes utilizados para os atributos.

Este trabalho apresenta uma solução para reduzir o impacto deste acoplamento, fa-

[5] O algoritmo **Needleman-Wunsch** foi proposto na década de 1970 por Saul Needleman e Christian Wunsch e é comumente utilizado na bioinformática para alinhar as sequências de proteínas ou nucleotídeos.

zendo com que, mesmo em situações em que ocorra heterogeneidade entre nos nomes utilizados para os atributos, o Servidor de Eventos seja capaz de identificar corretamente as assinaturas candidatas a recebimento das notificações de eventos.

Abordagens de avaliação de similaridade semântica, de termos e sentenças, foram utilizadas para avaliar a similaridade entre os nomes dos atributos das notificações dos publicadores e das assinaturas dos assinantes. O resultado dessa avaliação é o insumo utilizado pelo Servidor de Eventos para determinar corretamente se uma notificação de evento deve ser entregue aos assinantes.

3. Uma abordagem baseda em semântica para Pub/Sub

Este capítulo apresenta a Arquitetura *Publish/Subscribe* baseada em Similaridade Semântica proposta por este trabalho. Uma versão preliminar deste trabalho foi apresentada por Pimenta Jr. *et al.* [21]. Todo código fonte da solução proposta encontra-se disponível[1].

3.1 Visão Geral da Proposta

Este trabalho visa apoiar corporações na tarefa de integrar aplicações corporativas através de arquiteturas *Publish/Subscribe*.

A utilização do padrão *Publish/Subscribe* apresenta vantagem em relação às propostas tradicionais de integração, como as do tipo *Request-Response*, por promover o desacoplamento entre os sistemas participantes. Ao contrário das propostas mais tradicionais, em que a comunicação é ponto-a-ponto e síncrona, no padrão *Pub/Sub* as informações são disseminadas (notificadas) somente no momento em que são geradas, por meio de um intermediário (Servidor de Eventos), que garante a entrega a todos os consumidores. Dessa forma, os sistemas produtores e consumidores não precisam se conhecer, uma mesma informação do produtor pode ser enviada a vários consumidores, e os sistemas consumidores sempre terão disponíveis as informações mais atualizadas, sem a necessidade de checar constantemente se ocorreram modificações.

As implementações de *Pub/Sub* que utilizam assinaturas baseadas no conteúdo permitem maior flexibilidade em relação às demais (Seção 2.2.2), permitindo que o filtro das informações de interesse dos assinantes seja feito com base no conteúdo das informa-

[1] https://bitbucket.org/antoniojunior87/semanticpubsub

ções notificadas. Contudo, apresentam a desvantagem de provocar um forte acoplamento estrutural entre os dados das notificações de eventos e das assinaturas, por exigir homogeneidade entre os nomes dos atributos.

A solução apresentada neste trabalho pretende reduzir o impacto desse acoplamento, fazendo com que, mesmo em situações em que ocorra heterogeneidade entre os atributos das notificações e assinaturas, o Servidor de Eventos seja capaz de identificar corretamente as assinaturas candidatas ao recebimento das notificações. Para isso, o Servidor de Eventos foi dotado de mecanismos que se utilizam de técnicas de similaridade semântica que possibilitam realizar inferências. Além disso, o Servidor de Eventos foi estruturado em uma arquitetura de componentes plugáveis que podem ser substituídos a depender da aplicação.

A Figura 3.1 ilustra o funcionamento básico das interações dos participantes da solução de integração construída. O componente Publicador (*Publisher*) é o produtor de eventos, ou seja, representa os sistemas onde ocorre algum processamento lógico que produz ou altera alguma informação e que a disponibiliza como uma notificação de eventos. O Assinante (*Subscriber*) representa os consumidores de eventos, que ao assinar um padrão de notificações passam a ser notificados da ocorrência de eventos.

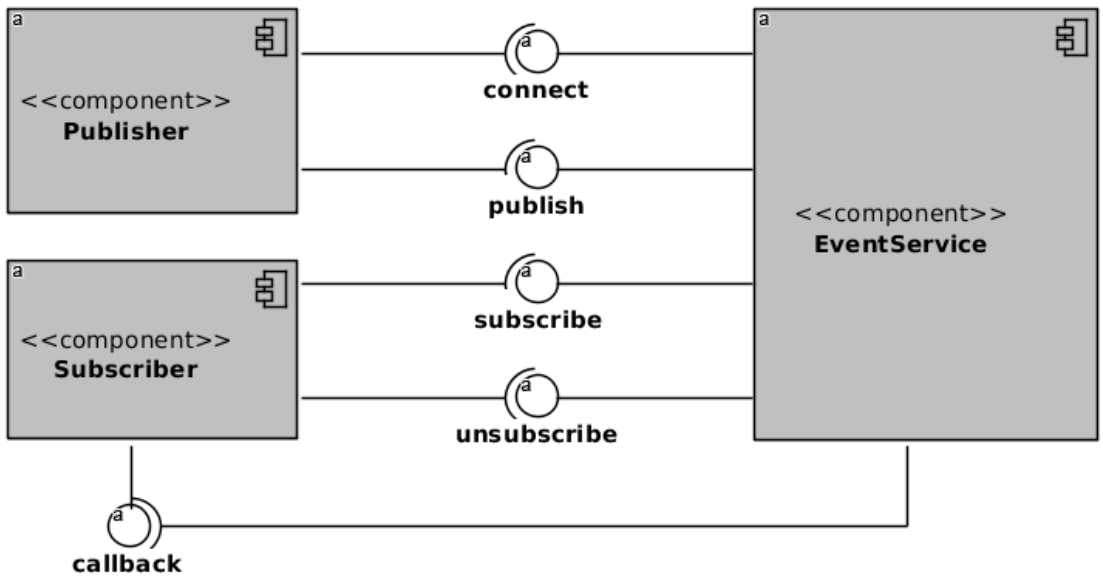

Figura 3.1: Integração entre Componentes

O Servidor de Eventos (*EventService*) é o principal componente deste tipo de pro-

posta e possui o papel de mediador na comunicação entre os publicadores e os assinantes. Sendo, então, o responsável por: (i) permitir que os componentes publicadores se conectem (*connect*) à solução e passem a notificar eventos; (ii) receber as notificações de eventos dos publicadores (*publish*); (iii) permitir que assinantes assinem (*subscribe*) e deixem de assinar (*unsubscribe*) tipos de eventos específicos que desejam consumir. Além dessas responsabilidades de integração com componentes externos, em seus processos internos, o servidor de eventos deve ser capaz de manter as assinaturas dos assinantes, armazenar temporariamente as notificações de eventos, identificar os assinantes de destino das notificações e, por fim, entregar as notificações aos assinantes (*callback*). As próximas seções apresentam com mais detalhes estes e outros processos internos do Servidor de Eventos.

3.2 Servidor de Eventos

A Figura 3.2 apresenta os componentes internos do Servidor de Eventos e a integração entre eles. Como um ponto de partida para a análise dos componentes que integram o Servidor de Eventos, o componente *EventService* representa a camada mais externa do Servidor de Eventos. Este componente é responsável por receber e tratar as requisições dos Publicadores e dos Assinantes. Nele estão disponíveis os pontos de acesso para os publicadores se conectarem à solução e enviarem notificações, e para os assinantes assinarem e deixarem de assinar eventos.

Dentre os fluxos internos implementados no Servidor de Eventos, é importante destacar como principais o tratamento das assinaturas e das notificações de eventos.

Ao receber uma nova assinatura, o *EventService* delega o seu tratamento ao *SubscriptionService* (*subscribe*). Este componente é responsável por solicitar a conversão da assinatura para o modelo de classes interno, pelo componente *SubscriptionConverter* (*convert*) e, posteriormente, solicitar que o componente *SubscriptionHandler* armazene as informações da assinatura no repositório de assinaturas (*insertSubscription*).

Ao receber uma notificação de evento, o *EventService* solicita o seu tratamento ao *NotificationService* (*publish*). Por sua vez, este componente coordena as etapas de processamento necessárias para preparar a notificação para envio aos assinantes. Em primeiro

lugar, é invocado o componente *NotificationConverter* (*convert*), responsável por converter a notificação para o modelo de classes interno. Em seguida, a notificação convertida é enviada ao componente *NotificationHandler* (*publishNotification*), onde o processo de publicação da notificação continua.

NotificationHandler é o componente responsável por orquestrar os processos necessários para entregar as notificações aos assinantes. Primeiro, são obtidas todas as assinaturas presentes no repositório de assinaturas (*findAllSubscription*). Em seguida, o componente *SubscriptionMatcher* é invocado (*matchSubscription*) para identificar, dentre a lista de assinantes obtida, os assinantes interessados na notificação em análise. Uma lista de assinantes interessados na notificação é gerada. Por fim, o componente *CallbackHandler* é invocado (*callback*) para que entregue a notificação a cada assinante identificado.

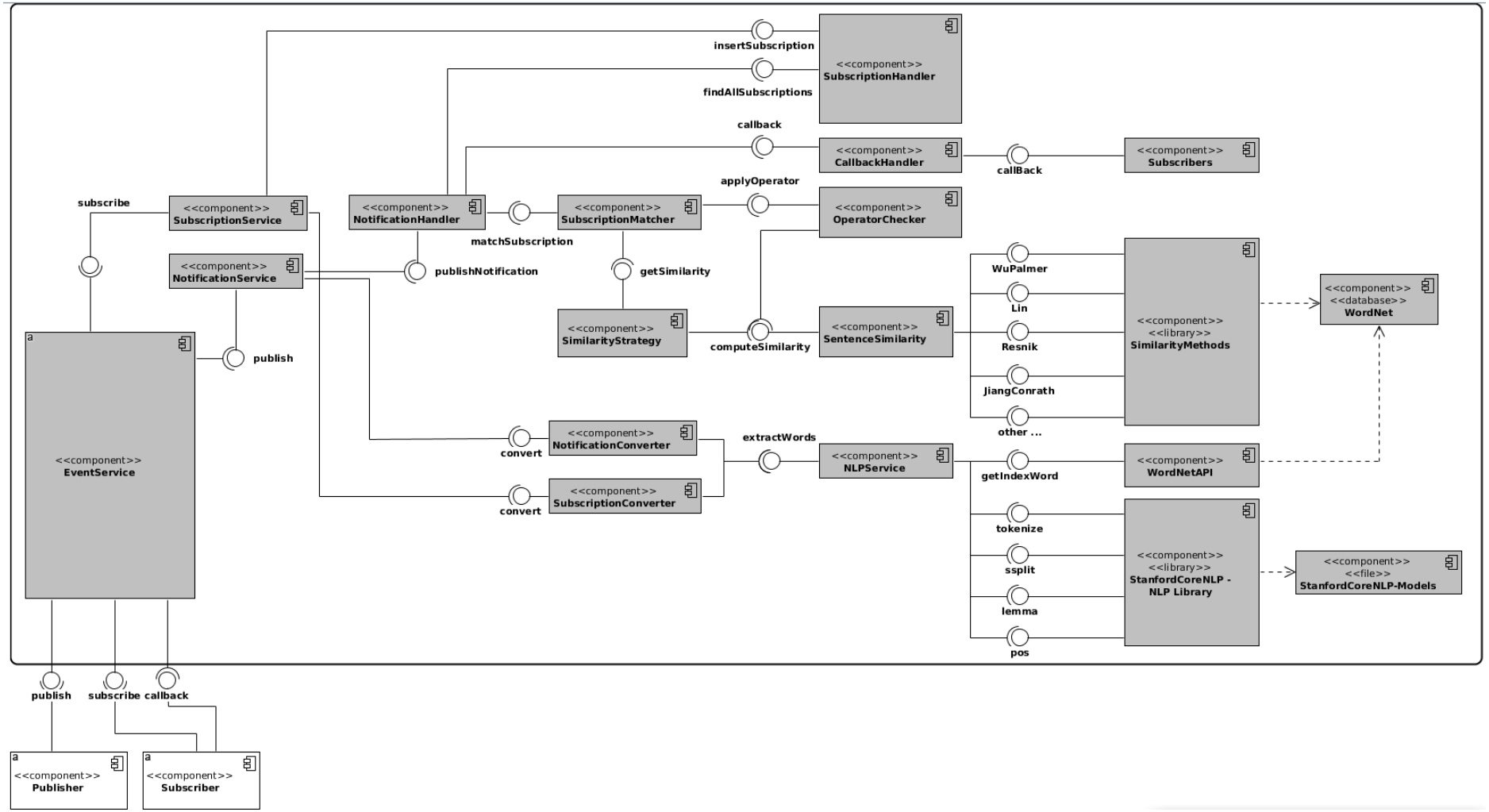

Figura 3.2: Componentes Internos do Servidor de Eventos

3.3 Componentes internos do Servidor de Eventos

Nesta seção são apresentados, com mais detalhes, todos os componentes internos do Servidor de Eventos, incluindo os processos de conversão das assinaturas e notificações, o processo de identificação dos assinantes interessados na notificação, bem como o modelo de classes interno.

3.3.1 Componente SubscriptionMatcher

No contexto do padrão *Pub/Sub*, afirma-se que um assinante assina um evento quando todo conjunto de restrições, descritas em sua assinatura, são atendidas pela notificação do evento, e consequentemente deve ser notificado de tal evento. Como dito, uma assinatura corresponde a uma lista de restrições aplicadas sobre os atributos das notificações. Cada restrição é composta pelo nome do atributo sobre o qual a restrição será aplicada, o operador de restrição e o valor de referência.

A solução proposta por este trabalho pretende reduzir o acoplamento existente entre assinantes e publicadores em relação à estrutura das notificações e assinaturas. Para isso, permite que os assinantes tenham a liberdade de expressar quais são seus eventos de interesse, sem necessariamente utilizar para isso as mesmas estruturas utilizadas pelos publicadores em suas notificações.

Para isso, o componente *SubscriptionMatcher* implementa o Algoritmo 1, que identifica para cada restrição da assinatura o atributo da notificação correspondente. Isto é, o algoritmo cria uma lista de pares "Restrição de Assinatura/Atributo de Notificação", onde cada par é composto por uma restrição da assinatura e do atributo da notificação com maior similaridade. A similaridade, entre os nomes do atributo da restrição e da notificação, é computada pelo método de similaridade entre sentenças adotado pela solução (Seção 3.3.2).

Após a identificação dos pares correspondentes, são aplicadas as restrições de cada par. O resultado final do algoritmo corresponde ao produto final de todas restrições combinadas e é um booliano indicando se o assinante da assinatura deve receber a notificação de evento ou não.

Algoritmo 1: Identificação de assinatura interessada na notificação

```
1  function match (notification, subscription);
   Input  : The notification and the subscription
   Output: result
2  similarPairs ← [pair1, ..., pairN] ;
                  |subscription|
3  for constraint in subscription do
4      maxSimilarity ← 0;
5      selectedAttribute ← NULL;
6      for attribute in notification do
7          similarity ← computeSimilarity(constraint, attribute);
           /* Seleciona atributo com maior similaridade          */
8          if similarity = 1.0 then
9              maxSimilarity ← similarity;
10             selectedAttribute ← attribute;
11             break;
12         else if similarity > maxSimilarity and similarity > threshold
            then
13             maxSimilarity ← similarity;
14             selectedAttribute ← attribute;
15         end
16     end
17     if selectedAttribute is not null then
18         similarPairs ← (constraint, selectedAttribute)
19     end
20 end
21 if similarPairs is empty then
22     return false;
23 end
24 for pair in similarPairs do
25     operatorResult ← applyOperator(pair);
26     if operatorResult is false then
27         return false;
28     end
29 end
30 return true;
```

3.3.2 Componentes SimilarityStrategy e SentenceSimilarity

Como apresentado na Seção 2.5, determinar similaridade semântica entre sentenças é uma tarefa crucial para a área de pesquisa de processamento de linguagem natural, pois são usadas em tarefas como a recuperação de informações, resposta automática a perguntas e resumo de textos [1]. Neste trabalho, estas técnicas foram utilizadas para computar a similaridade entre os nomes dos atributos das notificações de eventos e das restrições das assinaturas.

Abordagens de avaliação de similaridade semântica, de termos e sentenças, foram utilizadas para avaliar a similaridade entre os nomes dos atributos das notificações dos publicadores e das assinaturas dos assinantes. O resultado dessa avaliação é o insumo utilizado pelo Servidor de Eventos para determinar corretamente se uma notificação de evento deve ser entregue aos assinantes.

O componente *SimilarityStrategy* é o responsável por verificar se um atributo da notificação é correspondente a um atributo da assinatura. Para isso, segue a seguinte estratégia: (i) solicita ao componente *SentenceSimilarity* o percentual de similaridade entre as sentenças (nomes dos atributos); (ii) verifica se o percentual de similaridade é superior ao limite mínimo (*Threshold*) configurado na solução.

O componente *SentenceSimilarity* é o responsável por computar a similaridade entre os nomes dos atributos das notificações de eventos e das restrições das assinaturas. Este componente implementa a abordagem de avaliação de similaridade entre sentenças adotada. Foram construídas duas versões deste componente, a primeira delas implementa a verificação da similaridade entre as sentenças baseando-se na similaridade média entre as palavras das sentenças. A outra versão implementa a abordagem proposta por Feng [8] (Seção 2.5.1), que avalia a similaridade considerando a relevância direta e a relevância indireta entre sentenças.

As medidas de similaridade semântica de termos ou conceitos são a base das técnicas de avaliação de similaridade entre sentenças ou frases. Por isso, a abordagem implementada utiliza o componente *SimilarityMethods* (Seção 3.3.3), que fornece a implementação dos métodos de similaridade entre termos.

Os dois componentes são plugáveis e podem ser substituídos por outros componentes que implementem outras abordagens. Para isso, os componentes substitutos devem implementar as interfaces apresentadas na Figura 3.3.

ISimilarityStrategy deve implementar o método *checkSimilarity*, que recebe como parâmetro o atributo da restrição da assinatura avaliada e o atributo da notificação. O método deve retornar resultado positivo caso a verificação aponte que as duas sentenças são similares de acordo com a estratégia adotada.

ISentenceSimilarity deve implementar o método *computeSimilarity*, que recebe como parâmetro duas listas de *Words*, uma de cada sentença, e retorna um número decimal entre zero e um.

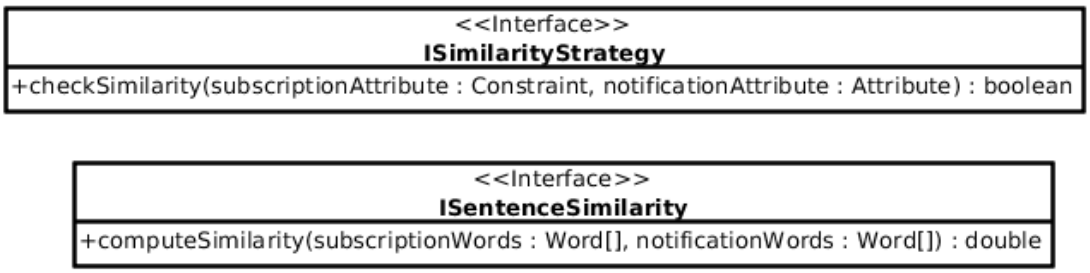

Figura 3.3: Componentes de avaliação de similaridade

3.3.3 SimilarityMethods

O componente *SimilarityMethods* é uma fachada para componentes que implementam métodos de similaridade, como, por exemplo, o WS4J empregado na implementação de referência deste trabaho.

O componente WS4J [25] (*WordNet Similarity for Java*) fornece implementações de métodos de similaridade entre termos (Seção 2.4), utilizando a WordNet (Seção 2.3) como recurso linguístico. O componente inclui as medidas de similaridade de Wu Palmer, Jiang Conrath, Lin, Resnik e muitas outras. Estas medidas foram selecionadas por apresentarem os melhores resultados em avaliações realizadas no trabalho [20] e [27].

Todos os métodos disponíveis no componente seguem o mesmo padrão de uso. Quando invocados, precisam receber como parâmetro o par de palavras ao qual se deseja computar a similaridade. Como resultado, os métodos retornam um número decimal entre zero

e um, que representa o percentual de similaridade entre as palavras.

Outro aspecto importante do componente *SimilarityMethods* é que ele guarda as similaridades computadas em cache. Assim, quando um par de palavras é submetido ao processo de verificação de similaridade, o primeiro passo do componente é verificar se o resultado desta verificação já existe no *cache*. Caso não exista, o processo de verificação é iniciado normalmente e, no final, o resultado é armazenado no *cache*. Essa estratégia proporciona uma melhoria de desempenho a um dos processos mais custosos da solução.

O *cache* também é utilizado para implementar outra funcionalidade importante da solução. Em certas situações a solução pode não identificar o grau de similaridade entre duas palavras da maneira desejada, por isso foi incluída a possibilidade de incluir similaridades pré-computadas manualmente através dos arquivos de configuração. Dessa forma, ao ser iniciado, o Servidor de Eventos trata de carregar essas similaridades pré-configuradas no *cache* de similaridades. Isso garante que no momento da verificação de similaridade entre um par de palavras pre-computado seja utilizado o valor configurado.

3.3.4 Modelo de classes

Todos os processos internos do Servidor de Eventos seguem um modelo canônico de classes que padroniza os dados trafegados entre os processos. A Figura 3.4 apresenta o modelo de classes em questão.

A entidade *Subscription* é a representação das Assinaturas neste modelo. Ela é composta por um conjunto de restrições, chamados de *Constraints*. Além disso, as *Subscriptions* possuem um identificador único (*id*), o endereço de chamada de retorno (*callback*) para o qual as notificações de eventos serão enviadas, e o campo *originalSubscription* onde é armazenada a assinatura em seu formato original.

Já *Notification*, é a entidade que representa as Notificações dos publicadores. Ela também possui um campo identificador (*id*), os atributos (*Attribute*) da notificação, e o campo *originalEvent* que armazena a notificação em seu formato original.

A entidade *Attribute* possui as informações básicas de um atributo de uma Notificação. São eles, o nome (*name*), o valor (*value*) e uma lista *Words*.

Um *Constraint*, por sua vez, representa uma restrição a ser aplicada sobre um atributo da Notificação (*Notification*). Por isso, utiliza *Attribute* através de uma composição, e é formada pelo nome do atributo (*name*), o valor (*value*), uma lista *Words* e, adicionalmente, o *Operator*.

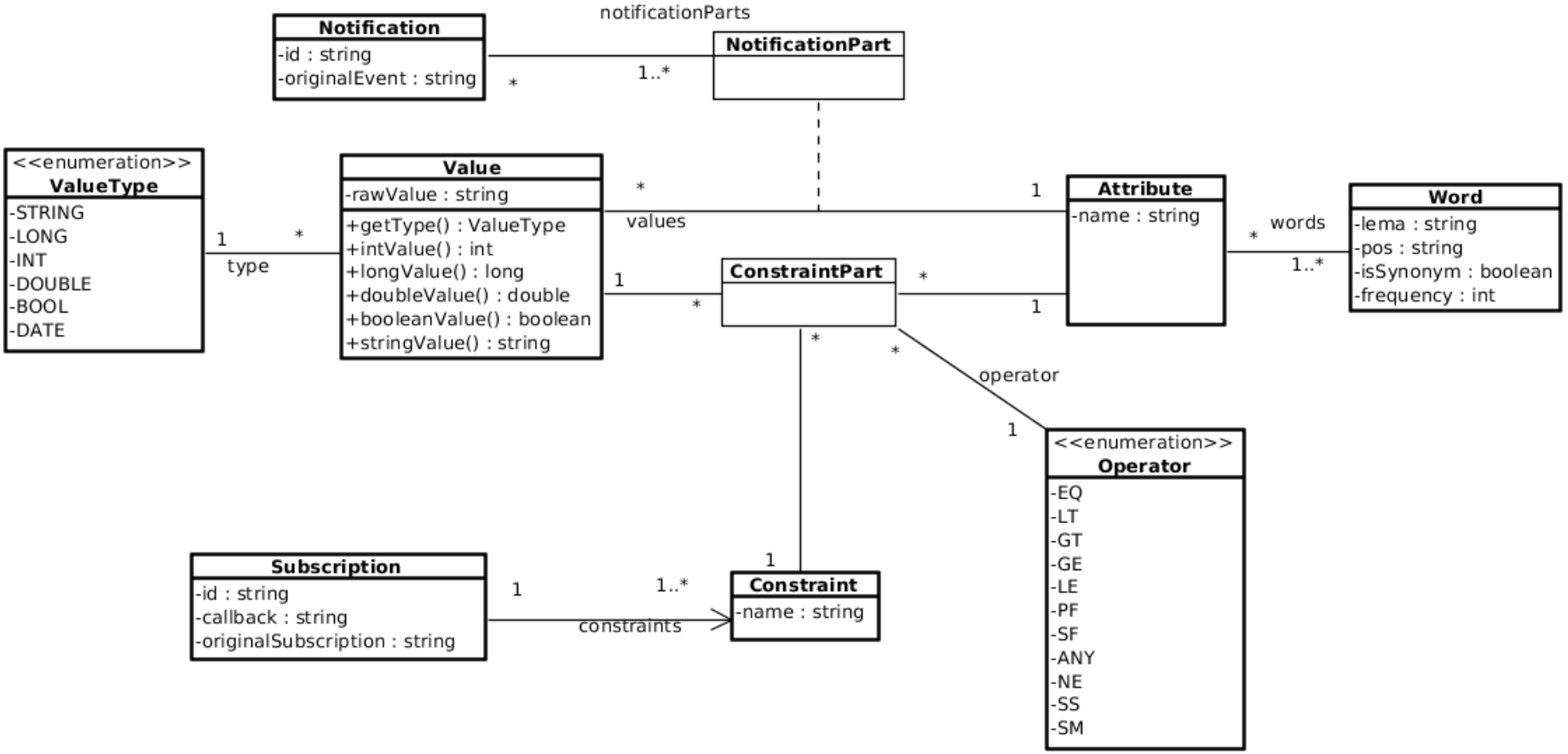

Figura 3.4: Modelo de Classes do Servidor de Eventos

Operador	**Descrição**
GT	maior
LT	menor
GE	maior ou igual
LE	menor ou igual
NE	diferente
PF	possui o prefixo (Apenas p/ *Strings*)
SF	possui o sufixo (Apenas p/ *Strings*)
SS	possui a *substring* (Apenas p/ *Strings*)
SM	é similar a *string* (Apenas p/ *Strings*)
ANY	Qualquer valor será considerado válido

Tabela 3.1: Operadores válidos

O *Operator* é a representação dos operadores das restrições. A Tabela 3.1 apresenta todos os Operadores implementados na solução e uma descrição do seu respectivo funcionamento.

Os nomes dos *Attributes* das *Notifications* e os nomes dos *Constraints* das *Subscriptions* podem ser compostos por várias palavras. Devido ao processo de avaliação de similaridade proposto pela solução deste trabalho, estes nomes passam pelo processamento de linguagem natural e enriquecimento de sentença (Seção 3.3.8) que gera a lista de *Words* necessária para esta avaliação. Estas listas são associadas aos *Attributes* e *Constraints*.

A entidade *Word* armazena todas as informações, de cada palavra dos nomes de atributos e restrições, necessárias para o processo de avaliação de similaridade. Entre elas o lema, a classe gramatical (POS), a frequência da palavra no Corpus (*Frequency*) e um identificador se o Word foi gerado pelo processo de enriquecimento (*isSynonym*).

Os valores dos *Attributes* e dos *Constraints* são representados no modelo pela entidade *Value*. Esta entidade é capaz de armazenar o valor nos diferentes tipos definidos pela Enumeração *ValueType*. A Tabela 3.2 apresenta os tipos disponíveis e seu respectivo correspondente na linguagem Java.

ValueType	Correspondente em Java
STRING	String
LONG	Long
INT	Integer
DOUBLE	Double
BOOL	Boolean

Tabela 3.2: Tipos de valores válidos

3.3.5 Modelo de assinatura

Os assinantes geralmente estão interessados em eventos ou padrões de eventos específicos e não em todos os eventos. O padrão *Pub/Sub* permite que o assinante expresse quais eventos são de seu interesse através de uma assinatura.

Uma assinatura corresponde a uma lista de restrições aplicadas sobre os atributos das notificações. A solução proposta por este trabalho estabelece que a lista de restrições das assinaturas deve ser informada entre chaves, como mostra o Exemplo 3.1. Cada restrição é composta pelo nome do atributo sobre o qual a restrição será aplicada (por exemplo, *umidade*), o operador de restrição (por exemplo, maior que - *GT*) e o valor de referência (por exemplo, *60*). Os campos são separados por vírgula.

Exemplo 3.1 (Assinatura)**.** *{umidade, GT, 60; temperatura média, LE, 25}*

Nas assinaturas, o nome do atributo poderá ser um termo simples ou composto, como "temperatura média" no exemplo. Para garantir maior poder de restrição, a linguagem de assinatura suporta diversos operadores (Tabela 3.1). Além disso, a solução mapeia automaticamente os tipos de dados Numéricos (*Integer* e *Double*), Booleanos (*True/False*), e sequencia de caracteres (*Strings*).

3.3.6 Modelo de evento

Como definido na Seção 2, em Engenharia de Software, o termo *Evento* é usado para denominar um acontecimento que provoca mudança em alguma informação relevante para o contexto do negócio. A nível de Software, os Eventos são abstraídos na forma de objetos *Evento*, que encapsulam os dados do Evento em formato que o computador pode

entender. Geralmente, documentos XML ou JSON são utilizados para esse fim.

Na solução proposta por este trabalho foi definido que os eventos devem ser enviados em uma estrutura baseada na especificação do JSON Object [12]. A estrutura de um objeto JSON é representada como um par de chaves envolvendo um ou mais pares de "chave:valor" separados por vírgulas, como ilustra a Figura 3.5, onde a *chave* é uma *String* e o *valor* é do tipo de dados numérico (*Integer* e *Double*), Booleano (*True/False*), ou *String*.

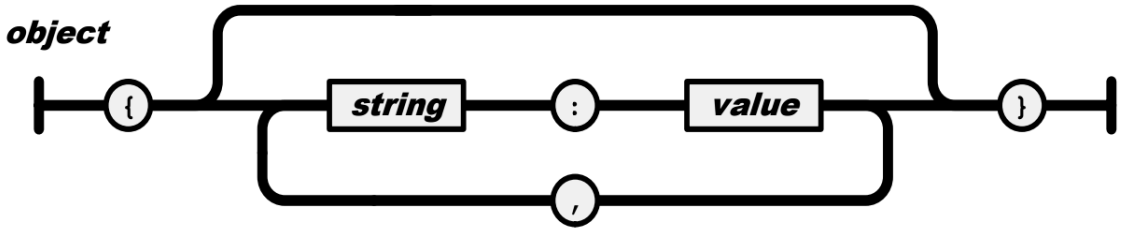

Figura 3.5: Objeto JSON

3.3.7 Conversores de Assinaturas e Notificações

Os componentes conversores são responsáveis por transformar e encapsular os dados originais das notificações e assinaturas nas classes internas da solução *Notification* e *Subscription*, detalhadas na Seção 3.3.4. A conversão está diretamente relacionada ao modelo de notificação e assinatura escolhido no desenvolvimento da solução. Por esse motivo, os componentes conversores são substituíveis, de maneira que é possível empregar na solução outros conversores capazes de tratar o modelo de assinaturas e notificações escolhido.

Os conversores já implementados e presentes na solução são capazes de converter assinaturas e notificações nos moldes apresentados nas Seções 3.3.5 e 3.3.6. Caso seja necessário tratar outros modelos, é possível incluir novos conversores. Para isso é necessário seguir o padrão estabelecido pelas interfaces apresentadas na Figura 3.6. O conversor de notificações deve implementar a interface *INotificationConverter* e deve implementar o método *convert*, que recebe como parâmetro a notificação em seu formato original e retorna uma instância de *Notification*. O conversor de assinaturas reflete o comportamento de *ISubscriptionConverter*, implementando o método *convert* que deve receber a assinatura em seu formato original e a URL de *Callback* do assinante.

Para gerar a lista de *Words* das *Notifications* e *Subscriptions* é necessário invocar o componente *NLPServices*, cujo comportamento é detalhado na Seção 3.3.8.

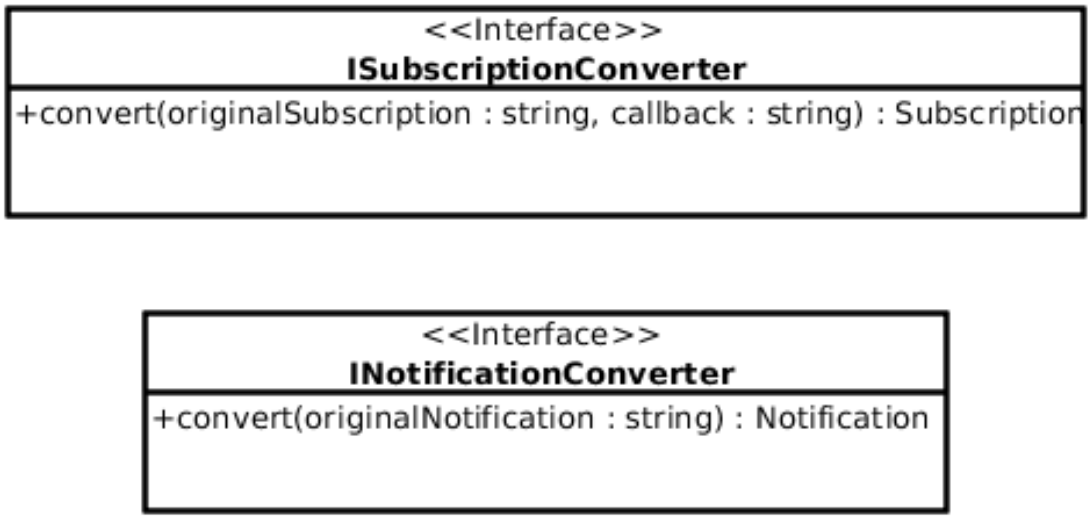

Figura 3.6: Interfaces dos conversores

3.3.8 Componente NLPServices

Devido à característica dos dados tratados neste trabalho, notificações de eventos e assinaturas descritos na Seção 3.3.6 e na Seção 3.3.5, é necessário, antes de mais nada, fazer um tratamento nestes dados para que eles possam ser processados pelo restante dos processos que envolvem a solução proposta neste trabalho.

O componente *NLPServices* implementa todas as questões que envolvem o processamento de linguagem natural utilizadas neste trabalho. Sua principal responsabilidade é preparar os nomes dos atributos das notificações de eventos e assinaturas, e obter todas as informações necessárias para os demais processos da solução. A classe *Word*, apresentada na Seção 3.3.4, armazena todas essas informações. Entre elas estão o lema, a classe gramatical (POS), a frequência da palavra no Corpus (*Frequency*) e um identificador se o Word foi gerador pelo processo de enriquecimento (*isSynonym*).

A Figura 3.7 apresenta o diagrama das atividades realizadas pelo componente *NLPServices*. Este processo é aplicado a todos os atributos, seja ele atributo de notificação ou uma restrição de assinatura. A Figura 3.8[2] exemplifica a execução do fluxo dos dados ao

[2]Notação livre, empregada para ilustrar a execução das atividades referentes ao diagrama da Figura 3.7.

longo das etapas do processamento no componente *NLPServices*.

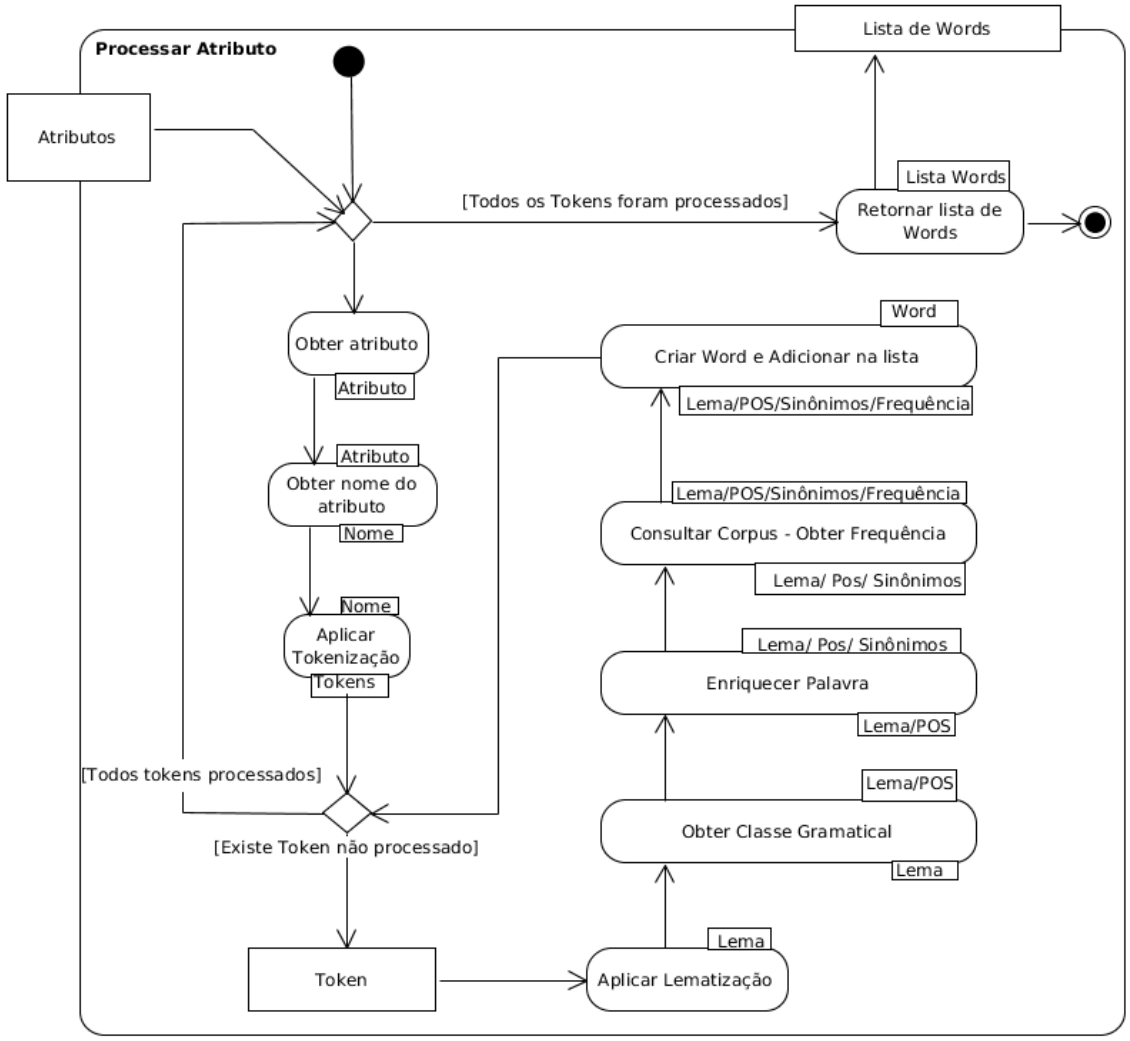

Figura 3.7: Processamento dos atributos

Após receber os atributos a serem processados, a primeira atividade obtém o nome de cada atributo sendo tratado. Em seguida, sobre o nome obtido é aplicada a atividade chamada de ***Tokenização*** (*Tokenization*), que consiste em remover, se existir, a pontuação e outros caracteres indesejados e separar palavras escritas na forma *CamelCase*[3]. Desse processo são originados os *Tokens* das sentenças.

Posteriormente, na terceira atividade é aplicada a técnica de Lematização (*Lemmatization*) para a identificação do Lema[4] do *Token*.

Na atividade seguinte, cada Lema passa pelo processo de **Marcação** (*Tagging*), que

[3] ***CamelCase*** é a denominação em inglês para a prática de escrever palavras compostas ou frases, onde cada palavra é iniciada com maiúsculas e unidas sem espaços.

[4] O **Lema** é a forma gráfica canônica de uma palavra. Também é utilizada como entrada de verbete em dicionários ou vocabulários.

consiste em identificar a classificação gramatical (POS - *Part Of Speech*)[5].

Na etapa seguinte, de **Enriquecimento** da palavra, com o auxílio da WordNet no papel de dicionário, são obtidos o significado (*Sense*) mais comum do Lema e seus sinônimos. Cada sinônimo identificado é adicionado à lista de palavras.

Por fim, é obtida a frequência de ocorrência de cada palavra presente na lista no Corpus[6] utilizado pela solução. Como objeto de saída de todo esse processo é gerada uma lista de objetos da classe *Word*, que é atribuída ao atributo tratado.

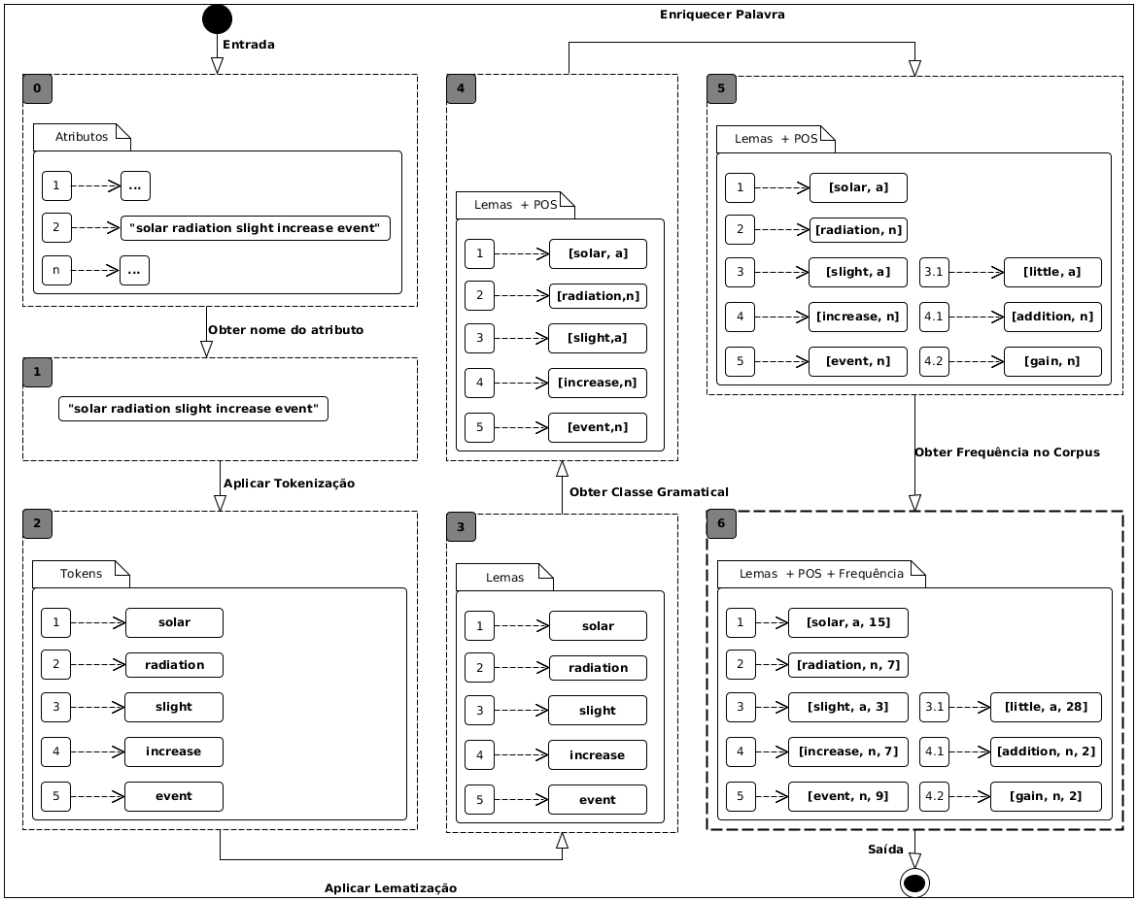

Figura 3.8: Etapas do NLPServices

Todos essas etapas do processo foram desenvolvidas utilizando como apoio as bibliotecas *Stanford CoreNLP* [16], JWNL [6] e a *WordNet* [18].

[5] **n** - substantivo; **v** - verbo; **a** - adjetivo. **r** - adverbio.

[6] Corpus extraído de um Dump das páginas Wikipedia através da ferramenta *Wikiextractor*.

3.3.9 Componente CallbackHandler

O componente *CallbackHandler* é o responsável por estabelecer a conexão com os assinantes e enviar as notificações de eventos. Na implementação da solução proposta, o componente *CallbackHandler* é capaz de enviar notificações para os assinantes através do método POST dos protocolos HTTP[7] e HTTPS[8], tendo como *endpoint* a URL presente no campo *Callback* da *Subscription* do assinante.

Caso seja necessário utilizar outros protocolos ou mecanismos de comunicação, é possível plugar um novo *CallbackHandler*, desde que ele siga as especificações da interface *ICallbackHandler* (Figura 3.9). Esta interface estabelece que o componente deve implementar o método *callback* que recebe como parâmetros a *Subscription* e a *Notification*, e retorna um booliano informando o resultado do envio.

Figura 3.9: Interfaces ICallbackHandler

[7] O *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto) é um protocolo de comunicação utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos. Ele é a base para a comunicação de dados da Internet.

[8] HTTPS é uma implementação do protocolo HTTP sobre uma camada adicional de segurança.

4. Experimentos e Análise dos Resultados

Este capítulo apresenta os experimentos computacionais realizados para avaliação da proposta. Os dados utilizados nos experimentos e os resultados encontram-se disponível na Internet[1]. Todos os experimentos foram realizados em uma máquina com processador *Intel Xeon* com 8 núcleos de 3.70 GHz e com 32 GB de RAM disponível e dedicação exclusiva.

4.1 Planejamento do Experimento

O método de pesquisa aplicado neste trabalho foi o quantitativo. Este método é centrado na coleta de dados com o objetivo de avaliar o estado de alguma variável de um determinado domínio no mundo real. O método aplicado segue o processo proposto por Recker [22], que compreende as seguintes atividades:

1. **Geração da teoria e hipótese**: Inicialmente foi realizado um levantamento teórico para identificar as possíveis soluções para atendimento da hipótese apresentada.

2. **Desenvolvimento de instrumentos para medição**: Foi implementada uma solução de referência para a arquitetura proposta e realizados experimentos para analisar sua viabilidade e eficácia.

3. **Coleta de dados**: Foi utilizada uma abordagem experimental para coleta de dados. Diferentes cenários foram definidos com variações nos tipos de eventos e assinaturas, para cobrir as diferentes possibilidades de aplicação da solução. Dados sobre

[1] https://bitbucket.org/antoniojunior87/semanticpubsub

os eventos publicados e encaminhados aos assinantes e o tempo de processamento foram coletados.

4. **Análise de dados e avaliação dos resultados**: Os dados coletados foram analisados empregando a técnica descritiva. Algumas métricas foram utilizadas para avaliar o desempenho e a eficácia da solução.

4.2 Métricas Utilizadas

De maneira geral, este trabalho propõe uma solução para garantir o desacoplamento de estrutura de dados das notificações de eventos em situações em que ocorre heterogeneidade entre os sistemas produtores e os consumidores que utilizam o padrão *Publish/-Subscribe*. O que consiste, entre outras coisas, em uma solução capaz de inferir se uma notificação de evento possui informações compatíveis com as descritas em uma assinatura. Esta inferência requer o mapeamento dos campos da notificação aos da assinatura.

Bellahsene *et al.* [2] apresentam um *Benchmark* para avaliação de soluções de mapeamento de campos. Neste *Benchmark* são avaliados, entre outros fatores, o tempo de geração dos mapeamentos e a eficácia do mapeamento gerado. O tempo de geração diz respeito ao tempo que a solução demanda para determinar a relação entre origem e destino. A eficácia é mensurada em relação à *Cobertura* e à *Precisão* do mapeamento gerado.

Na avaliação deste trabalho foram utilizadas três das métricas de avaliação de eficácia utilizadas por Bellahsene *et al.* [2]: Cobertura, Precisão e Medida-F. Para isso, os dados coletados são organizados nos conjuntos a seguir (Figura 4.4):

- O conjunto de **Elementos Relevantes** (*Relevant Elements*) representa o resultado esperado, ou seja, todos os valores que deveriam ser selecionados por uma solução ótima.
- O conjunto de **Elementos Selecionados** (*Selected Elements*) representa os elementos que foram selecionados pela solução em análise.
- Os **Verdadeiro Positivos** (TP ou *True Positives*) são os elementos presentes no conjunto de elementos relevantes e que foram selecionados pela solução em análise.

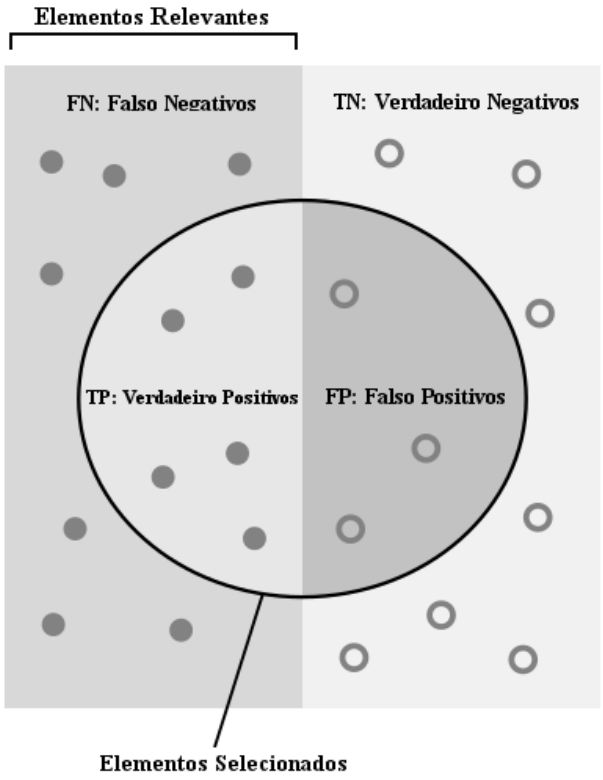

Figura 4.1: Conjuntos de dados para análise

- Os **Falso Positivos** (FP ou *False Positives*) são os elementos presentes no conjunto de elementos selecionados mas que não estão presentes no conjunto de elementos relevantes.

- Os **Falso Negativos** (FN ou *False Negatives*) são os elementos presentes no conjunto de elementos relevantes mas que não estão presentes no conjunto de elementos selecionados.

- Os **Verdadeiro Negativos** (TN ou *True Negatives*) são os elementos que não estão presentes no conjunto de elementos relevantes e que, corretamente, não foram incluídos no conjunto de elementos selecionados.

A medida de **Precisão** (P ou *Precision*) mede a proporção de elementos realmente relevantes selecionados (TP) em relação ao total de elementos selecionados (TP+FP).

$$P = \frac{TP}{TP + FP} \tag{4.1}$$

A **Cobertura** (C ou *Recall*) mede a proporção de elementos relevantes selecionados (TP) em relação ao conjunto de todos os elementos relevantes. Sendo que o conjunto de todos os elementos relevantes corresponde à união dos Verdadeiro Positivos e Falso Negativos (TP+FN).

$$C = \frac{TP}{TP + FN} \tag{4.2}$$

Na análise da proposta desta dissertação, *Precisão* corresponde à capacidade da solução errar pouco, ou seja, minimizar a quantidade de eventos enviados incorretamente aos assinantes. Já a *Cobertura* corresponde à capacidade de enviar o máximo de eventos relevantes para os assinantes.

Alta *Cobertura* combinada com baixa *Precisão* não é um bom resultado. Um cenário como este corresponde a uma solução capaz de identificar a maior parte dos eventos previstos para serem entregues, mas que também acaba enviando incorretamente muitos eventos não previstos. Da mesma forma, alta *Precisão* combinada com baixa *Cobertura* significa que a solução envia poucos eventos incorretamente, mas em contrapartida não cobre todos os casos previstos.

A **Medida-F** (*F-measure* ou *F-Score*) é a medida que combina *Precisão* e *Cobertura* em uma média harmônica. Esta média é importante pois é através dela que se pode observar as desproporções em relação à Cobertura e à Precisão, como assinaladas acima. Uma situação onde o valor da Medida-F for baixo demais pode indicar que a solução apresenta baixa eficácia. Ou seja, não seleciona corretamente os elementos assinalados como relevantes.

$$F = \frac{2 \times P \times C}{P + C} \tag{4.3}$$

Além das métricas de avaliação de eficácia, também foram utilizadas métricas para avaliar o desempenho da solução. Para esta avaliação foram utilizadas as métricas apresentadas no trabalho de Murth *et al.* [19]:

- **Taxa de Notificação (*Throughput*)**: é definida pelo número de notificações com-

pletamente processadas em um dado intervalo de tempo. Entende-se como notificação completamente processada aquela que foi recebida do *Publicador* e notificada ao *Assinante* ou descartada, caso não sejam identificados *Assinantes* interessados por aquela notificação.

- **Tempo de Notificação**: é definido pelo intervalo de tempo compreendido entre o momento em que o *Servidor de Eventos* recebe o evento publicado pelo *Publicador* e o momento em que o *Assinante* é notificado.

Na avaliação deste trabalho a *Taxa de Notificação* será apresentada na escala de *quantidade de eventos processados por segundo* (Notificações/Segundo) e o *Tempo de Notificação* estará na escala de milissegundos.

4.3 Modelagem dos Cenários e dos Datasets

O Capítulo 2 apresentou os diferentes tipos de métodos de avaliação de similaridade. Por sua vez, o Capítulo 3 destacou uma característica fundamental da proposta, a possibilidade de configurar a solução com esses diferentes tipos de métodos. Outra opção de configuração é a mudança do parâmetro *Threshold*, o ponto de corte usado para considerar os campos similares. A solução permite ainda que sejam adotadas diferentes estratégias de seleção dos atributos correspondentes.

Foi importante incluir no experimento variações de cenários que utilizassem diferentes estratégias de avaliação de similaridade, entre palavras e sentenças, e variações nos valores do *Threshold*, para permitir uma análise de viabilidade da proposta e a eficácia da solução implementada. Foram utilizadas quatro diferentes estratégias de avaliação de similaridade entre palavras: (i) Wu & Palmer [28]; (ii) Lin [14]; (iii) Resnik

Estas variações de cenários foram importantes para registrar comportamentos da solução, e consequentemente da arquitetura proposta, diante de diferentes situações. Este registro pode servir de importante contribuição para a tomada de decisão de como configurar a solução dependendo do cenário de aplicação da arquitetura.

Para a avaliação de desempenho foi utilizada a configuração da solução que apresentou

maior eficácia. Foi testado o comportamento diante de diferentes quantidades de assinantes simultâneos, desde o cenário sem nenhum assinante conectado até situações mais extremas onde foram simulados até 50 assinantes simultâneos. Nestes experimentos, foram avaliadas as Taxas de Notificação (*Throughput*) e o Tempo Médio de Notificação.

4.3.1 *Datasets*

O *Benchmark* proposto por Bellahsene *et al.* [2] foi útil para embasar o planejamento dos experimentos e fornecer as métricas, porém, o *Dataset* proposto não possui as características particulares de um solução *Publish/Subscribe*. Por isso, para avaliação da arquitetura proposta e da solução implementada foi necessário utilizar um conjunto de dados que simulasse este tipo de cenário, com larga escala e com a heterogeneidade esperada entre assinantes e publicadores. O trabalho de Hasan [11] apresentou um *Dataset* modelado com base em dados reais e de larga escala que atendeu estes requisitos.

Para gerar os *Datasets* foram utilizados dados reportados pelos sensores IoT (Internet das Coisas) dos *SmartSantander Smart City*[2] [24] e *Linked Dataspace for Energy Intelligence* [4][3].

Entre os dados utilizados no *Dataset* estão: (i) as medições dos sensores *Smart City*: radiação solar, velocidade e direção do vento, temperatura, umidade relativa, pressão atmosférica e precipitação pluvial; (ii) localização geográfica dos sensores: continente e país; (iii) localização no interior de prédios: sala, sala reunião, sala de conferência, recepção, escritório, banheiro; (iv) as medições dos sensores de monitoramento de energia: informação de presença em uma sala, consumo de energia de notebook, consumo do arcondicionado, e muitos outros.

O *Dataset* é composto por todo conjunto de dados necessário para executar o experimento e fazer as avaliações propostas. Eles incluem as assinaturas que serão subscritas ao servidor de eventos, todos os eventos que serão publicados e os *labels* que mapeiam o resultado esperado (conjunto dos elementos relevantes). Os *labels* ligam cada assinatura

[2]O projeto *SmartSantander Smart City* visa, entre outras coisas, dar subsídios para as pesquisas envolvendo IoT. Entre as contribuições deste trabalho estão os dados coletados de mais de 14 mil sensores espalhados por cidades da Europa.

[3]O projeto *Linked Dataspace for Energy Intelligence* propôs uma plataforma de monitoramento de consumo inteligente para empresas.

aos eventos que deverão ser entregues ao assinante até o final do experimento. Estes dados estão organizados em três arquivos de texto, um para as assinaturas, um para os eventos e outro para os *labels*. São cerca de 15 mil notificações de eventos e 200 assinaturas diferentes.

Nos arquivos das assinaturas e das notificações de eventos, cada linha inicia com um número que identifica a Assinatura ou Evento. O número identificador é utilizado para rastrear a dinâmica dos dados durante os experimentos. Após o número identificador, e separado por tabulação, aparece a Assinatura ou Evento propriamente dito. A Figura 4.2 mostra um exemplo de linha destes arquivos.

136-2 {**type**=atmospheric pressure increase event, **measurement unit**=pascals, **continent**=European countries, **residential area**=Lubeck, **country**=Germany}

Figura 4.2: Linha do arquivo de notificações de eventos

No arquivo dos *labels*, cada linha associa uma assinatura ao conjunto de eventos esperados para aquela assinatura. A linha é iniciada com o identificador da assinatura, seguido de tabulação e da lista, entre colchetes, de identificadores dos eventos esperados. Na Figura 4.3 está representada uma linha do arquivo.

11-1 [75-32, 75-33, 75-30, 75-31, 75-89, 75-88, 75-87, 75-86]

Figura 4.3: Linha do arquivo dos *labels*

4.4 Coleta de Dados

Para permitir a avaliação do modelo arquitetural proposto, foi implementada uma solução de referência. Esta solução foi submetida a experimentos simulando cenários reais mapeados pelos *Datasets*.

Para coleta de dados foi criada uma configuração da solução que permitisse a execução dos experimentos e alternância dos cenários, através da substituição dos *Datasets*. Para isso alguns componentes precisaram ser construídos. A Figura 4.4 ilustra a configuração do experimento.

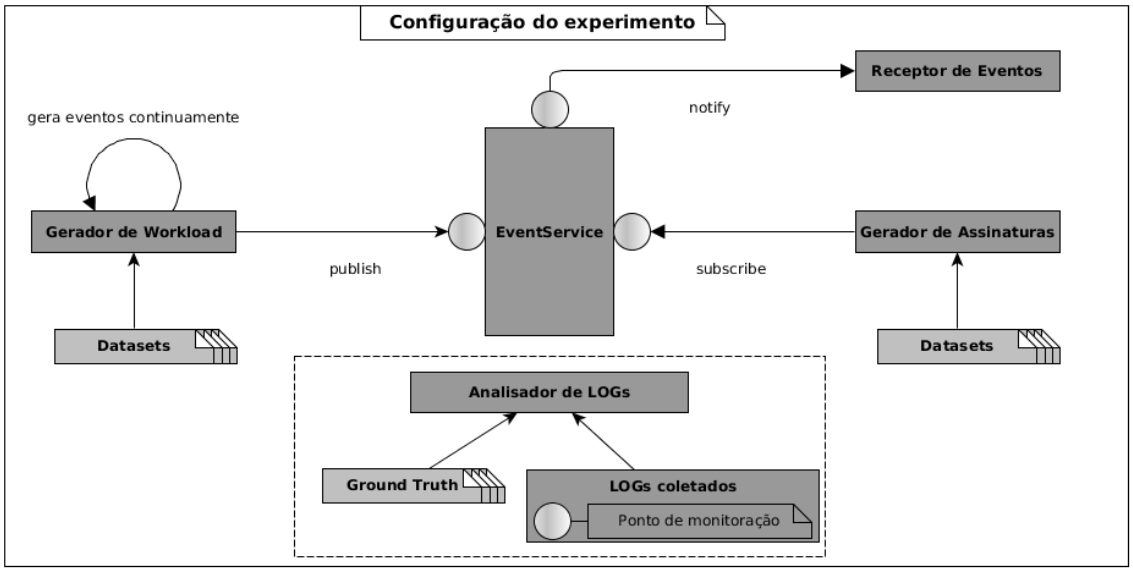

Figura 4.4: Configuração para coleta de dados

O **Gerador de *Workload*** simula o comportamento de um Publicador e é o responsável por ler de maneira contínua o *Dataset* de eventos e enviá-los ao Servidor de Eventos. O **Gerador de Assinaturas** simula o processo de assinar eventos realizado pelos Assinantes, ele cria as assinaturas com base no *Dataset* de assinaturas. O **Receptor de Eventos** simula um tratamento que poderia ser dado por um Assinante a um evento recebido.

Foram inseridos no **Servidor de Eventos** três pontos de monitoração, representados na Figura 4.4 com os pequenos círculos. Eles são responsáveis por registrar em *Log* o recebimento de eventos, recebimento de assinaturas e o envio de eventos aos assinantes. Cada uma dessas ocorrências é identificada com o ID único do evento ou assinatura, como já foi mencionado na seção anterior que detalhou os *Datasets*. Estes pontos de monitoração também registram os tempos de execução da solução para posterior análise de desempenho.

O componente mais importante é o **Analisador de *Logs***, que é responsável por ler os LOGs gerados e construir os conjuntos de dados utilizados para a extração das métricas. Além disso, ele é responsável por aplicar as métricas definidas e apresentar um relatório da execução que é utilizado na análise.

4.5 Análise dos Dados e Apresentação dos Resultados

Esta seção apresenta a análise dos dados resultantes dos experimentos realizados.

4.5.1 Eficácia da avaliação de similaridade média entre sentenças

Os gráficos apresentados nas figuras a seguir mostram os resultados da avaliação de eficácia da solução, quando configurada para utilizar o método de avaliação de similaridade entre sentenças baseada na média de similaridade entre palavras. Neste experimento, o método de avaliação de sentenças foi associado aos quatro diferentes métodos de avaliação de similaridade entre palavras, já apresentados no Capítulo 2. Foram feitas execuções do experimento variando a configuração de *Threshold*, de zero a cem porcento.

Como pode ser observado no gráfico da Figura 4.5, de maneira geral, quando a configuração utilizada determinava um *Threshold* baixo, a solução apresenta alta cobertura. A cobertura cai gradativamente à medida que o valor do *Threshold* é elevado. Este comportamento ocorre porque ao elevar o *Threshold* eleva-se o nível de similaridade mínima necessário para o Servidor de Eventos considerar dois nomes de campos como semelhantes. Dois campos podem ser considerados semelhantes em um experimento e deixar de ser em outro experimento com *Threshold* maior.

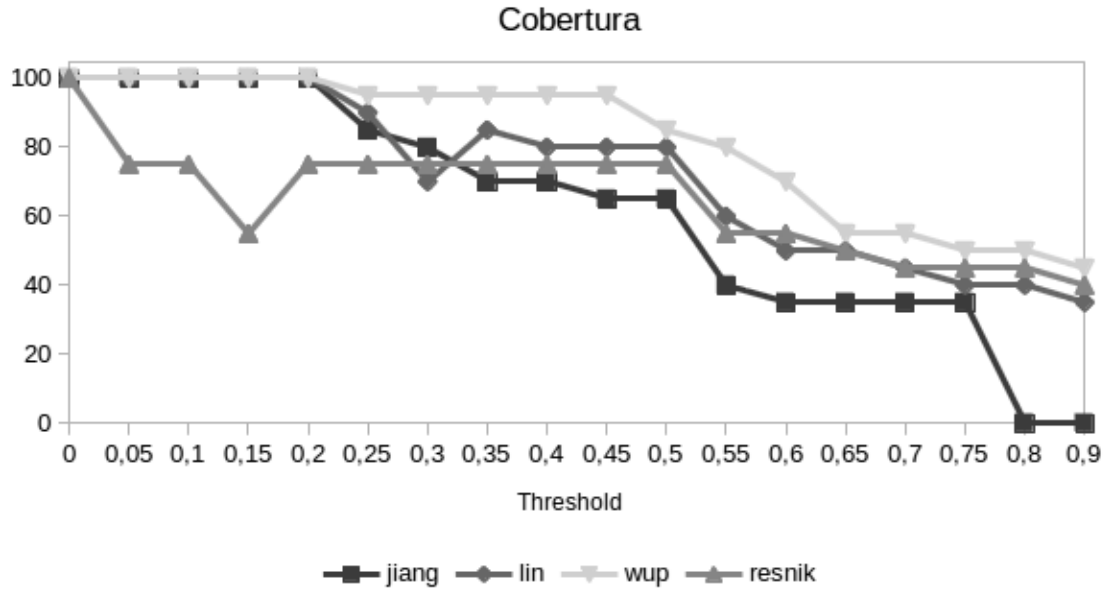

Figura 4.5: Cobertura - Similaridade Média entre sentenças

Já em relação à precisão (Figura 4.6), percebe-se que o aumento do *Threshold* não provoca variações consideráveis, apresentando apenas uma ligeira melhora e alcançando o valor máximo de 60% com o método de Jiang e Conrath.

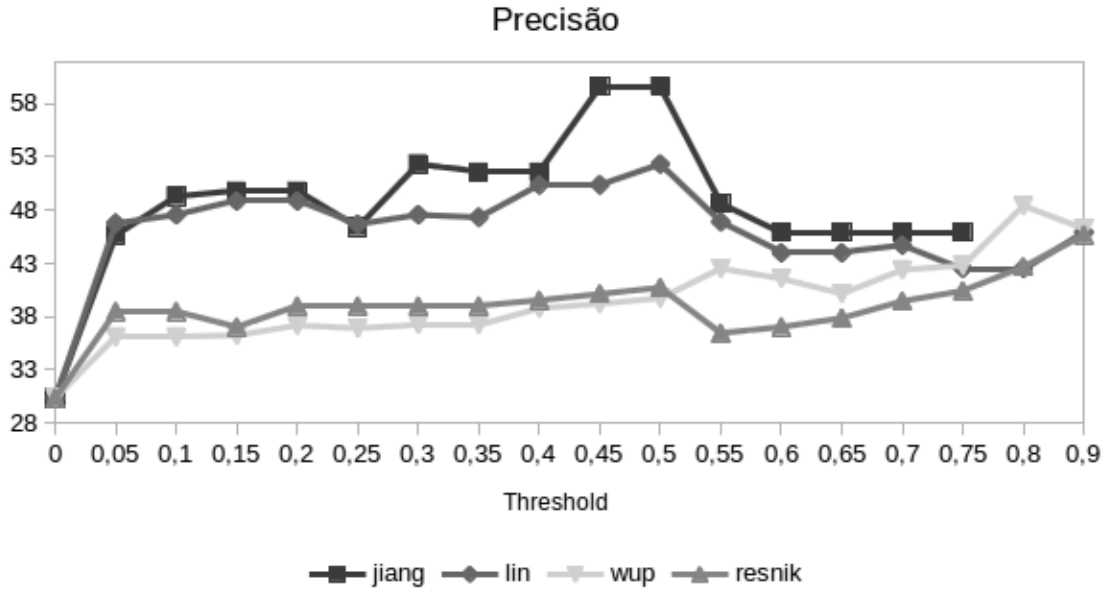

Figura 4.6: Precisão - Similaridade Média entre sentenças

Por fim, a Figura 4.7 mostra o comportamento da Medida-F. Esta medida, influenciada pela constância dos valores da precisão, apresenta comportamento estável para valores de *Threshold* inferiores a 50%, onde alcançou o máximo de 66%. A partir desse ponto, devido a influência do comportamento da Cobertura, apresenta um declínio, chegando aos valores mínimos com os métodos de Lin e Jiang & Conrath.

4.5.2 Eficácia da avaliação de similaridade entre sentenças de Feng

Nas figuras a seguir são apresentados os resultados da avaliação de eficácia da solução quando configurada para utilizar o método de Feng para medição de similaridade entre sentenças. A este método foram associados as estratégias Wu & Palmer e Jiang & Conrath de similaridade semântica entre palavras. Foram feitas execuções do experimento variando a configuração de *Threshold* entre 0% e 55%. Com *Threshold* acima de 55% a cobertura chegou a zero, impossibilitando computar as demais medidas.

Como ilustra a Figura 4.8, nestes experimentos a solução apresenta alta cobertura para valores baixos de *Threshold* para ambos os métodos, com ligeira vantagem para o método

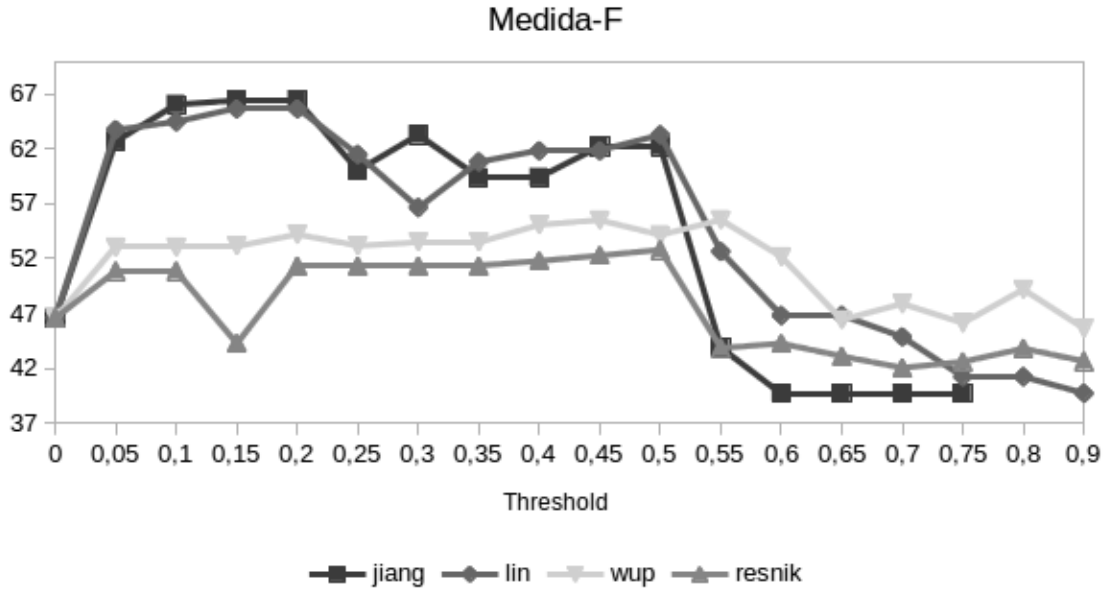

Figura 4.7: Medida-F - Similaridade Média entre sentenças

de Wu & Palmer. A cobertura apresenta um declínio à medida que o valor do *Threshold* é elevado, chegando aos seus valores mais baixos com *Threshold* a 55%. A partir desse valor de *Threshold* a cobertura apresenta valores muito baixos e, por isso, foram omitidos no gráfico.

O comportamento da precisão é apresentado na Figura 4.9. Para valores de *Thresholds* baixos, a precisão é inferior a 50% para ambos os métodos. Porém, à medida que se eleva o *Threshold*, os métodos apresentam uma melhoria acentuada na precisão, chegando a valores elevados de até 98% para o método de Jiang & Conrath no *Threshold* a 55%. A partir desse ponto a precisão, assim como a cobertura, apresenta valores irrelevantes, os quais foram omitidos no gráfico.

O gráfico da Figura 4.10 mostra que o método de Jiang & Conrath apresenta melhores resultados de Medida-F para a maioria dos valores de *Threshold*, chegando a atingir a marca de 63%.

4.5.3 Resultados da avaliação de desempenho

Os Gráficos apresentados a seguir mostram os resultados da avaliação do desempenho da solução diante da variação do número de assinantes, desde o cenário sem nenhum assinante conectado até situações mais extremas onde foram simulados até 50 assinan-

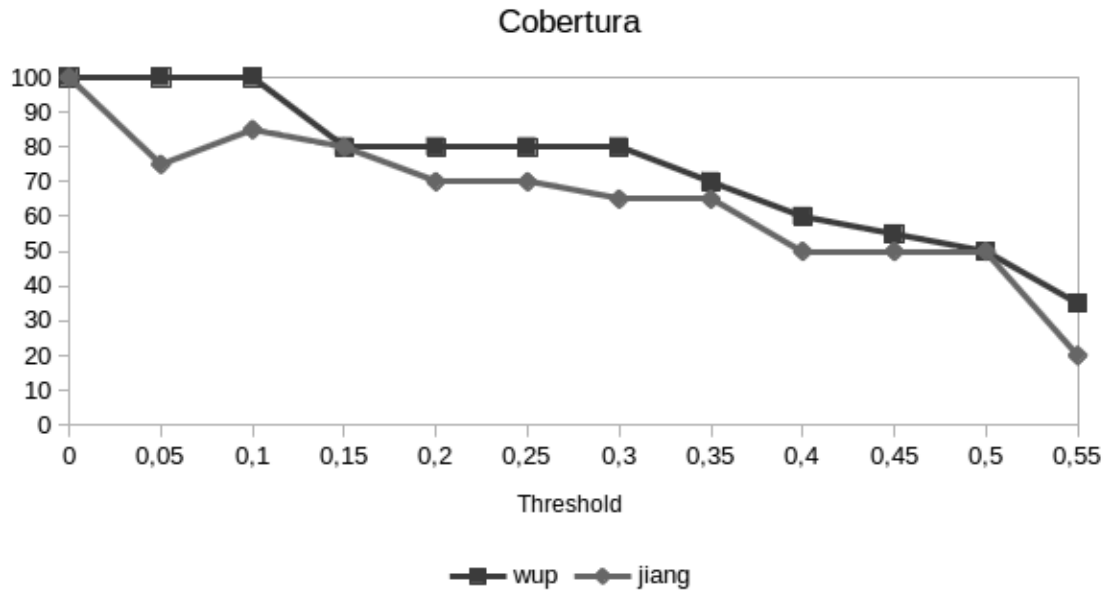

Figura 4.8: Cobertura - Método de similaridade entre sentenças de Feng

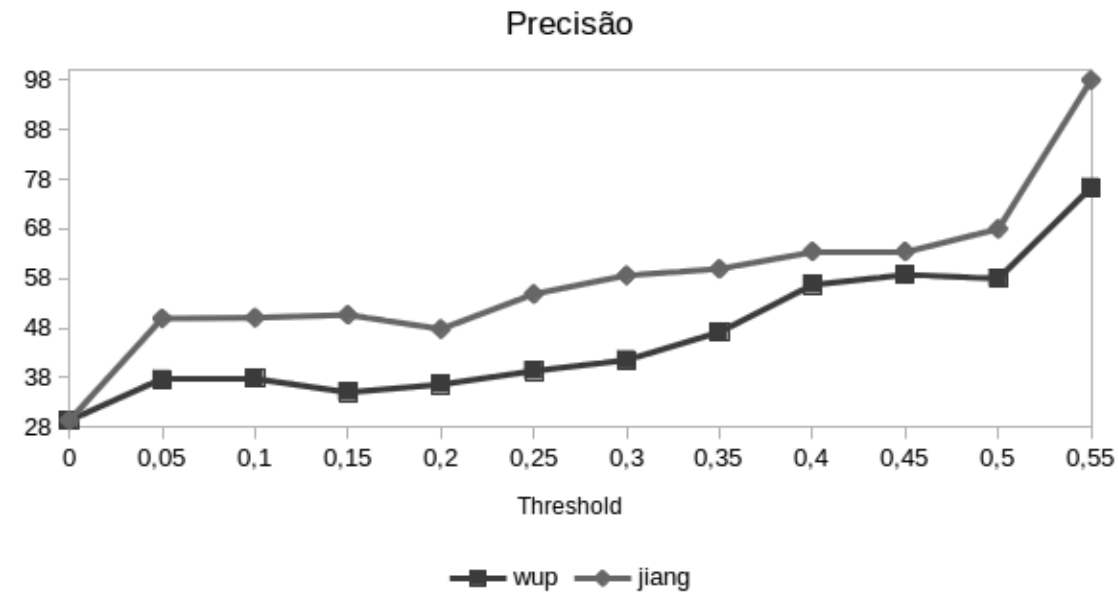

Figura 4.9: Precisão - Método de similaridade entre sentenças de Feng

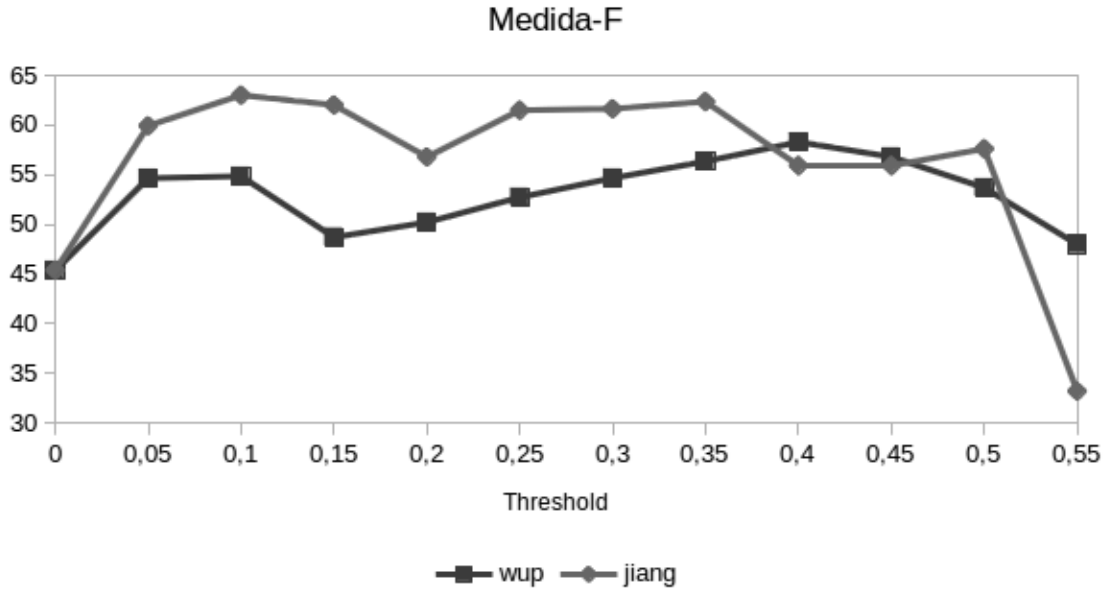

Figura 4.10: Medida-F - Método de similaridade entre sentenças de Feng

tes conectados simultaneamente. Neste experimento, a solução estava configurada para utilizar a estratégia de avaliação de similaridade que apresentou maior eficácia nos primeiros experimentos (maior *F-Score*). Nestes experimentos foram avaliadas as Taxas de Notificação (*Throughput*) e o Tempo Médio de Notificação.

A Figura 4.11 apresenta a taxa de notificação alcançada pela solução. O gráfico mostra que a solução apresenta boa taxa de notificação mesmo em cenários com muitos assinantes registrados. A solução consegue manter uma taxa de notificação por volta de 300 notificações de eventos por segundo, e esse valor não sofre grande degradação à medida que a quantidade de assinantes aumenta. O mesmo comportamento ocorre para o tempo médio de notificação (Figura 4.12), que fica sempre abaixo de 3 milissegundos por notificação evento.

É importante destacar a relevância do *cache* para este tipo de solução. Para isso, deve-se lembrar duas características inerentes aos dados processados neste tipo de solução: (i) as notificações de um mesmo publicador tendem a ser estruturadas em um único modelo de dados, ocorrendo variação apenas nos valores dos campos; (ii) as assinaturas que estão armazenadas no Servidor de Eventos não se modificam ao longo do tempo. Por isso, armazenar em *cache* as similaridades já computadas, entre os pares de campos das notificações e das assinaturas, evita que o Servidor de Eventos tenha que refazer essa parte do

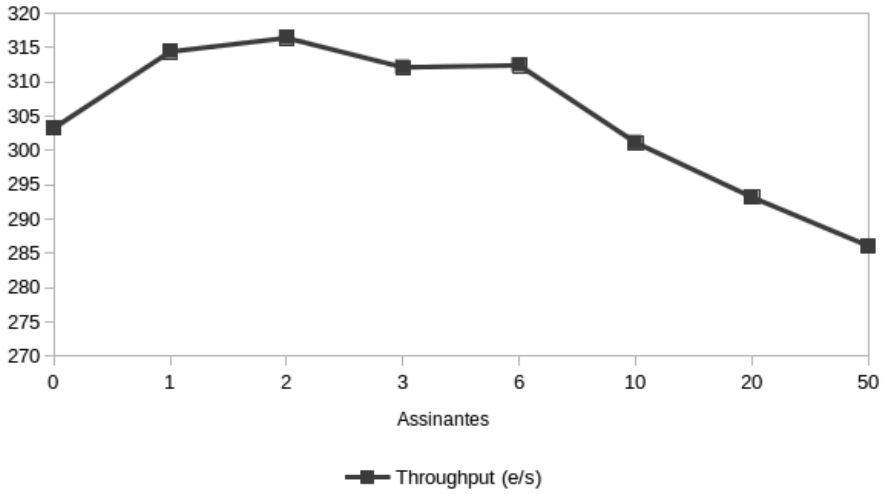

Figura 4.11: Taxa de notificação com cache ativado

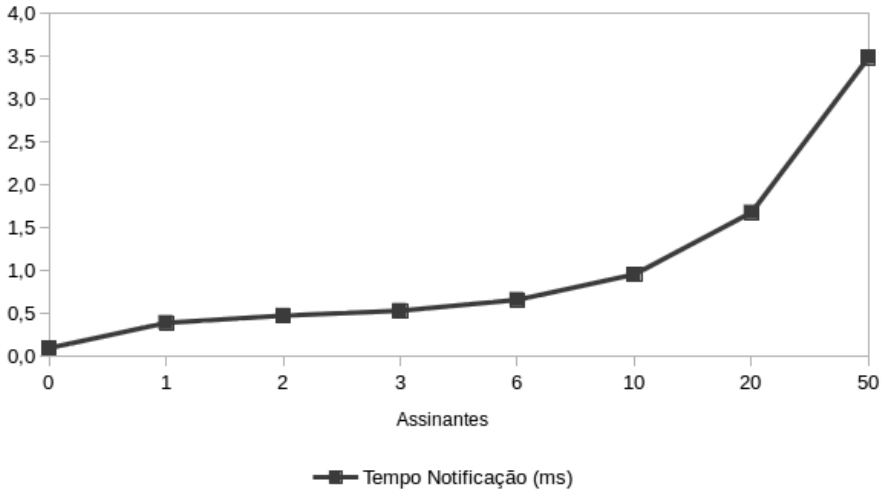

Figura 4.12: Tempo médio de notificação com cache ativado

processo à cada nova notificação, de um mesmo publicador, e resulta em uma redução do tempo de processamento do processo como um todo.

Os gráficos das Figuras 4.13 e 4.14 provam a importância do *cache* para a solução. Com o *cache* desativado, a taxa de notificação sofre forte degradação à medida que a quantidade de assinantes simultâneos é elevada, chegando ao mínimo de 50 notificações por segundo. O mesmo ocorre com o tempo médio de notificação, que no cenário com 50 assinantes chegou à marca de 380 milissegundos, até 100 vezes maior do que no mesmo cenário com *cache* ativado.

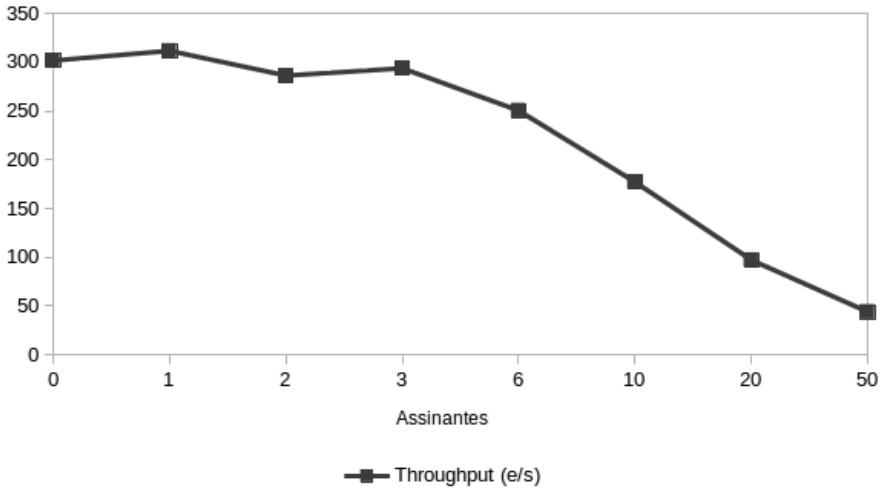

Figura 4.13: Taxa de notificação com cache desativado

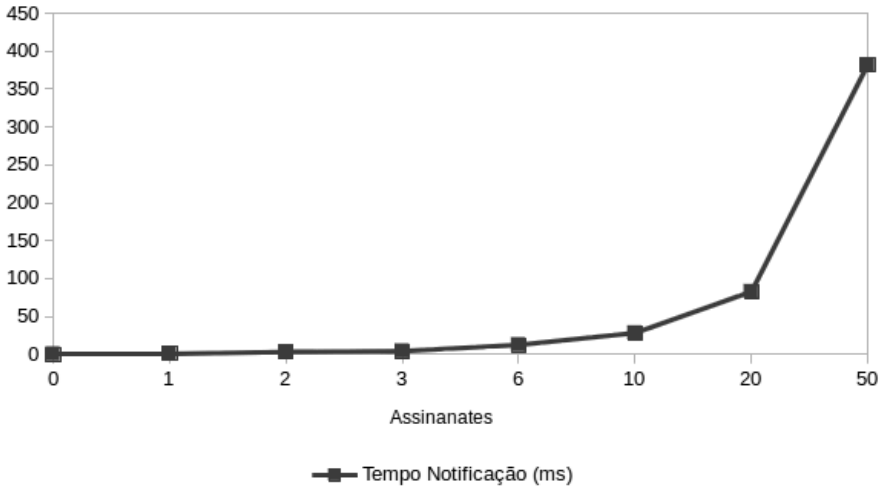

Figura 4.14: Tempo médio de notificação com cache desativado

4.6 Considerações Finais

Este trabalho apresentou uma solução com propósito de reduzir o impacto do acoplamento de estrutura de dados entre as notificações de eventos e assinaturas, fazendo com que, mesmo em situações em que ocorra heterogeneidade entre os nomes utilizados para estes atributos, o Servidor de Eventos seja capaz de identificar corretamente as assinaturas candidatas a recebimento das notificações de eventos. Este capítulo apresentou os resultados dos experimentos realizados com objetivo de avaliar o comportamento desta proposta diante de um cenário real.

A execução dos experimentos produziu uma amostra de dados relativamente grande e o uso das análises gráficas foi um recurso que facilitou a análise.

As avaliações dos experimentos mostraram que, de maneira geral, a solução obteve bom desempenho e atingiu a taxas de notificação acima de 300 eventos por segundo e um tempo médio de notificação abaixo de 4 milissegundos.

Além disso, a combinação do método de avaliação de similaridade entre palavras de Jiang & Conrath e a abordagem de Feng para avaliar a similaridade semântica entre sentenças obteve uma medida-F de até 63%. Esta combinação de métodos, associada à configuração de *Threshold* em 55%, conseguiu ainda uma precisão de 98%.

Apesar destes resultados serem interessantes, não é possível tirar conclusões confiáveis quanto à eficácia da solução, pois não houve uma base representativa para estabelecer comparações. Porém, também cabe ressaltar que o intuito deste experimento não foi estabelecer comparações, mas demonstrar a viabilidade e aplicabilidade da arquitetura proposta diante de um cenário real.

5. Conclusão

Este capítulo apresenta a conclusão deste trabalho, suas contribuições, bem como as limitações da abordagem e trabalhos futuros.

5.1 Considerações finais

Este trabalho propôs um modelo arquitetural baseado no padrão *Publish/Subscribe* para integração de sistemas corporativos, que permita a redução do acoplamento de modelo de dados, entre assinaturas e notificações de eventos, através da avaliação de similaridade semântica entre os nomes dos atributos. O modelo proposto permite que componentes que implementem diferentes métodos e estratégias de avaliação de similaridade pudessem ser conectados à arquitetura a depender do cenário de aplicação.

A partir do modelo arquitetural proposto, foi implementada uma solução de referência que utilizou uma variedade de métodos de avaliação para demonstrar o poder da solução. O Capítulo 4 demonstrou, a partir da análise dos cenários executados, que os resultados obtidos pela abordagem proposta são suscetíveis à configuração da solução. Os diferentes métodos de avaliação de similaridade e as diferentes configurações de *Threshold*, aplicados na solução de referência, produzem resultados interessantes.

Os valores observados para Cobertura, Precisão e Medida-F demonstram que o componente Servidor de Eventos foi capaz de determinar com relativa eficácia os assinantes interessados nos eventos publicados. Contudo, estes resultados, de certa forma, restringem os cenários de aplicação possíveis para a solução. Como, em nenhuma das configurações experimentadas, os valores de precisão e cobertura atingiram 100%, a solução não pode

ser aplicada em cenários com estes níveis de exigência e que não admitam a perda de alguns eventos. Ainda assim, existem diversos cenários onde é possível tirar proveito das vantagens e características apresentadas pela solução, entre eles é possível destacar a integração de dispositivos eletrônicos de IoT, análises com *Business Intelligence* em tempo real e Big Data, ou outros cenários que apresentem as seguintes características:

- grande volatilidade dos sistemas publicadores e/ou grande variação nos esquemas dos dados publicados;
- a detecção automática dos assinantes interessados represente um ganho importante em detrimento à perda de eventos;
- admissibilidade de consistência eventual dos dados;
- tratamento de grandes volumes de dados;
- priorização do processamento em tempo real em detrimento à perda de evento.,

Por fim, diante do exposto e considerando o objetivo deste trabalho, é possível concluir que a arquitetura proposta atende aos requisitos estabelecidos.

5.2 Contribuições

As principais contribuições deste trabalho são o modelo arquitetural proposto e a solução de referências implementada. Pode-se considerar ainda como contribuição, mesmo que de forma secundária, as análises dos resultados produzidos nos experimentos, que demonstraram o comportamento da solução com as diferentes combinações de métodos de similaridade. Esta análise pode servir de insumo para a tomada de decisão de como configurar a solução a depender do cenário de aplicação da arquitetura.

5.3 Limitações

A solução proposta mostrou-se eficaz no que diz respeito à capacidade de identificar os assinantes interessados nos eventos publicados, mesmo em situações onde ocorre heterogeneidade. Porém, este trabalho possui a limitação de, pela ausência de outros trabalhos

semelhantes, não ter sido comparado e com isso pudesse ser exposto o quão boa é a solução em relação a outras. Ou seja, temos valores de precisão, cobertura, Medida-F e outros valores a respeito do desempenho, porem não foi possível mensurar o quão interessante são estes valores para a literatura.

5.4 Trabalhos futuros

A solução proposta é capaz de identificar corretamente os assinantes interessados nos eventos publicados. Porém, após este processo o evento é enviado em seu esquema original para o assinante, ou seja, em um formato que muitas vezes não é o utilizado pelo assinante. Em um cenário de aplicação real deste tipo de arquitetura, isto exigiria que o assinante tivesse um esforço extra de converter o evento para o modelo esperado. Por isto, esta abordagem poderia ser evoluída de maneira que fosse possível adicionar um componente capaz de realizar a transformação do evento original em um evento mais próximo do modelo esperado pelo assinante de destino. O insumo para esse mecanismo são os registros das similaridades identificadas pela solução, o esforço adicional seria somente o de converter o modelo e trocar os nomes dos atributos para os nomes esperados pelo assinante.

Limitações de tempo impossibilitaram que outros métodos de similaridade entre sentenças e palavras pudessem ser implementados e experimentados. Em oportunidades futuras este estudo poderia ser realizado para complementar as análises apresentadas neste trabalho, podendo, eventualmente, apresentar resultados ainda melhores em relação à eficácia.

Outra evolução interessante seria adicionar um componente de interface gráfica que permita a configuração da solução de maneira mais amigável. Na proposta atual as diversas configurações da solução são feitas através de arquivos de configuração. Esta interface gráfica poderia permitir, inclusive, que o administrador da solução configurasse os valores de similaridades pré-computados que são utilizados pelos componentes da solução.

Referências Bibliográficas

[1] I. Atoum, A. Otoom, and N. Kulathuramaiyer. Article: A comprehensive comparative study of word and sentence similarity measures. *International Journal of Computer Applications*, 135(1):10--17, February 2016. Published by Foundation of Computer Science (FCS), NY, USA.

[2] Z. Bellahsene, A. Bonifati, F. Duchateau, and Y. Velegrakis. On evaluating schema matching and mapping. In *Schema matching and mapping*, pages 253--291. Springer, 2011.

[3] K. Chandy and W. Schulte. *Event Processing: Designing IT Systems for Agile Companies*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2010.

[4] E. Curry, S. Hasan, and S. O'Riain. Enterprise energy management using a linked dataspace for energy intelligence. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*, pages 1--6. IEEE, 2012.

[5] V. de Paiva, A. Rademaker, and G. de Melo. Openwordnet-pt: An open Brazilian Wordnet for reasoning. In *Proceedings of COLING 2012: Demonstration Papers*, pages 353--360, Mumbai, India, Dec. 2012. The COLING 2012 Organizing Committee. Published also as Techreport http://hdl.handle.net/10438/10274.

[6] J. Didion and B. Walenz. Jwnl (java wordnet library), 2004.

[7] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114--131, June 2003.

[8] J. Feng, Y.-M. Zhou, and T. Martin. Sentence similarity based on relevance. In *Proceedings of IPMU*, volume 8, page 833, 2008.

[9] M. M. Gomes, W. Beltrame, and D. Cury. Automatic construction of brazilian portuguese wordnet. In *Proceedings of X National Meeting on Artificial and Computational Intelligence, ENIAC*, 2013.

[10] H. Gonçalo Oliveira. CONTO.PT: Groundwork for the Automatic Creation of a Fuzzy Portuguese Wordnet. In *Proceedings of 12th International Conference on Computational Processing of the Portuguese Language (PROPOR 2016)*, volume 9727 of *LNAI*, pages 283--295, Tomar, Portugal, July 2016. Springer.

[11] S. Hasan, S. O'Riain, and E. Curry. Approximate semantic matching of heterogeneous events. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 252--263. ACM, 2012.

[12] E. International. *The JSON Data Interchange Format*, 2013.

[13] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[14] D. Lin. Principle-based parsing without overgeneration. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 112--120. Association for Computational Linguistics, 1993.

[15] F. Lin and K. Sandkuhl. A survey of exploiting wordnet in ontology matching. In *IFIP International Conference on Artificial Intelligence in Theory and Practice*, pages 341--350. Springer, 2008.

[16] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55--60, 2014.

[17] S. Microsystems. *Sun Java System Message Queue 4.1 Developer's Guide for Java Clients*. 2007.

[18] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39--41, 1995.

[19] M. Murth, D. Winkler, S. Biffl, E. Kühn, and T. Moser. Performance testing of semantic publish/subscribe systems. In *OTM Confederated International Conferences"On the Move to Meaningful Internet Systems*

, pages 45--46. Springer, 2010.

[20] E. G. Petrakis, G. Varelas, A. Hliaoutakis, and P. Raftopoulou. Design and evaluation of semantic similarity measures for concepts stemming from the same or different ontologies. In *4th Workshop on Multimedia Semantics (WMS'06)*, pages 44--52, 2006.

[21] A. F. Pimenta Jr, L. G. Azevedo, and F. M. Santoro. Arquitetura publish/subscribe baseada em semântica. *X Workshop de Teses e Dissertações em Sistemas de Informação (WTDSI)*, pages 37 -- 39, 2017.

[22] J. Recker. *Scientific research in information systems: a beginner's guide*. Springer Science & Business Media, 2012.

[23] P. Resnik et al. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *J. Artif. Intell. Res.(JAIR)*, 11:95--130, 1999.

[24] L. Sanchez, J. A. Galache, V. Gutierrez, J. M. Hernandez, J. Bernat, A. Gluhak, and T. Garcia. Smartsantander: The meeting point between future internet research and experimentation and the smart cities. In *Future Network & Mobile Summit (FutureNetw), 2011*, pages 1--8. IEEE, 2011.

[25] H. Shima. Wordnet similarity for java. `https://code.google.com/archive/p/ws4j`. Acessado em 20/08/2017.

[26] J. Skovronski and K. Chiu. Ontology based publish subscribe framework. In *iiWAS'2006 - The Eighth International Conference on Information Integration and Web-based Applications Services, 4-6 December 2006, Yogyakarta, Indonesia*, pages 49--58, 2006.

[27] T. Slimani. Description and evaluation of semantic similarity measures approaches. *International Journal of Computer Applications*, 80:25--33, 2013.

[28] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133--138. Association for Computational Linguistics, 1994.

[29] L. Zeng and H. Lei. A semantic publish/subscribe system. In *E-Commerce Technology for Dynamic E-Business, 2004. IEEE International Conference on*, pages 32--39. IEEE, 2004.

6. Apêndice A - Resultados Eficácia

Este apêndice apresenta os dados coletados nos experimentos realizados e apresentados no Capítulo 4 que se referem à eficácia da solução. Os dados são apresentados em formato tabular, onde cada tabela representa os resultados obtidos com cada combinação de método de similaridade entre palavras e entre sentenças.

Tabela 6.1: Similaridade Média - Método Jiang & Conrath

Threshold	**Precisão**	**Cobertura**	**F-Score**
0	30,364	100	46,5834125986
0,05	45,667	100	62,7005430194
0,1	49,303	100	66,0442188034
0,15	49,769	100	66,4610166323
0,2	49,769	100	66,4610166323
0,25	46,419	85	60,0463403313
0,3	52,333	80	63,2743155524
0,35	51,558	70	59,3800490301
0,4	51,558	70	59,3800490301
0,45	59,625	65	62,1965897693
0,5	59,625	65	62,1965897693
0,55	48,63	40	43,8948437324
0,6	45,923	35	39,7243058216
0,65	45,923	35	39,7243058216
0,7	45,923	35	39,7243058216
0,75	45,923	35	39,7243058216
0,8	0	0	0
0,9	0	0	0

Tabela 6.2: Similaridade Média - Método Lin

Threshold	**Precisão**	**Cobertura**	**F-Score**
0	30,364	100	46,5834125986
0,05	46,778	100	63,7397975173
0,1	47,571	100	64,4720168597
0,15	48,851	100	65,6374495301
0,2	48,851	100	65,6374495301
0,25	46,692	90	61,4853831973
0,3	47,571	70	56,6461117112
0,35	47,333	85	60,8057702916
0,4	50,373	80	61,8201621501
0,45	50,373	80	61,8201621501
0,5	52,333	80	63,2743155524
0,55	46,907	60	52,6517440392
0,6	44	50	46,8085106383
0,65	44	50	46,8085106383
0,7	44,714	45	44,8565441291
0,75	42,529	40	41,2257509482
0,8	42,529	40	41,2257509482
0,9	45,923	35	39,7243058216

Tabela 6.3: Similaridade Média - Wu & Palmer

Threshold	**Precisão**	**Cobertura**	**F-Score**
0	30,364	100	46,5834125986
0,05	36,094	100	53,0427498641
0,1	36,094	100	53,0427498641
0,15	36,241	100	53,2013123803
0,2	37,182	100	54,208278054
0,25	36,925	95	53,1798370286
0,3	37,269	95	53,5356735138
0,35	37,269	95	53,5356735138
0,4	38,792	95	55,0890935183
0,45	39,213	95	55,5122827148
0,5	39,732	85	54,1516210756
0,55	42,529	80	55,5349345869
0,6	41,581	70	52,1714270351
0,65	40,154	55	46,4188578515
0,7	42,404	55	47,8875610858
0,75	42,81	50	46,1264949898
0,8	48,412	50	49,1931878226
0,9	46,273	45	45,6276226266

Tabela 6.4: Similaridade Média - Resnik

Threshold	**Precisão**	**Cobertura**	**F-Score**
0	30,364	100	46,5834125986
0,05	38,481	75	50,8644618923
0,1	38,481	75	50,8644618923
0,15	37,033	55	44,2627101149
0,2	39	75	51,3157894737
0,25	39	75	51,3157894737
0,3	39	75	51,3157894737
0,35	39	75	51,3157894737
0,4	39,548	75	51,7878967769
0,45	40,127	75	52,2818278944
0,5	40,739	75	52,79853809
0,55	36,46	55	43,8508637656
0,6	37,033	55	44,2627101149
0,65	37,868	50	43,0964628761
0,7	39,455	45	42,0454680007
0,75	40,429	45	42,5922110759
0,8	42,684	45	43,8114137129
0,9	45,667	40	42,646059743

Tabela 6.5: Similaridade Feng - Wu & Palmer

Threshold	Precisão	Cobertura	F-Score
0	29,364	100	45,397483071
0,05	37,608	100	54,6596128132
0,1	37,802	100	54,8642254829
0,15	35,021	80	48,7159736048
0,2	36,605	80	50,2276917799
0,25	39,333	80	52,7371305506
0,3	41,529	80	54,6753449794
0,35	47,167	70	56,3587016822
0,4	56,71	60	58,3086282238
0,45	58,741	55	56,8089782928
0,5	58	50	53,7037037037
0,55	76,333	35	47,9939460897
0,6	68	5	9,3150684932

Tabela 6.6: Similaridade Feng - Jiang & Conrath

Threshold	Precisão	Cobertura	F-Score
0	29,364	100	45,397483071
0,05	49,915	75	59,9387583557
0,1	50,075	85	63,0223949658
0,15	50,653	80	62,0305695239
0,2	47,787	70	56,798967628
0,25	54,842	70	61,5007769821
0,3	58,625	65	61,6481294237
0,35	59,935	65	62,3648297115
0,4	63,455	50	55,9296637433
0,45	63,455	50	55,9296637433
0,5	68	50	57,6271186441
0,55	98	20	33,2203389831
0,6	0	0	0

7. Apêndice B - Resultados Desempenho

Este apêndice apresenta os dados coletados nos experimentos realizados e apresentados no Capítulo 4 que se referem ao desempenho da solução. Os dados estão em formato tabular, onde são apresentados os resultados alcançados pela solução, em relação à taxa de notificação e tempo médio de notificação, à medida que foram variadas as condições experimentais. Foram alteradas as quantidades de assinantes conectados e o uso do cache, ativado e desativado.

Tabela 7.1: Desempenho com Cache Ativado

Qtd. Assinantes	Taxa de Notificação (eventos/seg.)	Tempo Médio de Notificação
0	303,303	0,1
1	314,416	0,4
2	316,427	0,5
3	312,133	0,5
6	312,431	0,7
10	301,160	1,0
20	293,212	1,7
50	286,100	3,5

Tabela 7.2: Desempenho com Cache Desativado

Qtd. Assinantes	Taxa de Notificação (eventos/seg.)	Tempo Médio de Notificação
0	301,714	0,1
1	311,744	1,1
2	286,316	3,0
3	293,796	4,0
6	250,440	12,2
10	177,271	28,0
20	97,250	82,6
50	44,068	382,0