

Gerenciamento de Regras de Qualidade em Cadeias Produtivas

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Maurício Augusto Figueiredo e aprovada pela Banca Examinadora.

Campinas, 29 de Maio de 2009.

Profa. Dra. Claudia Bauzer Medeiros
Instituto de Computação - UNICAMP
(Orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Substitua pela ficha catalográfica

(Esta página deve ser o verso da página anterior mesmo no caso em que não se imprime frente e verso, i.é., até 100 páginas.)

Substitua pela folha com as assinaturas da banca

Gerenciamento de Regras de Qualidade em Cadeias Produtivas

Maurício Augusto Figueiredo¹

Março de 2009

Banca Examinadora:

- Profa. Dra. Claudia Bauzer Medeiros
Instituto de Computação - UNICAMP (Orientadora)
- Prof. Dr. Edmundo Roberto Mauro Madeira
Instituto de Computação - UNICAMP
- Prof. Dr. André Santanchè
Universidade Salvador - UNIFACS
- Prof. Dr. Ricardo da Silva Torres
Instituto de Computação - UNICAMP
- Prof. Dr. Rubens Augusto Camargo Lamparelli
CEPAGRI - UNICAMP

¹Suporte financeiro de: FAPESP (Processo 07/53610-0), além dos projetos Agroflow e WEBMAPS II do CNPq.

© Maurício Augusto Figueiredo, 2009.
Todos os direitos reservados.

Resumo

Cadeias produtivas têm se tornado cada vez mais dependentes de sistemas computacionais. Além dos desafios científicos, há várias conseqüências econômicas. Esta dissertação trata de mecanismos de gerenciamento de regras que especificam a qualidade de produtos em cadeias produtivas sob dois aspectos: (i) a especificação e armazenamento destas regras e (ii) a análise dos eventos ocorridos na cadeia face a tais restrições.

A dissertação parte de um modelo de rastreabilidade para cadeias produtivas agrícolas desenvolvido na UNICAMP. As regras de qualidade gerenciadas definem condições atribuídas a produtos de forma que eles possam ser consumidos. A verificação de regras é baseada na análise de variáveis consideradas críticas para a garantia de qualidade, que são monitoradas por sensores. Portanto, esta pesquisa combina trabalhos em gerenciamento de dados de sensores, bancos de dados ativos e restrições de integridade.

As principais contribuições são: um estudo detalhado sobre rastreabilidade associada a regras de qualidade, um modelo para gerenciar a especificação, aplicação e análise dessas regras, a especificação e implementação de algoritmos que transformam as regras em código executável e um protótipo para validar a arquitetura. O protótipo é baseado em serviços Web e disseminação de eventos. Os estudos de caso são baseados em problemas na área de agricultura.

Abstract

Supply chains have become increasingly dependent on information systems. Besides posing scientific challenges, there are many correlated economical issues. This thesis deals with mechanisms to manage rules that specify the quality of products in supply chains, under two aspects: (i) the specification and storage of those rules and (ii) the analysis of the events in this environment, under quality restrictions.

This research adopts a traceability model for agricultural supply chains developed at UNICAMP. The quality rules considered define conditions that products must meet so that they can be consumed. Rule checking is based on analysis of variables considered critical for quality assessment, which are monitored by sensors. Hence, this research combines work in sensor data management, active databases and integrity constraints.

The main contributions are: a detailed study on traceability associated to quality rules; a model to manage the specification, application and analysis of those rules; the specification and implementation of algorithms that transform the rules into executable database triggers; and a prototype to validate the architecture, based on Web Services and event dissemination. Case studies are based on agricultural challenges.

Agradecimentos

À Professora Claudia Bauzer Medeiros por guiar-me sabiamente durante vários anos de trabalho.

À minha família e amigos, pela confiança, motivação e apoio prestados durante a realização deste Mestrado.

À todas as pessoas que passaram pelo LIS no período do meu mestrado, pelas contribuições e pela amizade.

Às entidades financiadoras deste projeto: FAPESP (Processo 07/53610-0), além dos projetos Agroflow e WEBMAPS II do CNPq.

Sumário

Resumo	vi
Abstract	vii
Agradecimentos	viii
1 Introdução	1
1.1 Visão Geral	1
2 Trabalhos Correlatos e Conceitos Básicos	3
2.1 Cadeias Produtivas e o Modelo de Bacarin	3
2.2 Rastreabilidade	5
2.3 O Modelo de Kondo	8
2.4 Serviços Web	11
2.5 Bancos de Dados Ativos	13
2.6 Sistemas Baseados em Eventos	15
2.7 Redes de Sensores	17
2.8 Conclusões	18
3 O Modelo de Gerenciamento de Qualidade	19
3.1 Visão Geral	19
3.2 Premissas	21
3.3 Arquitetura	22
3.4 Modelo de Gerenciamento Global	25
3.5 O Modelo de Persistência de Dados	27
3.5.1 Repositório de Regras	28
3.5.2 Repositório de Sensores	30
3.5.3 Sumário de Regras	31
3.5.4 Sumário de Sensores	33
3.5.5 Adaptações ao Modelo de Kondo	34

3.6	Modelo de Gerenciamento de Regras	35
3.6.1	Lógica do gerenciamento de regras	35
3.6.2	Conversão de Regras	37
3.6.3	Deteção de violações	40
3.7	Modelo de Disseminação de Eventos	41
3.8	Conclusões	43
4	Aspectos de implementação	44
4.1	Tecnologias Utilizadas	44
4.2	Repositórios de dados	45
4.3	Serviços Web	48
4.4	Módulo de Gerenciamento de Regras	50
4.4.1	Módulo de Controle de Regras	50
4.4.2	Módulo de Controle de Notificações	53
4.5	Módulo de Disseminação de Eventos	55
4.5.1	Serviço de Publicação	55
4.5.2	Serviço de Recebimento	57
4.6	Módulo de Gerenciamento Global	58
4.7	Conclusões	58
5	Conclusões e Trabalhos Futuros	59
5.1	Conclusões	59
5.2	Extensões	60
5.2.1	Complexidade da especificação das regras	60
5.2.2	Uso de ontologias	61
5.2.3	Integração entre regras e workflows	61
5.2.4	Outros tipos de regras de qualidade	61
	Bibliografia	62

Lista de Tabelas

2.1	Tabela comparativa de trabalhos em rastreabilidade.	8
3.1	Repositório de Regras.	29
3.2	Repositório de Sensores.	30
3.3	Sumário de Regras.	31
3.4	Sumário de Sensores.	33
3.5	Exemplo de chave de identificação.	35
4.1	Tabela de operações do <i>DataAccessControler</i>	48
4.2	Exemplo de regra a ser processada pelo Módulo de Controle de Regras. . .	51
4.3	Exemplo de notificação a ser processada pelo Módulo de Controle de No- tificações.	54

Lista de Figuras

2.1	Cadeia Produtiva do Leite [2]	5
2.2	Transporte T2 - Cadeia Encapsulada [35]	5
2.3	Arquitetura de Kondo [35].	9
2.4	Seqüência de atividades em uma arquitetura de serviços Web. Modificado de [28].	12
2.5	Componentes de um Banco de Dados Ativo [11].	14
2.6	Modelo de interação de um Sistema Baseado em Eventos	16
2.7	Espectro de dispositivos sensores [27]	17
3.1	Visão geral da proposta.	20
3.2	Arquitetura proposta.	23
3.3	Diagrama de Seqüência de Autorização de Acesso a um Repositório.	26
3.4	Repositórios de dados.	27
3.5	Arquitetura do servidor de dados do modelo.	28
3.6	Diagrama de Seqüência de atividades da conversão de regras.	38
3.7	Diagrama de conversão de regras em linguagem procedural.	38
3.8	Diagrama de Seqüência de Detecção de Violações.	40
3.9	Diagrama de Seqüência: Registro de Notificações.	42
3.10	Diagrama de Seqüência: Subscrição e distribuição.	42
4.1	Tecnologias utilizadas.	45
4.2	Esquema básico de comunicação do SPTracer via serviços Web	50
4.3	Interface de consulta a dados de repositórios.	58

Capítulo 1

Introdução

1.1 Visão Geral

É cada vez mais evidente a dependência que cadeias produtivas têm do controle proporcionado pelas ferramentas computacionais. A automatização de processos de produção e serviços é uma tendência adotada universalmente. Seus objetivos compreendem a otimização de etapas de produção, aumento dos padrões de qualidade, aprimoramento dos processos de gestão, melhoria na utilização de recursos e, conseqüentemente, ganho de desempenho e eficiência das cadeias como um todo.

Uma cadeia produtiva é composta por um conjunto de componentes interativos do qual fazem parte sistemas produtivos, fornecedores de insumos e serviços, indústrias de processamento e transformação, agentes de distribuição e comercialização, além dos consumidores finais[15]. Em Computação, cadeias produtivas são estudadas sob vários aspectos, como, por exemplo, preocupação com os algoritmos adotados, logística, estratégias de armazenamento e sistemas distribuídos. Um problema em aberto é a questão de rastreabilidade, motivada pela necessidade real de garantia de qualidade de um produto. A rastreabilidade refere-se à habilidade de descrever e seguir a vida de um elemento conceitual ou físico. Pode ser realizada a partir da origem, desenvolvimento até a utilização dos produtos e também pode ocorrer em ordem inversa [41].

Esta questão deu origem a pesquisas que buscam mapear a lógica de funcionamento de cadeias produtivas reais em ambientes computacionais. A meta principal é proporcionar uma maior homogeneização e padronização das informações geradas pelos processos, serviços e produtos envolvidos. O objetivo final deste esforço é possibilitar que estas informações possam ser interpretadas de maneira correta, sintática e semanticamente, independentemente da localização e padrões adotados nos sistemas em que são utilizadas.

Em particular, uma série de estudos visa a assegurar a qualidade de produtos e processos. A própria definição de qualidade é um problema: como definí-la, como garantí-la

e solucionar casos em que é violada. Critérios de qualidade de um produto envolvem itens como: se a técnica de produção de uma de suas matérias-primas é potencialmente prejudicial ao ambiente, se foi produzido por uma indústria ou num país que utiliza mão-de-obra infantil ou mal-remunerada, se contém ingredientes geneticamente modificados, entre outros. Além disso, podem ocorrer, por exemplo, vazamentos químicos ou radioativos. Neste caso, seria necessário remover do mercado todos aqueles produtos que direta ou indiretamente tenham sido contaminados.

A rastreabilidade na indústria de alimentos é mencionada como preservação da identidade e está tornando-se um importante assunto com a disseminação de alimentos geneticamente modificados [50].

Esta dissertação contribui para a solução dos problemas apontados, propondo e implementando mecanismos para o gerenciamento e monitoramento de regras de qualidade em processos e serviços de cadeias produtivas. A hipótese básica é que especialistas do domínio tenham especificado todos os critérios a serem aplicados, correspondendo às regras que definem qualidade. Para o desenvolvimento da pesquisa, foram utilizados como base trabalhos desenvolvidos na UNICAMP em rastreabilidade, serviços Web e documentação de processos, em particular a proposta de Bacarin [2] e o modelo de rastreabilidade proposto por Kondo [35]. O trabalho envolve pesquisa em serviços Web, sensores, bancos de dados ativos e sistemas baseados em eventos, provendo mecanismos que possibilitem análises espaço-temporais de violação de restrições pré-estabelecidas.

As principais contribuições são:

- Um estudo detalhado referente ao uso de regras de qualidade associado à rastreabilidade;
- Proposta de um modelo capaz de gerenciar a elaboração, aplicação e análise dessas regras de qualidade, com auditoria dos eventos registrados;
- A especificação e desenvolvimento de algoritmos que transformem tais regras em gatilhos executados pelo SGBD;
- Implementação e validação da arquitetura proposta para o modelo usando simulações.

Resultados deste trabalho foram publicados no VII Workshop de Teses e Dissertações em Banco de Dados do XXIII Simpósio Brasileiro de Banco de Dados - 2008 [20].

A dissertação está organizada da seguinte forma. O Capítulo 2 apresenta os conceitos e trabalhos relacionados. O Capítulo 3 discorre sobre o modelo gerenciamento de qualidade proposto. O Capítulo 4 descreve o protótipo implementado. Finalmente, o Capítulo 5 conclui a dissertação.

Capítulo 2

Trabalhos Correlatos e Conceitos Básicos

Os principais temas necessários como embasamento para este trabalho são cadeias produtivas agrícolas e o Modelo de Bacarin (seção 2.1), Rastreabilidade (seção 2.2) e o Modelo de Kondo (seção 2.3), serviços Web (seção 2.4), bancos de dados ativos (seção 2.5) e sistemas baseados em eventos (seção 2.6). A seção 2.7 explora técnicas de utilização de redes de sensores para a obtenção de dados.

2.1 Cadeias Produtivas e o Modelo de Bacarin

A dissertação se concentra em cadeias produtivas agrícolas, tendo em vista a experiência já existente no Laboratório de Sistemas de Informação (LIS) do IC neste domínio. A vantagem adicional é a proximidade com especialistas do domínio, dentro da própria UNICAMP.

Uma cadeia produtiva agrícola compreende todas as atividades que ocorrem desde a produção agrícola primária até a chegada do alimento processado na mesa do consumidor final. Cada um dos componentes da cadeia é autônomo, devido ao grau de liberdade que os bancos de dados possuem em tomar decisões e para definir quando a comunicação com outros componentes deve ser estabelecida ou não [15].

A dissertação utiliza o modelo de cadeias agrícolas de Bacarin [2]. Este modelo, ilustrado na figura 2.1, é composto pelos seguintes elementos básicos:

- **Atores:** são agentes externos de software, pessoas e empresas que interagem com a cadeia, podendo estar direta ou indiretamente relacionados à execução de atividades. Alguns exemplos são agências certificadoras, reguladoras, advogados e negociadores;

- **Produção:** elemento que encapsula algum processo produtivo. A partir de entradas de insumos obtidas de outros componentes, produz um produto para seguir pela cadeia produtiva;
- **Armazenamento:** elemento que armazena produtos ou matérias-primas;
- **Transporte:** elemento que faz o deslocamento de produtos e matérias-primas entre componentes de produção e/ou armazenamento.

Além disso, a dinâmica da cadeia é modelada a partir dos seguintes elementos:

- **Contratos:** são declarações de obrigações e autorizações mútuas que refletem o acordo entre parceiros de negócios definindo qualidade, data de entrega e custos;
- **Plano de coordenação:** é um conjunto de diretivas que descreve um plano para a execução da cadeia. Ou seja, este elemento organiza as interações entre os componentes da cadeia produtiva;
- **Sumários:** são elementos introduzidos para fazer a rastreabilidade, tendo sido objeto do trabalho de Kondo [35], apresentada na seção 2.3.

A ênfase da dissertação é neste último aspecto. A figura 2.1, retirada de [2], ilustra uma simplificação da cadeia produtiva do leite, cujo objetivo é processar leite, produzindo e comercializando produtos relacionados. O monitoramento da qualidade dos produtos é realizado a cada etapa desta cadeia (elementos "*Regulation*" da figura, correspondente às regras).

A figura 2.1, retirada de [2], mostra que o produtor de leite disponibiliza o leite para o laticínio ("*Dairy*"), onde é processado de acordo com as regras ("*Regulation*") e transformado em derivados. A seguir, o laticínio fornece os produtos para o atacadista ("*Wholesale*") que distribui ("*Transport 3*") para o varejista ("*Retail*"), chegando ao consumidor final. Pode-se notar que entre os elementos de produção e armazenamento são necessários transportes.

A Figura 2.2, também retirada de [2], mostra que cada componente da cadeia pode encapsular outras cadeias. Por exemplo, o componente "*Transport 2*" mostrado na Figura 2.1 pode ser encapsulado, conforme ilustra a Figura 2.2, por um "*T2*" que faz o transporte dos produtos criados em "*Dairy*" para um armazém "*W1*", o qual armazena o produto até que "*T3*" faça o transporte do produto para o atacadista ("*Wholesale*"). O monitoramento também é realizado nas cadeias encapsuladas devido à possibilidade de ocorrência de alguma falha a cada estágio. O transporte, encapsulado em "*T2*", também pode ser orientado a regras - por exemplo, temperatura no interior do veículo. Este exemplo simples ilustra alguns dos problemas a serem atacados, como o monitoramento de regras em diferentes níveis de execução e o gerenciamento das mesmas.

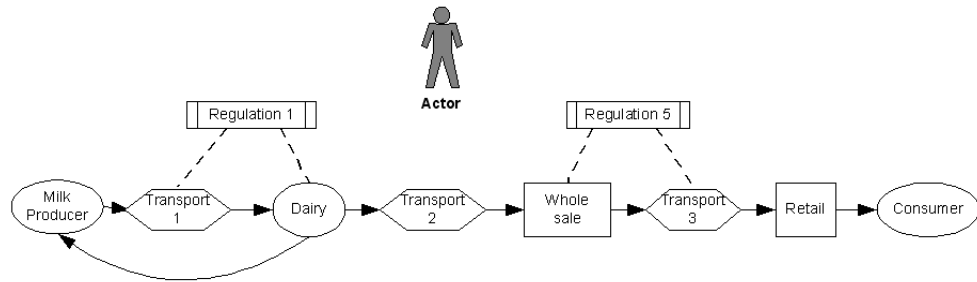


Figura 2.1: Cadeia Produtiva do Leite [2]

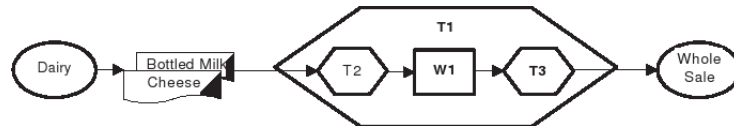


Figura 2.2: Transporte T2 - Cadeia Encapsulada [35]

Bacarin [2] também aponta a necessidade de um repositório específico para regras e um serviço Web voltado à sua gestão (RM - Regulation Manager), sem precisá-los. A dissertação se ocupa igualmente de tais aspectos.

2.2 Rastreabilidade

O conceito de rastreabilidade é fundamental em sistemas de controle qualidade em cadeias produtivas, uma vez que abrange todas as atividades de armazenamento e processamento de informações originadas pela cadeia em cada uma de suas etapas. Mais especificamente, rastreabilidade em cadeias produtivas alimentícias é definido por Moe [40] como a capacidade de localizar um produto e seu histórico através da cadeia produtiva, passando por etapas de colheita, transporte, armazenamento, processamento, distribuição e vendas. Pode ainda ser realizada internamente aos passos da cadeia produtiva, em nível de composição dos produtos e matérias-primas constituintes.

As principais vantagens oferecidas pela rastreabilidade são [25]:

- garantir a utilização de componentes de qualidade no produto final;
- identificar produtos univocamente;
- permitir a identificação e localização de produtos defeituosos;
- localizar fontes do desvio da qualidade;
- disponibilizar o histórico completo de cada produto e seus componentes.

Diversos fatores devem ser levados em consideração ao tratar a rastreabilidade, como os produtos envolvidos, os processos aos quais estes produtos são submetidos, matérias-primas utilizadas, seus tipos e origens, requisitos de qualidade e composição. O monitoramento constante de tais fatores de maneira integrada e automatizada é a chave para a construção de um modelo eficaz de rastreabilidade.

O trabalho Jansen-Vullers et. al. [33] descreve a possibilidade de se realizar a rastreabilidade em dois sentidos diferentes em relação ao processo da cadeia produtiva:

- Para frente (*forward*): indica quais os caminhos percorridos por um determinado produto ou componente a partir de um ponto da cadeia produtiva, como qual o distribuidor final que o comercializou;
- Para trás (*backward*): descreve os acontecimentos passados na vida de um produto, como os locais onde foi armazenado e seus elementos constituintes.

O foco principal em cadeias produtivas alimentares é garantir a qualidade do produto que será entregue ao consumidor final. Nesta linha, podemos citar trabalhos como os de Hobbs [30], Wilson e Clarke [54], Stock [50] e Olsson e Skjöldebrand [43]

Hobbs [30] realiza um estudo sobre os padrões aquisição de carne bovina nos supermercados da Inglaterra, avaliando a influência de fatores como qualidade do produto, rastreabilidade e técnicas de manejo dos animais sobre o modo como os varejistas direcionam suas compras, e evidenciando o aumento de interesse por parte dos consumidores sobre as práticas de produção e processamento dos produtos que são adquiridos.

Wilson e Clarke [54] analisam a qualidade, segurança e rastreabilidade de produtos em cadeias produtivas agrícolas em escala global. O Sistema Food Trak, proposto no artigo, tem como objetivo descrever um mecanismo padrão de concepção e desenvolvimento de software, capaz de prover coesão, localização e disseminação de dados de rastreabilidade de forma homogênea e segura entre os participantes da cadeia produtiva, baseando-se no uso de um meio de comunicação comum (Internet).

Já Stock [50] trata a rastreabilidade como uma forma de preservação da identidade dos produtos, enfatizando sua importância com o crescimento de alimentos produzidos a partir de plantas e animais geneticamente modificados. Ele utiliza a cadeia produtiva alimentar dos Estados Unidos como base para sua análise, concentrando-se nos principais fatores que influenciaram as mudanças da indústria alimentar nos últimos anos.

Olsson e Skjöldebrand [43] analisam o estágio atual do gerenciamento de riscos e garantia de qualidade em cadeias produtivas alimentícias, baseando-se em estudos de caso da indústria de alimentos da Suécia, com foco nos consumidores finais. São destacados os parâmetros críticos envolvidos no intercâmbio de produtos e informações entre os diferentes participantes da cadeia, as interfaces de comunicação entre estes participantes, e

o modo como as soluções têm focado apenas os sistemas e a tecnologia a serem utilizados, e não a segurança do consumidor final ou o relacionamento entre os diferentes atores envolvidos no processo produtivo.

A importância desta área de pesquisa no Brasil é evidenciada pelo surgimento de diversas frentes de trabalho relacionadas à rastreabilidade de alimentos. Exemplos são os trabalhos de Machado [36], Santo e Medeiros [48], Silveira, Resende e Pilatti [18] e Kondo [35]. Este último foi a base principal para o desenvolvimento desta dissertação e, portanto, será descrito mais detalhadamente na seção 2.3.

O trabalho de Machado [36] apresenta dados referentes a custo, importância e dificuldade de implantação de sistemas de rastreabilidade em sistemas agroindustriais, de forma a prover controle de qualidade em cadeias de produção de alimentos, tomando como exemplos os produtos soja e carne bovina. É destacada a necessidade de adaptações da cadeia face às necessidades de rastreabilidade, o que resulta no aumento de custo dos processos envolvidos.

Focados na coordenação e qualidade na rastreabilidade da cadeia produtiva de carne bovina, Santo e Medeiros [48] apresentam uma análise comparativa entre os sistemas francês e brasileiro. O sistema francês de rastreabilidade bovina identifica os animais com dois brincos no momento do nascimento, que contêm um número de identificação e são aprovados pelo Ministério de Agricultura e da Pesca. Os criadores informam às autoridades regionais de agropecuária (*EDE - Etablissement Departamental d'Élevage*) o nascimento do bezerro e suas características. Com essas informações, a EDE provê um passaporte ao animal, que o acompanha por toda a vida. Já o sistema brasileiro permite apenas a identificação do lote de bovinos de onde se originou o produto final, e foi desenvolvido para cumprir a exigência de rotulagem com garantia de qualidade imposta pela União Européia devido à crise provocada pela doença da "vaca louca".

Silveira, Resende e Pilatti [18] destacam em seu trabalho a importância do uso da rastreabilidade como forma de garantir a conformidade dos produtos, relacionada principalmente aos padrões estabelecidos pelos órgãos reguladores de países que importam os produtos agrícolas brasileiros. Nesta linha, apontam a existência da figura da cooperativa como elo produtivo central, atuando como integrador entre os participantes da cadeia produtiva e realizando a rastreabilidade dos produtos comercializados.

Novas exigências dos mercados importadores levaram o Ministério da Agricultura, Pecuária e Abastecimento a criar, em 2002, o Sistema Brasileiro de Identificação e Certificação de Origem Bovina e Bubalina (SISBOV), com o objetivo de identificar, registrar e monitorar, individualmente, todos os bovinos e bubalinos nascidos no Brasil ou importados. Esta iniciativa governamental foi a base para que o Instituto Gênese [21] criasse o Sistema Gênese de Certificação SISBOV (SGCS), com o intuito de proporcionar maior transparência à metodologia de certificação e maior credibilidade ao sistema SISBOV.

Isto possibilitou a transferência do serviço de identificação e rastreabilidade às empresas interessadas em se tornar credenciadoras.

Outros trabalhos relacionados a sistemas de rastreabilidade em cadeias produtivas incluem [12], [45] e [52]. Nossa proposta procura reunir as melhores características de tais trabalhos, como arquitetura distribuída e encapsulamento via Serviços Web, além de se diferenciar pela existência de um mecanismo ativo de controle de qualidade e um modelo de comunicação entre os participantes baseado em disseminação de eventos.

A tabela 2.1 faz uma análise comparativa entre as diferentes propostas. O símbolo “+” indica que o trabalho contempla uma característica, “-” que ele não contempla e “0” que não existe referência à característica no texto analisado.

	Artigo	Bello et al. [3]	Cimino et al. [12]	Pinto et al. [45]	Taniguchi e Sagawa [52]	Wakayama [49]	Kondo [35]	Nossa Proposta
Característica								
Armazenamento de Dados	Local	+	+	0	+			
	Distribuído					+	+	+
Execução de Tarefas	Local		+	+				
	Distribuído	+	+		+	+	+	+
Multiplataforma	+	+	-	0	+	+	+	+
Uso de Padrões Web	+	+	-	-	0	+	+	+
Controle Ativo de Qualidade	-	-	-	-	-	-	-	+
Baseado em Eventos	-	-	-	-	-	-	-	+

Tabela 2.1: Tabela comparativa de trabalhos em rastreabilidade.

Várias outras linhas de pesquisa têm intersecção com nosso trabalho, como por exemplo aquelas que tratam de gerenciamento de eventos em processos de negócio (*Business Processes*). Um exemplo é o trabalho de [22] para tratamento de processos no contexto de *data warehouses*.

2.3 O Modelo de Kondo

Diversos eventos relacionados a processos e serviços executados em uma cadeia produtiva têm regras associadas à manutenção padrões de qualidade. Tais padrões são estabelecidos previamente tanto por órgãos regulamentadores de qualidade, como a ANVISA (Agência Nacional de Vigilância Sanitária), o INMETRO (Instituto Nacional de Metrologia, Normalização e Qualidade Industrial) e a ABNT (Associação Brasileira de Normas Técnicas), quanto por acordos de fornecimento estabelecidos entre as entidades produtoras e consumidoras, como é o caso dos normas de fabricação de produtos direcionados à exportação.

A garantia de qualidade ao consumidor exige definir mecanismos que verifiquem a correta aplicação de tais regras em uma cadeia produtiva.

Este é o foco da dissertação, que parte dos mecanismos de rastreabilidade em agricultura desenvolvidos por Kondo [35]. Kondo especifica um modelo de armazenamento de eventos em uma cadeia agropecuária que permite registrar tais ocorrências em vários níveis de granularidade. Seu modelo interliga eventos ocorridos na fabricação de produtos, processos e serviços envolvidos, de forma integrada.

A arquitetura do modelo está dividida em três camadas: camada de repositórios, camada de sumários e camada do gerenciador de sumários, conforme mostra a figura 2.3, extraído de [35]. Cada uma das camadas contém um conjunto de serviços Web (WS) que encapsula os sumários e repositórios correspondentes. A seguir detalhamos os elementos principais desta arquitetura.

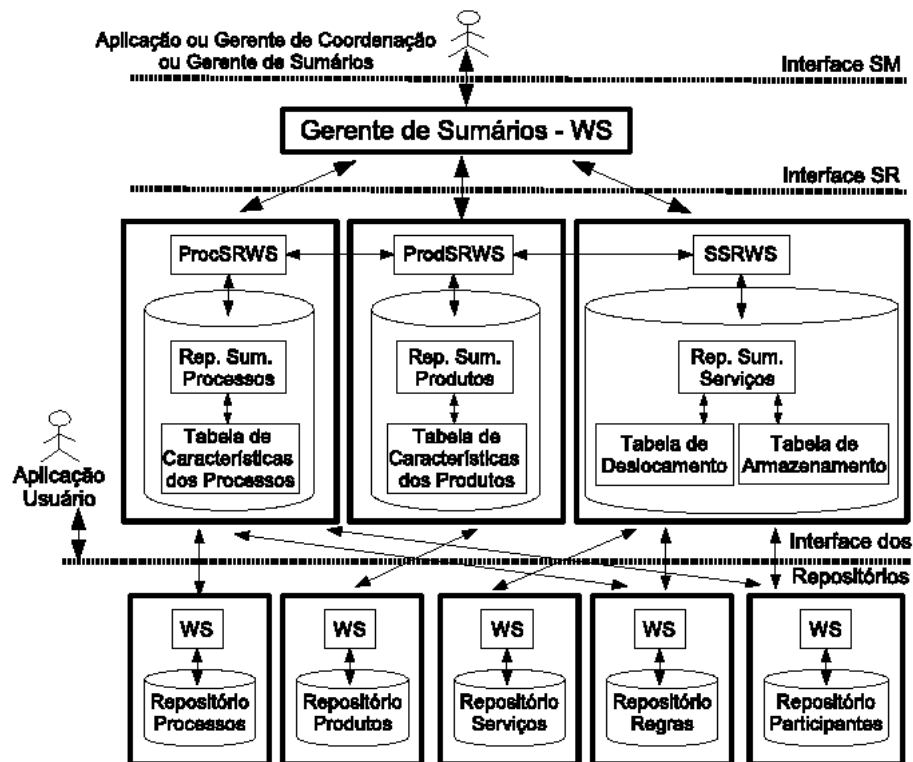


Figura 2.3: Arquitetura de Kondo [35].

Repositórios

O Repositórios armazenam informações sobre a execução e os elementos da cadeia produtiva. Há sete repositórios (Participantes, Produtos, Regras, Contratos, Sumários, Pro-

cessos e Serviços). Aqueles cujo conteúdo é detalhado por [35] são:

- **Participantes:** armazena dados cadastrais dos participantes da cadeia produtiva, como nome, URL e coordenadas geográficas.
- **Produtos:** contém a descrição genérica dos produtos utilizados ou gerados ao longo de uma cadeia produtiva. Possibilita ainda obter informações sobre um produto que pode participar de várias cadeias produtivas distintas.
- **Regras:** armazena as instâncias de regras a serem obedecidas em uma determinada etapa da cadeia produtiva. Este repositório foi detalhado nesta dissertação.
- **Processos:** armazena registros que descrevem os processos pelos quais passam os produtos gerados ou transformados. Estes processos são armazenados como work-flows, indicando a sequência de atividades necessárias para a fabricação de um determinado produto.
- **Serviços:** descreve os tipos genéricos de serviços realizados, como transporte e armazenamento.

O repositório de sumários foi subdividido em vários repositórios. Este repositório, como se verá, é um dos alvos desta dissertação.

Sumários

Segundo [35], sumários podem ser definidos como uma seqüência de registros que, semelhante ao log de um banco de dados, descreve eventos em uma cadeia produtiva. Eles podem relacionar-se a produtos, processos, ou serviços de transporte e armazenamento, onde cada evento gerado na cadeia ocasiona a criação de um ou mais registros no sumário.

Há três tipos de sumários utilizados na arquitetura:

- **Sumário de Produtos:** acompanha as etapas de transformação, armazenamento, e transporte de um produto, desde a sua criação a partir das matérias-primas até o seu consumo final.
- **Sumário de Processos:** armazena informações sobre o processo de fabricação de um determinado produto. Um processo consiste em uma atividade que transforma um conjunto de insumos/produtos (entradas) em um ou mais produtos distintos (saídas).
- **Sumário de Serviços:** armazena informações sobre um determinado serviço. Um serviço consiste de alguma atividade que influencia um produto, sem transformá-lo, estando atualmente restritos a transporte ou armazenamento de um produto.

Kondo [35] descreve em detalhes os campos que compõem os registros de sumários e seus relacionamentos com os demais sumários e repositórios.

O processamento de questões de rastreabilidade envolve consultas e atualizações nos repositórios de sumários, realizadas através de requisições externas ao Gerenciador de Sumários, responsável pela interface entre as aplicações clientes e os sumários. A camada inferior encapsula o acesso aos cinco repositórios por meio de cinco serviços Web distintos, que recebem requisições para os repositórios de Processos, Produtos, Serviços, Regras e Participantes. Já a camada de Sumário, que será o foco principal deste trabalho, é composta por três serviços Web: Produtos, Processos e Serviços. O serviço de Produtos é o central e o único a possuir acesso aos demais serviços de sumário.

2.4 Serviços Web

Segundo Bacarin [2], a especificação de execução de cadeias e sub-cadeias é feita usando workflows. Workflows são uma forma amplamente adotada na literatura para coordenar a execução distribuída de processos. Quando a modelagem de cadeias produtivas em um ambiente distribuído utiliza workflows, sua execução é freqüentemente considerada sob a perspectiva de serviços Web. Neste contexto, cada atividade do workflow invoca um serviço.

Serviços Web ou Web Services são aplicações auto-descritas e modulares para negócios, que apresentam a lógica do negócio como serviços na Internet por meio de interfaces programáveis e de protocolos da Internet. A sua finalidade é proporcionar formas de encontrar, assinar e invocar tais serviços [42]. A implementação de Serviços Web é baseada em padrões e tecnologias específicos, tais como: XML (Extensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) e UDDI (Universal Description, Discovery and Integration) [7].

Em mais detalhes, XML [17] é uma linguagem designada a descrever e estruturar dados. Possui a flexibilidade de definir estruturas de dados complexas [42]. É baseado em padrões de tecnologias para a Web. Esses padrões são definidos pela W3C (World Wide Web Consortium).

SOAP [5] é um protocolo de computação distribuído, que permite a troca de mensagens em ambientes descentralizados e distribuídos. Trata-se de uma das soluções para a invocação e comunicação entre serviços em que SOAP é usado, realizando o transporte de documentos XML por meio de protocolos de Internet como HTTP (Hypertext Transfer Protocol). Neste ambiente, as mensagens encapsulam informações que são transmitidas entre serviços Web e essas mensagens especificam a operação que será invocada. Uma mensagem é estruturada em XML com formato padronizado. A troca de mensagens SOAP baseia-se em uma estrutura para fazer chamada remota de procedimentos (RPC

ou Remote Procedure Calls). Estas mensagens permitem que clientes invoquem procedimentos, funções e métodos de um objeto remoto utilizando um protocolo baseado em XML [7].

WSDL [9] é uma gramática XML para descrever um serviço Web [7]. Muitos dos serviços Web publicados na Internet possuem um documento WSDL, que especifica as funcionalidades do serviço, sua localização na Web e as instruções de como acessá-lo. Esse documento também define a estrutura das mensagens que um serviço envia e recebe [17].

UDDI [13] tem como objetivo possibilitar que desenvolvedores e negócios publiquem e localizem serviços Web na Internet. UDDI define um formato de descrição de serviço baseado em XML, onde provedores podem armazenar suas informações. Este armazenamento ocorre quer em registros UDDI privados, que são acessíveis somente a participantes com permissão, quer em registro UDDI público, que qualquer participante pode acessar [28].

Como serviços Web empregam padrões conhecidos e difundidos, facilitam a comunicação entre diferentes aplicações (em linguagens distintas) e plataformas. Ou seja, interoperabilidade e padronização são características associadas. Além disso, promovem uma abordagem modular para a programação, de modo que várias organizações possam se comunicar com o mesmo serviço Web e várias aplicações possam ter seu processamento simplificado.

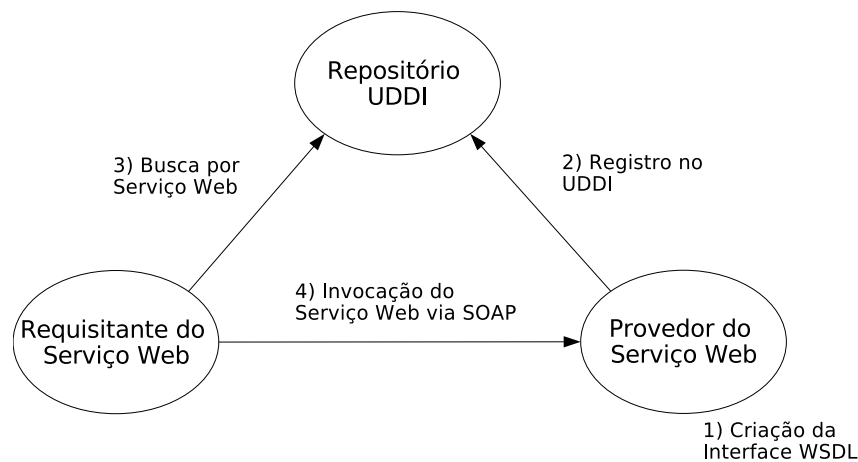


Figura 2.4: Sequência de atividades em uma arquitetura de serviços Web. Modificado de [28].

A figura 2.4 apresenta a sequência de atividades realizadas em uma arquitetura SOA (Service Oriented Architecture) onde um serviço Web opera. Nesta arquitetura, o provedor de serviços oferece diferentes funcionalidades, algumas das quais podem ser serviços

Web. O provedor publica a descrição de seus serviços em um registro de serviços por meio de uma especificação em WSDL. O registro de serviços é um repositório que contém informações sobre serviços Web. O UDDI é um exemplo de tal repositório. Além de armazenar informações técnicas sobre como ter acesso ao serviço (URL - Uniform Resource Locator, dentre outras), o registro de serviços também armazena informações para a categorização de tais serviços (auxiliando a procura) e informações sobre a própria organização que oferece o serviço (por exemplo, endereço e telefone). O cliente de serviços consiste em uma aplicação que procura por um serviço Web no registro de serviços e depois de encontrá-lo invoca tais serviços utilizando o SOAP.

2.5 Bancos de Dados Ativos

A dissertação usa conceitos de bancos de dados ativos para especificação e gerenciamento de regras de qualidade. Ao contrário de bancos de dados passivos, sistemas de bancos de dados ativos (SGBDA) oferecem suporte ao gerenciamento automático de condições definidas sobre o estado de um banco de dados em resposta a estímulos externos.

A aplicação mais comum para bancos de dados ativos é a manutenção de integridade. Uma restrição de integridade em bancos de dados é uma declaração de uma condição que precisa ser alcançada de forma a manter a consistência dos dados [38].

Eventos gerados interna ou externamente provocam uma resposta do próprio banco de dados, independentemente de solicitações de usuários. Neste caso, alguma ação é tomada automaticamente em virtude das condições que foram especificadas para aquele determinado estado do banco de dados [11].

A incorporação de regras a um banco de dados, ao invés de gerenciá-las em aplicações, apresenta as seguintes vantagens:

- não há necessidade de manter as regras dentro do programa de aplicação;
- existe independência de conhecimento em relação aos programas;
- o banco de dados executa automaticamente as regras quando necessário;
- várias aplicações, com a mesma visão do mundo, podem compartilhar um conjunto de regras pré-definido.

Sistemas ativos podem ser usados para vários tipos de aplicações: financeiras, multi-mídia, controle da produção industrial (CIM, controle de inventário, etc.), monitoramento (controle de tráfego aéreo), entre outros. Também são usados para funções do próprio núcleo do banco de dados, como por exemplo, manutenção de consistência, manutenção de visões, controle de acesso, gerenciamento de versões.

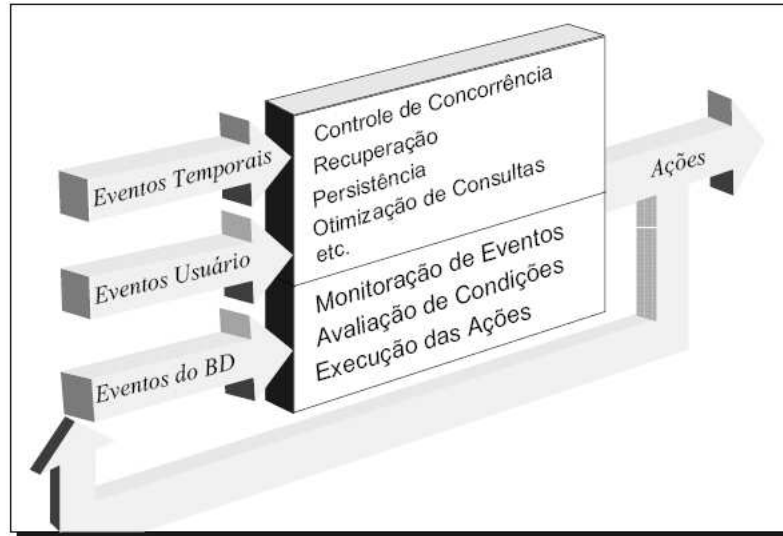


Figura 2.5: Componentes de um Banco de Dados Ativo [11].

A figura 2.5, retirada de [11], ilustra os componentes de um SGBDA, que adiciona características ativas a um banco de dados convencional com três componentes básicos:

- **Monitoramento de Eventos:** detecta a ocorrência de eventos e ativa as regras relacionadas a eles. Este passo requer uma consulta sobre as regras armazenadas para selecionar aquelas apropriadas ao evento ocorrido;
- **Avaliação de Condições:** Verifica se as condições estabelecidas nas regras são efetivamente cumpridas pelo evento;
- **Execução de Ações:** coordena a forma de execução de ações em função do tempo e modo de execução.

Existem ainda linguagens para especificação de regras e condições, além de álgebras que permitem a composição de eventos independentes, bem como a avaliação dos mesmos em função de sua seqüência, conjunção, disjunção e negação. Já a especificação de ações baseia-se no uso de linguagens de consulta simples, estendidas, e no acesso direto ao banco de dados[11].

Um banco de dados ativo é implementado por meio de mecanismos de regras de produção, que descrevem o comportamento a ser adotado pelo sistema. Regras são baseadas em três componentes: evento, condição e ação (E-C-A) [47].

Um evento é um indicador da ocorrência de uma determinada situação em um certo instante. Os eventos primitivos se dividem em três tipos básicos: temporais (agendadores),

operações do próprio BD ("*insert*", "*update*", "*delete*", "*select*") e explícitos ("*user-login*", "*time-out*") [38].

Uma condição é um predicado sobre o estado do banco de dados, relacionando-se a quais fatores devem ser avaliados. Condições são implementadas através de procedimentos da aplicação e, mais freqüentemente, a partir de consultas ao próprio BD.

Já ações são conjuntos de operações a serem executadas quando um determinado evento ocorre e a condição associada é avaliada como verdadeira. Elas representam a forma como o sistema deve responder a um evento, são expressadas na linguagem de programação do banco de dados e podem variar desde uma ação atômica até um programa completo [38]. Um evento pode disparar uma ou mais regras de produção.

Gatilhos são associações de condições e ações. A execução da ação ocorre quando o banco de dados evolui para um estado que leva o gatilho à condição verdadeira. No trabalho, um exemplo de evento é a variação de algum valor associado a um produto, detectado por um sensor que armazena este fato no banco de dados.

2.6 Sistemas Baseados em Eventos

Sistemas Baseados em Eventos ("*Event-Based Systems*") são usados para prover integração entre componentes autônomos ou aplicações em sistemas complexos baseados na troca de eventos. Os eventos trocados entre os participantes encapsulam dados referentes a um determinado acontecimento de interesse. Tais sistemas fazem uso de mecanismos de disseminação de eventos (ou serviço de notificação) para distribuir eventos relevantes a consumidores interessados [10].

O paradigma de interação produtor/consumidor é a base desta arquitetura, consistindo basicamente de um conjunto de clientes que trocam eventos de maneira assíncrona, por meio de um serviço de notificação interposto entre eles. Clientes podem ser caracterizados como produtores ou consumidores. Produtores publicam notificações e consumidores candidatam-se a receber tais notificações através de assinaturas ("*subscriptions*"), que são essencialmente filtros de mensagens. Consumidores podem subscrever várias assinaturas simultâneas para diferentes tópicos de interesse, e estas serão válidas até que o próprio consumidor as cancele junto ao serviço de notificação [10].

A figura 2.6 sintetiza os principais conceitos envolvidos em um arquitetura baseada em eventos. O produtor ("*Publisher 1*") publica ("*publish()*") um determinado evento cujo conteúdo pode ser descrito através de uma hierarquia de conceitos ou domínios ("*dom.msg*"). O consumidor ("*Subscriber 1*") declara-se interessado, junto ao serviço de notificação, em mensagens pertencentes a um determinado domínio ("*subscribe(dom)*"). Ao receber uma notificação do produtor, o serviço de disseminação a envia a todos os consumidores interessados ("*notify(dom.msg)*"). O consumidores que não tenham mais

interesse em receber eventos relacionados a determinado domínio devem cancelar suas assinaturas (`"unsubscribe(dom)"`).

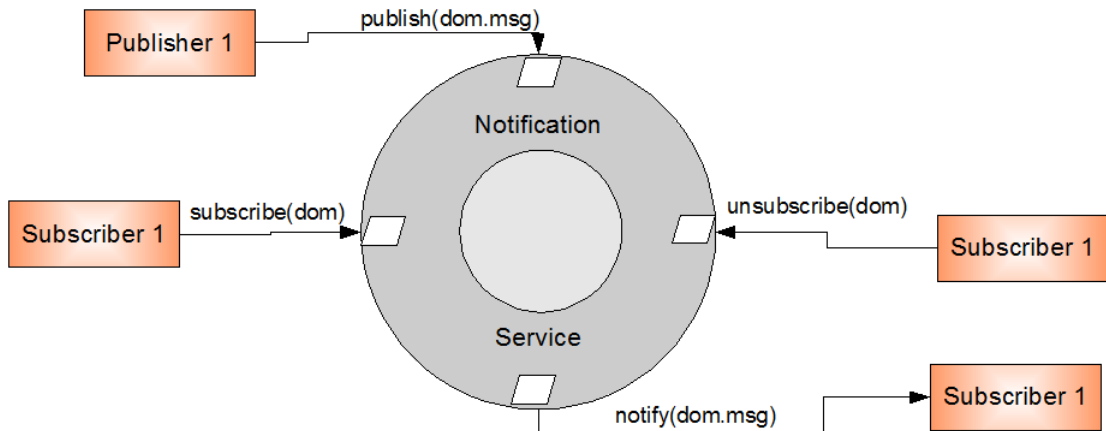


Figura 2.6: Modelo de interação de um Sistema Baseado em Eventos

Aplicações simples são aquelas em que o usuário está no controle, ou onde a velocidade de resposta do sistema não é um fator crítico. Tais aplicações são suficientemente atendidas por interações do tipo *"request-reply"* iniciadas por usuários. Já processos automatizados em ambientes distribuídos precisam reagir a uma grande quantidade de eventos e não podem ser dependentes de processos inicializados por *consumidores*.

Em sistemas baseados em eventos, os *produtores* dos eventos podem não ter conhecimento da existência dos *consumidores*, o que permite maior independência entre as partes envolvidas. Isto facilita a evolução do sistema, uma vez que novas aplicações que reagem a eventos podem ser adicionadas sem afetar a infra-estrutura existente. A reconfigurabilidade inerente a sistemas baseados em eventos é adequada a situações que requerem aplicações e infra-estruturas ágeis, como ocorre, por exemplo, em empresas que desejam acelerar seus processos de negócios cruciais[6].

Há vários exemplos de serviços de disseminação de eventos. O CORBA, por exemplo, foi estendido por um serviço de notificação para prover um mecanismo de interação assíncrona entre seus objetos, fazendo uso de um canal de eventos que funciona como mediador entre produtores e consumidores, além de implementar mecanismos de filtragem de eventos e qualidade de serviço [23].

O Java Message Service (JMS) provê à plataforma JAVA a capacidade de processar mensagens assíncronas, por meio da especificação de uma interface padrão (API) de troca de mensagens implementada por diversos servidores de aplicação. Dois modelos de troca são disponibilizados: *"point-to-point"* (usando filas) e *"publish/subscribe"* (usando classificação por tópicos)[26].

Esta dissertação aplica este modelo de disseminação de eventos como ferramenta para notificar eventuais violações às regras de qualidade, como será visto no Capítulo 3.

2.7 Redes de Sensores

A proliferação de novos tipos de sensores tem proporcionado o surgimento de diversas aplicações, relacionadas principalmente à compreensão, gerenciamento e monitoramento de ambientes e suas variáveis. Em uma cadeia produtiva, a utilização de sensores como forma de controle e obtenção de dados possibilita uma maior automatização dos processos envolvidos, além de atribuir uma maior credibilidade às informações obtidas durante o monitoramento.

Sensores são dispositivos capazes de realizar medições de fenômenos físicos em um dado ambiente. Tais manifestações físicas, como temperatura, luminosidade, humidade, pressão, som e magnetismo são as entradas para funções de processamento de sinais, as quais criam os dados que serão transmitidos a concentradores ou diretamente a aplicações.

O dispositivos variam desde os embutidos em satélites (poucos, grandes e caros) a etiquetas de radio-freqüência(RFID)(numerosas e baratas) [27], como mostra a figura 2.7, retirada de [27]. Redes de Sensores são conjuntos de sensores capazes de se comunicar com outros sensores e com nós controladores [14, 51]. Normalmente, um conjunto de sensores é coordenado por uma estação base que, por sua vez, é conectada a um servidor principal, o qual costuma ser conectada a uma rede, muitas vezes utilizando a Web para disponibilização dos dados capturados [37].

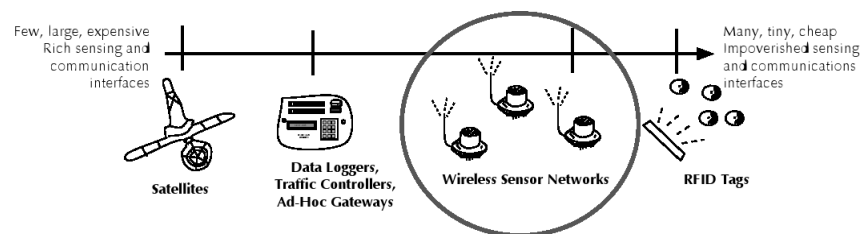


Figura 2.7: Espectro de dispositivos sensores [27]

As redes de sensores sem-fio mais comuns (WSN - Wireless Sensor Networks) são as que utilizam comunicação a rádio. Elas podem ser distribuídas em uma região para prover dados de sensoriamento, mas apresentam problemas de gerenciamento mais complexos, como qualidade na transmissão de sinais, decisão de roteamento e manutenção [44].

O uso de dados de sensores depende da associação às coordenadas geográficas do local onde o dado é coletado (georreferenciamento). As principais aplicações de redes de sensores concentram-se em monitoramento de condições ambientais, climáticas e de

biodiversidade[1]. As aplicações também requerem georreferenciamento dos dados em tempo real, pois muitas vezes os sensores não são fixos (por exemplo, podem estar ligados a objetos móveis).

No entanto, o grande volume de informações gerado e a heterogeneidade de dados observados nos nós das redes de sensores dão origem a novos desafios computacionais, relacionados à integração de fontes heterogêneas, redundância de dados, streams e dados em tempo real.

2.8 Conclusões

Este capítulo apresentou os principais conceitos e trabalhos relacionados à dissertação, a saber: Cadeias produtivas, modelos de rastreabilidade, o modelo de Kondo, serviços Web, bancos de dados ativos, sistemas baseados em eventos e redes de sensores. O próximo capítulo apresenta o modelo proposto para o gerenciamento de qualidade em cadeias produtivas.

Capítulo 3

O Modelo de Gerenciamento de Qualidade

Este capítulo descreve o modelo proposto para o gerenciamento de qualidade em cadeias produtivas. Este modelo foi batizado **SPTracer**, acrônimo para *Supply Chain Traceability* ou Rastreabilidade em Cadeias Produtivas. A seção 3.1 apresenta uma visão geral do problema abordado e a seção 3.2 estabelece as premissas para a realização do controle de qualidade. A seção 3.3 discorre sobre a arquitetura proposta, seguida por seu modelo de gerenciamento global na seção 3.4. As seções 3.5, 3.6 e 3.7 apresentam, respectivamente, os modelos de persistência de dados, de gerenciamento de regras e disseminação de eventos propostos. A seção 3.8 conclui este capítulo.

3.1 Visão Geral

O objetivo desta dissertação é a elaboração de um modelo de controle de qualidade de produtos em cadeias produtivas, baseado na aplicação de regras de qualidade referentes à constituição e comportamento de produtos, processos e serviços. A solução provê mecanismos ativos de detecção de violação de restrições previamente armazenadas, por meio da avaliação dos dados contidos no sistema de rastreabilidade. Para isto, utiliza aspectos de bancos de dados ativos, em que regras armazenadas são verificadas por meio de um de sistema de gatilhos.

Na arquitetura de Kondo [35], os registros que armazenam eventos de produtos, processos e serviços prevêm um campo que deve ser associado às regras aplicáveis. No entanto, o modelo não especifica tais regras, nem informa como devem ser gerenciadas ou como monitorar suas aplicações, deixando isso para trabalhos futuros. Nosso modelo preenche esta lacuna.

Seguindo a terminologia de Kondo [35], um “sumário” denota um arquivo que contém

registros de log sobre eventos da cadeia, enquanto o “repositório” correspondente contém dados cadastrais ou fatos.

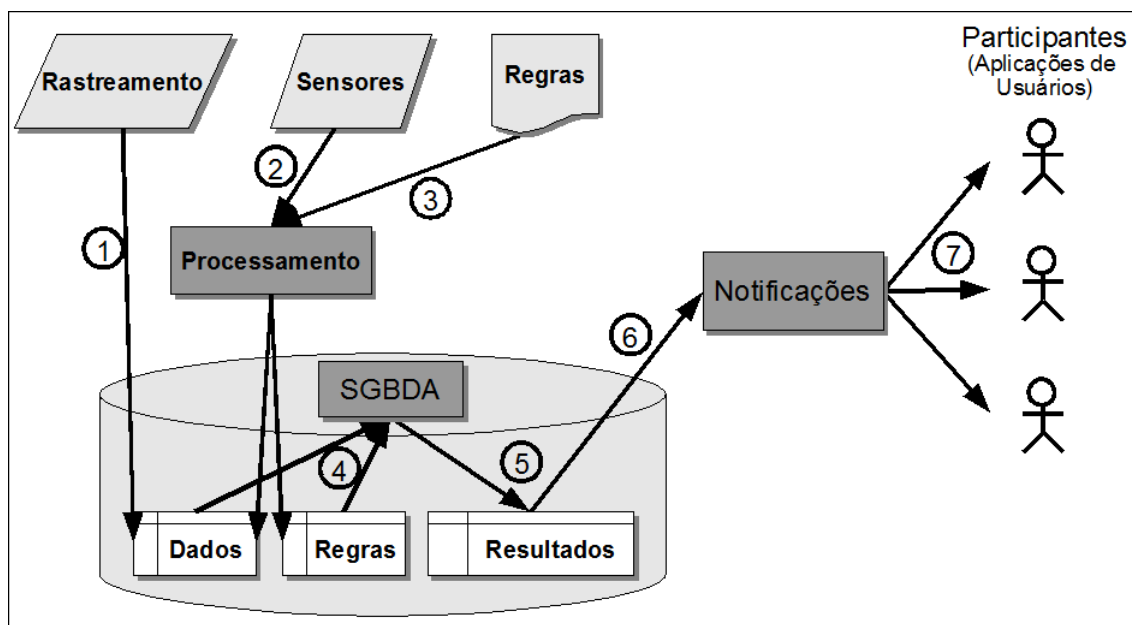


Figura 3.1: Visão geral da proposta.

O exemplo a seguir foi utilizado como um cenário básico para a especificação do modelo, ilustrado pela figura 3.1. Neste trabalho estamos restritos a questões de qualidade associadas a situações que podem ser detectadas por sensores. Consideremos a Portaria CVS 15, de 07/11/1991, expedida pelo Centro de Vigilância Sanitária de São Paulo, normatizando e padronizando o transporte de alimentos para consumo humano [16]. O Artigo 2º, inciso 13 da portaria define que o transporte de alimentos sob a categoria “refrigeração” deve ser realizado em um veículo fechado, sob temperatura ao redor de 4°C, e não superior a 6°C. Apesar da imprecisão de valores estabelecidos pela norma, podemos considerar que um transporte de refrigeração ideal deveria manter a temperatura dos produtos envolvidos sempre entre 2°C e 6°C.

Seguindo o modelo de Kondo, as informações referentes ao transporte destes produtos devem ser armazenadas no Sumário de Serviços do ator que o realiza (figura 3.1-1). O Sumário de Produtos que armazena o registro referente ao produto transportado também receberia informações relacionadas ao serviço, de forma a não perder dados que garantam a rastreabilidade.

Neste estudo de caso, entendemos que o controle de temperatura deve ser feito através de um sensor próprio, cujos dados devem ser processados e relacionados ao serviço de transporte no sistema de rastreabilidade (figura 3.1-2). O evento, neste caso, corresponde

à detecção da mudança de temperatura e armazenamento no banco de dados desta ocorrência. O uso de sensores permite realizar medições precisas relacionadas a fatores externos que possam vir a influenciar a qualidade e eficácia de um determinado produto. Surge assim a necessidade de se especificar meios para o armazenamento dos dados coletados pelos sensores. Tais dados podem ser analisados a posteriori para detectar violações (auditoria). Alternativamente, podem ser monitorados em tempo real. A dissertação se concentra em aspectos de auditoria.

A portaria, por sua vez, deve ser mapeada em uma regra descrita computacionalmente, permitindo aplicar condições impostas por essa restrição de qualidade sobre os dados coletados durante o transporte, e verificar qualquer violação diretamente nas bases de dados (figura 3.1-3). O padrão utilizado nesta especificação da regra deve possuir um alto poder de expressividade, uma vez que as restrições envolvidas podem ser de natureza complexa e, conseqüentemente, de difícil representação em um modelo computacional.

Uma vez confrontados os dados e as restrições (figura 3.1-4), os resultados das análises são armazenados no banco de dados local (figura 3.1-5), e a ocorrência de violações dá origem a notificações (figura 3.1-6). Estas devem ser entregues a todos os participantes interessados naquela determinada restrição, permitindo que as devidas ações corretivas sejam tomadas. Isto implica na existência de um mecanismo capaz tanto de registrar o interesse dos atores da cadeia em uma regra específica, quanto de garantir a entrega segura das notificações originadas por ela (figura 3.1-7).

Este cenário simples nos permite identificar os pontos chave para a definição do modelo de gerenciamento de qualidade, descritos na seção 3.2.

3.2 Premissas

O controle de qualidade efetivo sobre dados de rastreabilidade em cadeias produtivas parte das seguintes premissas:

- **Ferramenta de especificação** : é preciso disponibilizar uma ferramenta para a definição das regras de qualidade;
- **Armazenamento** : deve ser possível armazenar os dados a serem analisados, bem como as regras que definem qualidade;
- **Avaliação** : é preciso criar mecanismos para avaliação dos dados em função das regras.
- **Disseminação** : deve haver um mecanismo que permita utilizar os resultados desta avaliação.

Todos esses fatores devem ser relacionados à natureza do problema que se deseja abordar (na dissertação, cadeias produtivas agrícolas), ao ambiente distribuído e à heterogeneidade dos participantes.

Nosso trabalho parte do modelo de armazenamento de dados especificado por Kondo [35], estendendo-o por meio da especificação do Gerente de Regras mencionado originalmente por Bacarin [2] e de um módulo de disseminação de eventos relacionadas a essas regras. Contemplamos ainda a manipulação de dados de sensores utilizados no monitoramento das etapas da cadeia.

Este capítulo descreve o modelo proposto para o gerenciamento de qualidade em cadeias produtivas, baseado na aplicação de regras de qualidade referentes à constituição e comportamento de produtos, processos e serviços. O modelo é composto por quatro elementos principais:

- **Modelo de Gerenciamento Global** (seção 3.4), responsável pela lógica de funcionamento do sistema de gerenciamento da cadeia produtiva, e ligado à premissa de especificação.
- **Modelo de Persistência de Dados** (seção 3.5), ligado à premissa de armazenamento;
- **Modelo de Gerenciamento Ativo de Regras** (seção 3.6), ligado à premissa de avaliação de regras;
- **Modelo de Disseminação de Eventos** (seção 3.7), ligado à premissa disseminação de eventos relacionados às regras.

O modelo provê mecanismos ativos de detecção de violação das restrições armazenadas.

3.3 Arquitetura

A figura 3.2 ilustra os componentes da arquitetura proposta, composta por 3 camadas encapsuladas por serviços Web. As setas espessas indicam as interações entre camadas e atores externos, realizadas por meio de invocações dos serviços Web, enquanto as setas finas indicam a comunicação direta entre os módulos que compõem uma mesma camada.

A camada inferior engloba os serviços de persistência de dados e de gerenciamento, conversão e aplicação das regras de qualidade. O Módulo de Persistência é o responsável pelo armazenamento e recuperação dos dados de rastreabilidade gerados durante a permanência de um produto em uma cadeia produtiva. Ele é composto por todos os sumários e repositórios especificados pelo modelo de Kondo [35], os quais foram modificados e estendidos para suportar o armazenamento de dados de sensores e o armazenamento

e aplicação de regras de qualidade sobre estas informações, conforme detalhado na seção 3.5.

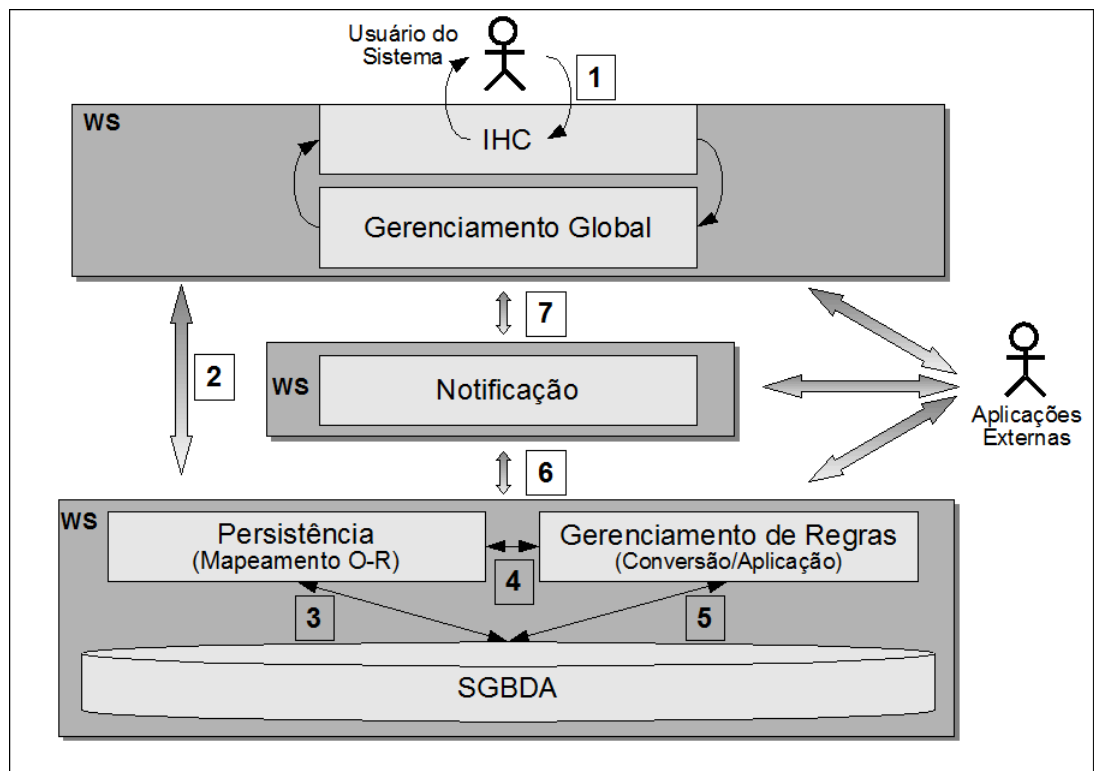


Figura 3.2: Arquitetura proposta.

O módulo de Gerenciamento de Regras é responsável pela conversão das regras armazenadas em gatilhos e procedimentos processáveis e pela aplicação e controle destas regras e das notificações geradas no SGBD local. Ele é parte integrante da camada de persistência pois interage diretamente com o SGBD. Neste modelo, cada participante da cadeia produtiva possui seu próprio gerente. Isso permite uma maior flexibilidade na escolha do SGBD, pois o modelo de dados e as conversões de regras podem ser customizado localmente. Isto também possibilita o controle de desempenho e da aplicação das regras de qualidade, uma vez que a comunicação não é influenciada pela latência natural dos serviços Web. A seção 3.6 apresenta maiores detalhes sobre este módulo.

No nível intermediário encontra-se a camada de disseminação de eventos, composta pelo módulo de notificação. Este módulo pode ser compartilhado entre diversos participantes pois age como canal de comunicação entre os mesmos. Cada participante deve especificar em seu sistema o endereço para o módulo de disseminação que irá utilizar para as notificações de qualidade originadas em seu repositório de dados, de modo que os demais participantes da cadeia interessados possam recebê-las. A especificação deste

serviço de comunicação é descrita na seção 3.7.

A camada superior é formada pelos módulos de interface homem-computador e de gerenciamento global do sistema, responsável pelo controle de acesso e elaboração de consultas complexas envolvendo diferentes repositórios de dados. Diferentemente do Gerente de Sumários de Kondo, este módulo não centraliza todas as interações entre os repositórios de dados e atores externos. Seu papel limita-se a prover a permissão de acesso aos repositórios de dados que a ele estão subordinados, e a interagir com outros gerentes para recuperar informações de outros repositórios quando necessário. Esta camada é detalhada na seção 3.4.

Um produto em uma cadeia produtiva é representado computacionalmente como um registro armazenado em um SGBD. Durante sua permanência na cadeia, estes elementos sofrem diferentes ações externas que podem modificar sua composição ou estado, como processos de transformação ou serviços de armazenamento e transporte. São ainda utilizados, na cadeia, sensores capazes de monitorar as variáveis envolvidas no ambiente em que se aplicam, como temperatura, luminosidade ou localização. Os dados coletados pelos sensores são processados e armazenados no SGBD, de maneira a permitir a auditoria dos eventos registrados.

O processo de controle é iniciado com a especificação de uma regra de qualidade que atue sobre um produto, um processo, um serviço, dados de sensores ou a combinação destes. Um especialista é responsável por especificar as regras e suas cláusulas, que também são armazenadas como registros no SGBD. No entanto, para que as regras passem a atuar sobre os dados armazenados é necessário convertê-las em gatilhos e procedimentos do banco de dados.

Uma vez armazenada, o mecanismo ativo passa a verificar a regra, atuando sobre os dados especificados em suas cláusulas. Encontrada uma violação, o módulo de notificação é acionado para providenciar a disseminação daquela informação aos participantes e atores externos que tenham declarado interesse em recebê-la. “Declarar interesse” significa registrar-se previamente no módulo de notificação para receber um conjunto específico de notificações, filtradas sobre algum critério. Determinadas as formas de armazenamento de dados, especificação e controle de regras e disseminação de violações, é fechado o ciclo de gerenciamento de qualidade.

O exemplo a seguir ilustra a forma como os módulos da arquitetura interagem. Consideremos um usuário que queira controlar a temperatura de armazenamento de um determinado produto cujo rastreamento já está sendo contemplado pelo sistema. O usuário especifica a regra usando a interface Web do sistema (figura 3.2-1), que os repassa ao Módulo de Gerenciamento Global. Este as transmite para o Módulo de Persistência via invocações de serviços Web (figura 3.2-2). O Módulo de Persistência recebe o conjunto de dados e, utilizando um mapeamento objeto-relacional, armazena-os no SGBD (figura

3.2-3).

Já a verificação da qualidade é realizada da seguinte forma. Periodicamente, o Módulo de Gerenciamento de Regras verifica a existência de regras adicionadas ou alteradas (figura 3.2-4), converte-as em scripts processáveis e os insere no SGBDA (figura 3.2-5). Os sensores instalados pelo usuário para a medição de temperatura de seu armazém fazem a coleta de dados, que são enviados para o Módulo de Persistência seguindo os mesmos passos descritos no parágrafo anterior. O script correspondente à regra monitora os dados de sensor inseridos, avalia as condições especificadas e armazena no banco de dados as violações encontradas e seus respectivos registros de notificação.

Finalmente, a notificação ocorre da seguinte maneira. Periodicamente, o Módulo de Gerenciamento de Regras verifica a existência de novas notificações (figura 3.2-4) e as envia ao Módulo de Notificação (figura 3.2-6). Este se encarrega de disseminar os alertas de violações aos Módulos de Gerenciamento Global (figura 3.2-5) dos usuários que se inscreveram no Módulo de Notificação para recebê-las, fechando assim o ciclo de controle.

3.4 Modelo de Gerenciamento Global

Segundo Bacarin [2], o Gerente de Coordenação é responsável pelo funcionamento geral da cadeia, compreendido pela lógica de negócio, a interação entre participantes, os contratos e planos de coordenação. Já o acesso aos repositórios de dados deve ser controlado pelo Gerente de Sumários.

Como nossa proposta concentra-se no gerenciamento de qualidade e não nos aspectos abordados por estes gerentes, introduzimos a figura do Gerente Global como centralizador dos aspectos envolvidos com a lógica do controle de qualidade, restringindo suas funções à concessão de acesso aos repositórios de dados, a prover a interface de especificação de regras aos usuários, ao registro de interesse no recebimento de notificações em módulos de disseminação e ao tratamento de tais notificações quando são recebidas.

O Gerente Global deve fornecer informações para a interface pela qual os usuário interagem com o sistema. Deve ser possível ao usuário especificar as regras de qualidade que serão armazenadas no Repositório de Regras (seção 3.5.1), além de fazer a instanciação das regras no Sumário de Regras (seção 3.5.3).

Outras funcionalidades disponíveis neste módulo são a especificação de consulta aos dados dos sumários e repositórios, o registro de interesse nos módulos de disseminação para o recebimento de notificações de violações e o controle das notificações recebidas.

Nosso modelo de gerenciamento de qualidade possui uma arquitetura de distribuição de violações baseado em mensagens assíncronas, e cujo processamento é de responsabilidade do Modelo de Disseminação de Eventos, descrito na seção 3.7. Para que um determinado

sistema receba as notificações de seu interesse, é necessário que se registre no Módulo de Disseminação correspondente, o que deve ser possível através da interação entre o usuário e o Módulo de Gerenciamento Global. Uma vez registrado, este sistema passará a receber informações sobre violações, que deverão ser processadas para a avaliação dos usuários do sistema.

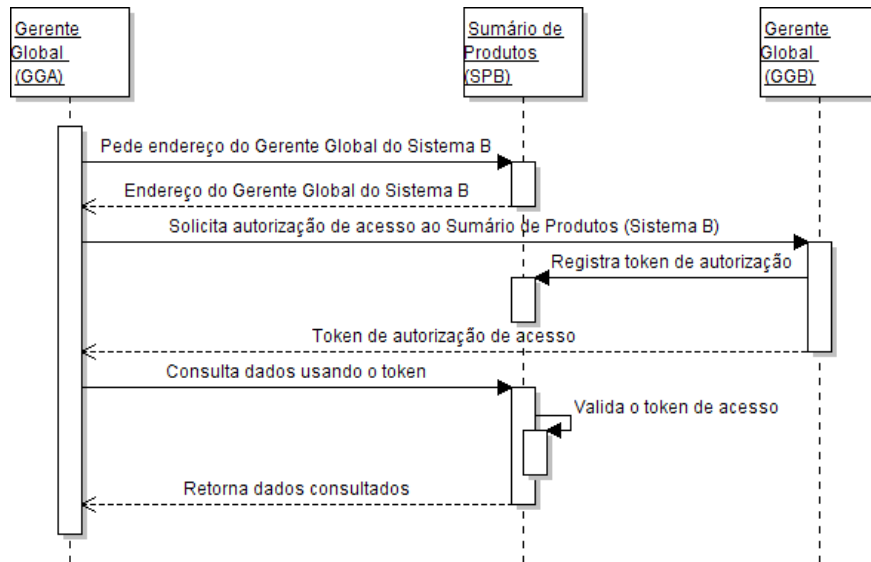


Figura 3.3: Diagrama de Seqüência de Autorização de Acesso a um Repositório.

Diferentemente do Gerente de Sumários de Kondo [35], o Gerente Global não centraliza todas as interações entre os repositórios de dados e atores externos. Seu papel limita-se a prover a permissão de acesso aos repositórios de dados que a ele estão subordinados, aumentando a eficiência do sistema como um todo por eliminar a figura de um “atravesador” de dados entre os participantes da cadeia, um potencial gargalo e ponto crítico de falha. As permissões são atribuídas aos participantes interessados através de tokens e em diferentes níveis de acesso, de forma a restringir as operações em um repositório de acordo com as diretrizes imposta pelo Gerenciador Global ao qual ele está subordinado.

A figura 3.3 ilustra a seqüência de atividades e os módulos envolvidos no processo de autorização de acesso a um determinado repositório de dados, neste caso representado pelo Sumário de Produtos. No exemplo, o Gerente Global do sistema “A” (*GGA*) deseja acessar as informações armazenadas no Sumário de Produtos de um sistema “B” (*SPB*), cujo acesso é controlado pelo Gerente Global do sistema “B” (*GGB*). Primeiramente, *GGA* obtém de *SPB* o endereço do Gerente Global ao qual *SPB* é subordinado (*GGB*). Então, *GGA* solicita a *GGB* o direito de acesso a *SPB*. *GGB* analisa a requisição e, caso autorizada, gera um token de acesso com período de validade determinado, o cadastra em *SPB* e informa o token a *GGA*. *GGA* passa então a utilizar o token em suas operações

aos dados de *SPB*, que permite então o acesso de *GGA* às informações.

3.5 O Modelo de Persistência de Dados

O modelo de Kondo[35] foi estendido com novos sumários e repositórios para permitir o gerenciamento de regras. A figura 3.4 mostra a nova estrutura dos repositórios.

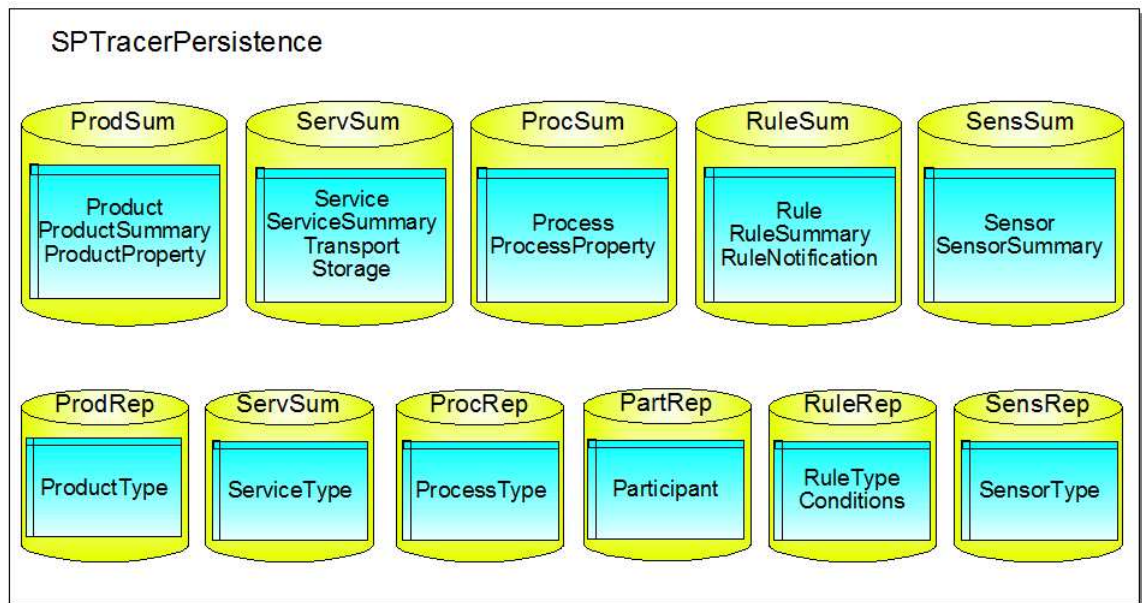


Figura 3.4: Repositórios de dados.

Dois novos sumários - Sumário de Regras (*RuleSum*) e Sumário de Sensores (*SensSum*) - e dois novos repositórios - Repositório de Regras (*RuleRep*) e Repositório de Sensores (*SensRep*) - foram adicionados. Estes novos elementos servem para armazenar os dados relacionados às regras e aos sensores utilizados para monitoramento dos produtos.

A comunicação entre os repositórios e os atores e módulos externos à camada de persistência é totalmente baseada em serviços Web, sendo que cada sumário e repositório possui seu serviço Web correspondente, conforme ilustrado na figura 3.5.

A seção 3.5.5 descreve as modificações realizadas no modelo de Kondo, incluindo reestruturação nos repositórios e a especificação de identificadores de registros em formato de URI. A primeira modificação inseriu dados de sensores e alterações necessárias para a verificação automática de regras. A segunda atribui ao identificador único dos objetos o endereço do serviço Web correspondente ao repositório em que estão armazenados e a entidade à qual correspondem neste repositório (ou seja, introduz a possibilidade de

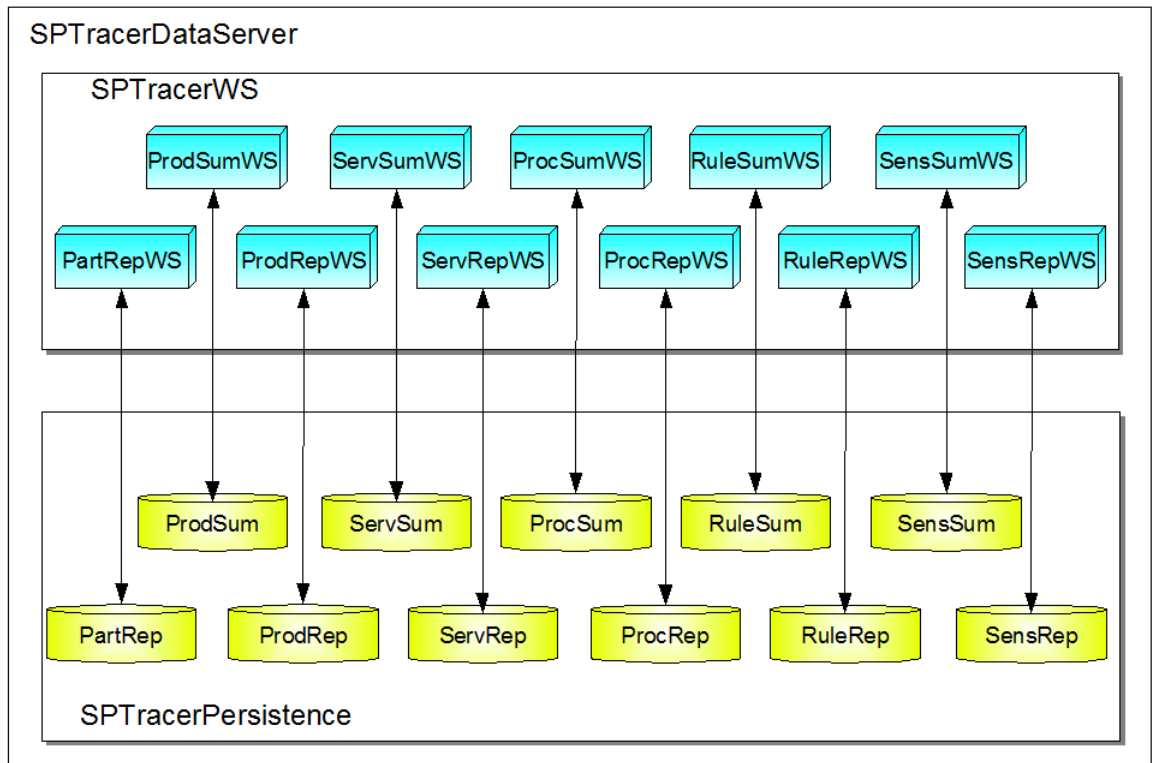


Figura 3.5: Arquitetura do servidor de dados do modelo.

monitoramento e rastreabilidade usando serviços distribuídos - enquanto Kondo pressupõe a existência de um serviço gerenciador central).

3.5.1 Repositório de Regras

O Repositório de Regras contém a especificação das regras de qualidade definidas pelos usuários, a serem aplicadas sobre os dados que registram os eventos da cadeia produtiva. O repositório armazena as informações relacionadas à natureza de cada regra, como sua descrição e a ligação com a portaria ou legislação que a instituiu. Isso permite o compartilhamento de regras pré-definidas entre os participantes da cadeia, favorecendo o reuso destas informações.

Regras são armazenadas em duas tabelas: Classes e Cláusulas. A primeira registra dados genéricos sobre um tipo de regra. A segunda detalha as cláusulas que compõem o predicado a ser verificado. Conforme ilustrado na tabela 3.1(a), uma regra é composta pelos seguintes atributos:

- `id_regratp` : campo identificador da regra;

Repositório de Regras - Classes			
id_regratp	descrição	origem	objeto
tempLeite	Controle de temperatura do leite	Portaria CVS 15, de 07/11/1991	ProdutoLeiteTipo1
transpCarga	Controle de localização da carga	Reg. Interno 12-23	ProdutoLeiteTipo1

(a)

Repositório de Regras - Cláusulas					
id_regra	parametro	operador	valor	unidade	precisao
tempLeite	Temperatura	<	10	Celsius	0.001
tempLeite	Temperatura	>	1,5	Celsius	0.001

(b)

Tabela 3.1: Repositório de Regras.

- descrição : descrição da regra para facilitar a identificação;
- origem : origem da regra (identificação da portaria, legislação ou procedimento interno que a criou.)
- objeto_tipo : define sobre qual tipo de objeto (registro) deve ser aplicada a regra. Corresponde a identificadores de produtos, processos e serviços, cuja descrição é armazenada respectivamente nos repositórios de produtos, processos e serviços.

A tabela 3.1(b) mostra o armazenamento dos componentes da regra, seguindo o modelo E-C-A, com condições a serem verificadas para que seja configurada uma violação a uma regra:

- id_regra : identificador da regra à qual pertencem as condições;
- parametro : atributo (campo) cujo valor será inspecionado. Trata-se de um atributo do objeto monitorado;
- operador : operador usado para a verificação de valor;
- valor : valor do atributo.
- unidade : a unidade em que são mensurados os dados do atributo valor.
- precisão : a precisão dos dados do atributo valor.

Com esse conjunto de informações é possível construir mecanismos de monitoramento ativo que atuarão sobre os dados armazenados nos sumários. Por exemplo, a regra “tempLeite” de que trata a figura deve ser aplicada a produtos do tipo “ProdutoLeiteTipo1”, cuja temperatura deve variar entre 1.5 e 10 graus Celsius medidos por sensores. Caso

a temperatura registrada no sumário de sensores para este produto tenha violado estes limites, haverá problema de qualidade.

A complexidade de uma regra varia conforme o poder de expressividade da linguagem em que ela é descrita. Considerando apenas a linguagem procedural PL/SQL, os eventos e condições avaliadas por determinada regra poderiam ser combinados via operadores lógicos. A título de simplificação, nosso modelo se limita à especificação de regras simples: de múltiplas cláusulas combinadas aditivamente (operador lógico "E"). Assim, a violação de uma das cláusulas é suficiente para configurar a não conformidade dos dados avaliados.

3.5.2 Repositório de Sensores

O Repositório de Sensores foi especificado para armazenar as características principais de cada modelo de sensor utilizado para medição de fatores críticos na cadeia produtiva, como temperatura, umidade e localização. Funciona como uma base de referência sobre qual o modelo de sensor a ser utilizado em uma determinada aplicação, facilitando a padronização e controle de modelos homologados para uso pelos órgãos reguladores.

A tabela 3.5.2 ilustra este repositório, formado pelos seguintes atributos:

- `id_sensortp` : identificador do modelo de sensor;
- `modelo` : modelo do sensor (código especificado pelo fabricante);
- `descricao` : descrição do sensor;
- `unidade` : a unidade em que são mensurados os dados coletados.
- `precisao` : a precisão dos dados coletados pelo sensor.

Repositório de Sensores				
<code>id_sensortp</code>	<code>modelo</code>	<code>descricao</code>	<code>unidade</code>	<code>precisao</code>
<code>sensorType1</code>	M1	Sensor de temperatura	Celsius	0.001
<code>sensorType2</code>	M2	Sensor de pressão atmosférica	ATM	0.1
<code>sensorType3</code>	M3	Sensor de posicionamento global	Graus	0.05

Tabela 3.2: Repositório de Sensores.

Conforme exemplo da tabela, um sensor com identificador “`sensorType3`” tem modelo M3 e realiza leituras sobre localização geográfica, as medidas são feitas em graus e a precisão do aparelho é de 0.05 Grau.

3.5.3 Sumário de Regras

Regras E-C-A têm um componente de Ações, que são medidas a serem tomadas quando ocorre um evento **E** e a condição **C** é satisfeita [11]. Estas ações são registros armazenados em um repositório específico, semelhante ao que ocorre com os *logs* de produtos e serviços no modelo de Kondo [35]. Dessa forma é possível manter um histórico e compartilhar todas as informações relacionadas à violações das regras especificadas.

A este repositório damos o nome de Sumário de Regras. Nele são armazenados três tipos de informações: as relacionadas à instanciação (aplicação) de uma determinada regra, os registros de eventos (*logs*) detalhados e as notificações relacionados à regra instanciada.

A tabela 3.3 ilustra a utilização deste sumário. A tabela 3.3(a) mostra que duas regras do tipo (“tempLeite” e “transpCarga”) foram aplicadas durante um determinado período. A tabela 3.3 (b) registra as violações detectadas para cada instanciação das regras. Ele mostra que a regra “tempLeite1” foi aplicada duas vezes sobre instâncias de leite “ProdLeite1” e “ProdLeite2”, sendo que na primeira vez foi detectada uma violação. A tabela 3.3 (c) mostra um exemplo de notificação a ser enviada aos demais participantes da cadeia, contendo informações sobre a violação detectada pela instância de regra “tempLeite1”.

Sumário de Regras - Instâncias de Regras							
id_regra	tipo	objeto_tipo	objeto_id	início	fim	flag_altera	flag_aplica
tempLeite1	tempLeite	ProdLeiteTipo1		12/01/08 00:00	12/01/09 00:00	verdadeiro	verdadeiro
transpCarga1	transpCarga	ProdLeiteTipo2	ProdLeite5	01/02/08 00:00	12/10/08 00:00	falso	verdadeiro

(a)

Sumário de Regras - Resultados das análises							
id_regra	tipo	data	id_objeto	condição	sensor_id	medida_id	flag_nova
tempLeite1	Violação	12/01/08 00:00	ProdLeite1	Temperatura < 10	sensor1	sensor1_545	verdadeiro
transpCarga1	Conformidade	01/02/08 00:00	ProdLeite2	Temperatura < 10	sensor3	sensor3_623	verdadeiro

(b)

Sumário de Regras - Notificações						
id_regra	data	objeto_tipo	objeto_id	sensor_id	flag_nova	
tempLeite1	12/01/08 00:00	ProdLeiteTipo1	ProdLeite1	sensor1	verdadeiro	

Tabela 3.3: Sumário de Regras.

Os atributos de um registro do sumário de regras são os seguintes:

- `id_regra`: identificador da regra;
- `tipo`: tipo da regra (definido no Repositório de Regras - tabela 3.1);
- `objeto_tipo`: tipo do objeto alvo desta regra (definido no Repositório de Produtos, Processos ou Serviços);
- `objeto_id`: identificador do objeto alvo desta regra;

- início : *timestamp* do momento em que a regra começar a ser aplicada;
- fim : *timestamp* do momento em que a regra deve deixar de ser monitorada.
- flag_altera: identificador booleano que indica se a regra foi instanciada ou modificada, e ainda não processada pelo Gerente de Regras. Campo adicionado para efeito de implementação.
- flag_aplica: identificador booleano indicando se a regra já foi convertida em gatilho armazenado no SGBD. Campo adicionado para efeito de implementação.

Já os registros de eventos relacionados à regra possuem os seguintes atributos:

- id_regra: identificador da regra;
- tipo : tipo da notificação (conformidade ou violação);
- data : *timestamp* do momento em que foi detectado o evento;
- objeto_id : identificador do objeto que deu origem à notificação;
- condição : condição da regra que foi violada;
- sensor_id : identificador da instância do sensor que detectou o evento.
- medida_id : identificador da medição do sensor que originou o evento.
- flag_nova: identificador booleano que indica se o registro foi recentemente inserido ou modificado, e deve ser processado pelo Gerente de Regras. Campo adicionado para efeito de implementação, é usado para gerenciar notificações e a disseminação de eventos.

Os registros de notificações são compostos pelas seguintes informações:

- id_regra: identificador da regra;
- data : *timestamp* do momento em que foi detectado o evento;
- objeto_tipo : tipo do objeto alvo da regra (identificador de um registro no Repositório de Produtos, Processos ou Serviços);
- objeto_id : identificador do objeto alvo da regra;
- sensor_id : identificador da instância do sensor que detectou o evento.
- flag_nova: identificador booleano que indica se a notificação foi recentemente inserida ou modificada, e deve ser processada pelo Gerente de Regras.

3.5.4 Sumário de Sensores

O Sumário de Sensores tem a função de armazenar dois tipos de registro: a instância referente à utilização de um sensor (ou seja, cada vez que um sensor é instanciado); e os dados por ele coletados a cada utilização. Deste modo é possível aplicar o mecanismo ativo de regras sobre os dados coletados da mesma maneira que acontece com as demais informações armazenadas pelo sistema.

A tabela 3.4 exemplifica a utilização deste sumário, e sua estrutura é definida no seguinte formato:

Sumário de Sensores - Instâncias de Sensores					
id_sensor	tipo	objeto_tipo	objeto_id	data_inicio	data_fim
sensor1	sensorType1	ProdLeiteTipo1		12/01/08 00:00	12/01/09 00:00
sensor2	sensorType3	ProdLeiteTipo2	ProdLeite5	01/02/08 00:00	12/10/08 00:00

(a)

Sumário de Sensores - Medições			
id_sensor	data	atributo	valor
sensor1	04/03/2008 12:34.12	Temperatura	9
sensor1	04/03/2008 12:35.12	Temperatura	15
sensor1	04/03/2008 12:36.12	Temperatura	14

(b)

Tabela 3.4: Sumário de Sensores.

- id_sensor: identificador do registro de utilização do sensor;
- tipo : tipo do sensor utilizado (definido no Repositório de Sensores);
- objeto_tipo : tipo do objeto alvo do sensor (identificador de um registro no Repositório de Produtos, Processos ou Serviços);
- objeto_id : identificador do objeto alvo do sensor;
- data_inicio : *timestamp* do momento em que foi iniciado o uso do sensor;
- data_fim : *timestamp* do momento em que foi finalizado o uso do sensor;

Já os dados coletados são armazenados no seguinte padrão:

- id_sensor: identificador do registro de utilização do sensor;
- data: *timestamp* do momento em que foi realizada a medição;
- atributo : propriedade analisada;
- valor : valor mensurado.

Diversos fatores podem influenciar a acurácia dos dados e a sua análise, como precisão, sincronização temporal, falhas de leitura e perda de informações. Considerar tais fatores foge ao escopo deste trabalho, para o qual é suficiente supor que as informações armazenadas são integralmente precisas e há compatibilidade entre unidades de grandezas especificadas e medidas.

3.5.5 Adaptações ao Modelo de Kondo

Foram realizadas adaptações ao modelo de repositórios e sumários especificado originalmente por Kondo [35], para possibilitar a aplicação de regras sobre os dados armazenados.

Reestruturação dos repositórios de dados

O modelo de Kondo [35] define um atributo nos registros de produtos, processos e serviços armazenados que apontaria para a regra à qual o elemento estaria submetido. Isto, no entanto, pressupõe que cada elemento da cadeia só esteja submetido a uma única regra.

Nosso modelo elimina este atributo e confere a responsabilidade de identificar os elementos aos quais uma regra será aplicada à própria definição da regra, como ocorre em sistemas ativos. Dessa forma, uma regra pode ser definida para agir sobre um elemento específico, sobre uma classe de elementos ou sobre elementos que satisfaçam determinadas condições. Além disso, esta modificação otimiza o reuso das regras já existentes em diferentes repositórios. Proporciona ainda uma otimização no armazenamento de dados, uma vez que elimina a repetição de chaves apontando para uma determinada regra em registros pertencentes a um mesmo elemento dentro dos sumários.

Houve também mudanças na estrutura de dados de alguns repositórios e sumários. Para uma maior compatibilidade com o modelo de objetos complexos utilizado na transmissão de dados via serviços Web, foram criadas entidades representando os elementos “Produto” no Sumário de Produtos e “Serviço” no sumário de Serviços, conforme ilustrado na figura 3.5. Assim, um elemento “Produto” seria a instância de um produto definido por um “Tipo de Produto” especificado no Repositório de Produtos, e teria um conjunto de “Sumário de Produto” e “Propriedades de Produto” ligados a ele por de sua chave primária. O mesmo conceito se aplica aos demais sumários e repositórios.

Chave de identificação

Dado nosso contexto de repositórios de dados distribuídos, é necessário que as chaves primárias dos registros contêm informação suficiente para identificar seus repositórios de origem, a classe de objeto que elas identificam e o número de identificação única do objeto dentro de um repositório. A tabela 3.5 apresenta um exemplo do modelo de chave

proposta. Conforme mencionado anteriormente, nosso modelo introduz a possibilidade de monitoramento e rastreabilidade usando serviços distribuídos - enquanto Kondo pressupõe a existência de um serviço gerenciador central. Se, por um lado, o processamento distribuído pode eliminar gargalos e melhorar o desempenho global do monitoramento, por outro lado aumenta as oportunidades para quebra de segurança do conjunto de serviços. Desta forma, a opção por uma ou outra solução deve levar esses fatores em consideração.

Chave Primária do Modelo
<code>http://dominio.com/SPTracerWS/ProdRepWS?ProductType.103</code>

Tabela 3.5: Exemplo de chave de identificação.

Nela é possível identificar a URL do repositório onde o registro está armazenado (“`http://dominio.com/SPTracerWS/ProdRepWS`”), a classe do objeto (“`ProductType`”) e seu identificador único naquele repositório (“`103`”).

Existem trabalhos na literatura que tratam especificamente da questão da identificação de produtos visando a rastreabilidade, como o padrão EPC (*Electronic Product Code*) [32], que pretende substituir os atuais códigos de barras através da utilização de etiquetas RFID (*Radio-frequency identification*) [31] de baixo custo. O modelo de identificação proposto por nosso trabalho é mais simples, pois dispensa a existência de um serviço de decodificação como do EPC, mas atende às necessidades da arquitetura de rastreabilidade proposta. Pode ser facilmente substituído por um modelo mais robusto, o que ficará a cargo de trabalhos futuros nesta linha.

3.6 Modelo de Gerenciamento de Regras

3.6.1 Lógica do gerenciamento de regras

O Gerente de Regras de Qualidade é o elemento responsável pela ativação das regras instanciadas no Sumário de Regras aos dados persistidos no banco de dados local. Esta ativação é possível através da conversão destas regras em gatilhos e procedimentos que serão executados automaticamente pelo SGBD sempre que um determinado evento ocorrer.

A complexidade de definição de eventos e condições a serem verificadas por uma regra é diretamente proporcional à expressividade da linguagem utilizada para especificá-la. A literatura apresenta diversas alternativas na área linguagens de especificação de restrições de integridade, como os estudos relacionados à conversão do padrão OCL (*Object Constraint Language*) [24] em SQL de forma automática [4]. O foco de nosso trabalho é o

modelo de gerenciamento de qualidade aplicado aos dados de rastreabilidade de cadeias produtivas e não linguagens formais de especificação de restrições. Assim, optou-se por uma representação entidade-relacional mais simples para as regras de qualidade, e pelo mapeamento direto entre os atributos dessas regras e os elementos componentes da sintaxe de gatilhos e procedimentos em linguagem SQL.

As seqüência a seguir define como as regras são controladas. Seja R1 uma regra definida por um usuário, armazenada no Repositório de Regras:

Registro de Regras e Gatilhos

1. O usuário solicita o cadastramento, via interface do Módulo de Gerenciamento Global, de uma instância de R1 em seu sistema, definindo o seu período de existência da mesma.
2. O Módulo de Gerenciamento Global armazena a instância de R1 no Sumário de Regras. Cada sistema deve ter seu próprio Sumário de Regras, mas pode compartilhar um Repositório de Regras com outros participantes.
3. O Gerente de Regras verifica periodicamente a existência de novas instâncias de regras a serem aplicadas ou alterações a serem realizadas nas regras já existentes. Por exemplo, se o prazo de validade da instância de R1 é alterado para que ela não seja mais verificada.
4. Cada instância de regra dá origem a um novo gatilho, no qual serão especificadas as verificações das condições determinadas pela regra. Estes gatilhos são configurados para agir a cada novo evento ocorrido nos sumários alvo, como inserções ou alterações de registros.

Caso seja necessário remover uma instância de uma regra antes de alcançado o prazo de validade estabelecido, basta que o Gerenciador Global realize a modificação do registro correspondente no Sumário de Regras. A alteração será propagada ao SGBD pelo Gerente de Regras, e as regras serão removidas. Periodicamente o Gerente de Regras verificará quais regras já expiradas continuam implantadas no SGBD como gatilhos SQL e os removerá para evitar a sobrecarga do banco de dados. Como as regras já possuem uma instância por um determinado período de existência, os gatilhos são implementados de forma a não mais atuarem após expirado seu o prazo de validade. Isto evita que o Gerente de Regras tenha que monitorar o Sumário de Regras continuamente para verificar quais regras já expiraram.

Verificação de qualidade

1. Encontrada uma violação, o gatilho verificador executará um procedimento para registrar a violação no Sumário de Regras local.
2. A violação armazenada dará origem a uma notificação, que será enviada aos demais participantes da cadeia interessados em recebê-la. Quando a violação for originada a partir da análise de dados de sensores, apenas uma violação será criada para todo o conjunto regra-sensor (ao invés de uma violação a cada leitura do sensor).
3. Após o registro, será invocado o Gerente do Regras para que seja providenciada a disseminação dos problemas verificados.
4. O Gerente de Regras enviará as notificações via invocações de serviço Web ao Módulo de Disseminação de Eventos, que se encarregará de distribuir estas mensagens aos demais participantes da cadeia interessados em recebê-las.

3.6.2 Conversão de Regras

A conversão de regras do modelo relacional em gatilhos SQL é baseada em informações provenientes da instância da regra armazenada no Sumário de Regras e da classe da regra definida no Repositório de Regras. O diagrama de seqüência 3.6 mostra as atividades e módulos envolvidos no processamento das regras. Toda regra inserida no Sumário de Regras é transformada em um gatilho que é armazenado no SGBD, o qual é acionado durante as verificações de violação. Se a instância é alterada, o gatilho é reconstruído.

O Gerente de Regras interage periodicamente com o Sumário de Regras local para obter instâncias de regras inseridas ou modificadas. Cada uma das regras recuperadas que satisfaça estas condições são processadas individualmente. Para cada regra, o Gerente de Regras busca no Repositório de Regras respectivo informações relativas ao seu tipo e condições por ela avaliada. Estas informações são convertidas em scripts de criação de gatilhos e procedimentos, posteriormente executados no SGBDA local. Aplicada a regra, o Gerente de Regras modifica os sinalizadores da mesma no Sumário de Regras para evitar que seja processada novamente, e começa a conversão da próxima regra disponível, até que todas tenham sido aplicadas.

Diferentes SGBDs utilizam formas distintas de implementar gatilhos e procedimentos. Visando simplificar as conversões e ampliar a gama de SGBDs compatíveis com o modelo, nossa proposta envolve uma etapa de conversão intermediária das regras, conforme apresentado na figura 3.7.

A primeira etapa da conversão (3.7-a) visa obter todas os dados necessários para a criação dos scripts. O objeto Rule do Sumário de Regras contem o identificador da

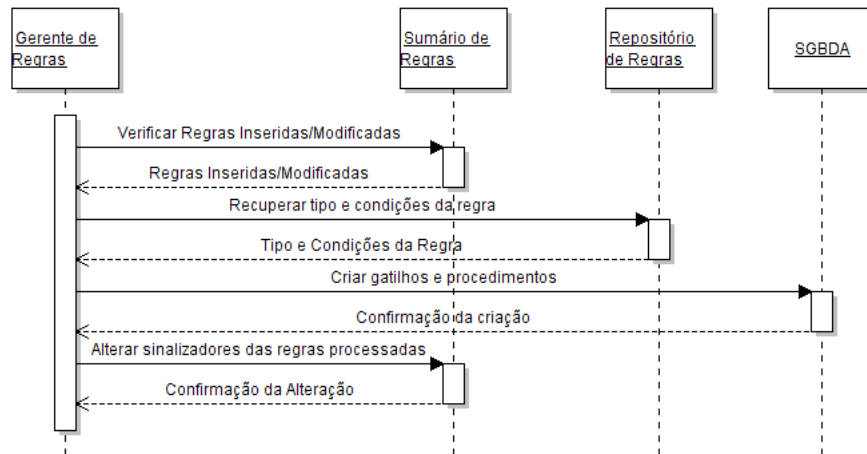


Figura 3.6: Diagrama de Seqüência de atividades da conversão de regras.

regra naquele sumário (id_regra), o tempo de validade da regra, o tipo de objeto a ser monitorado e o identificador da classe da regra que aquela instância representa. Esta última informação permite que o Gerente de Regras obtenha o Tipo da Regra ($RuleType$) e as Condições por ela verificadas no Repositório de Regras correspondente.

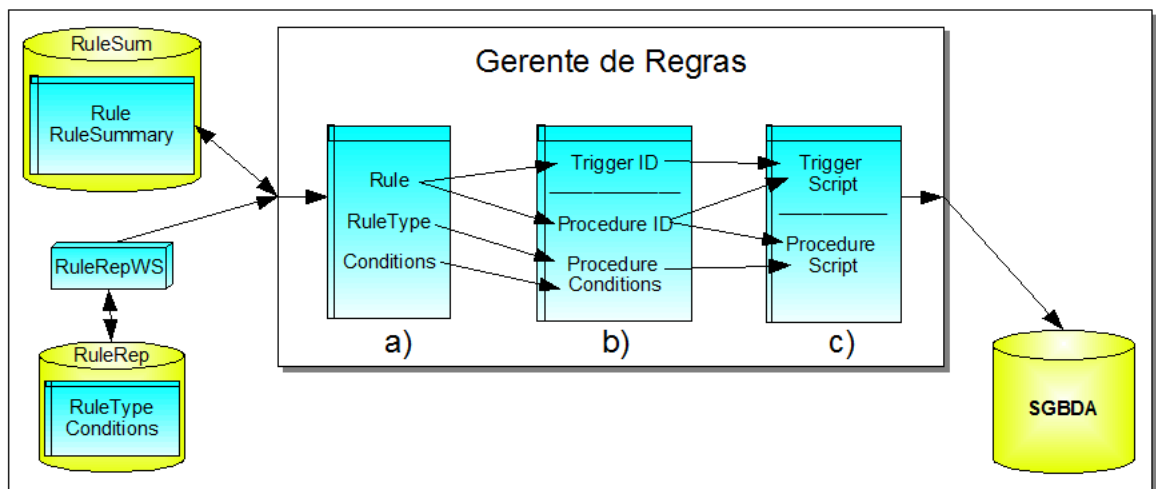


Figura 3.7: Diagrama de conversão de regras em linguagem procedural.

Na segunda etapa, as informações são mapeadas em um conjunto de elementos do padrão PL/SQL (*Procedural Language/Structured Query Language*) para a especificação de gatilhos e procedimentos. As informações necessárias para o mapeamento são:

- ID do Gatilho: Identificador do objeto *Rule* (Sumário de Regras);

- ID do Procedimento: Identificador do objeto *Rule* (Sumário de Regras);
- Data de Início: Atributo do objeto *Rule* (Sumário de Regras);
- Data de Fim: Atributo do objeto *Rule* (Sumário de Regras);
- Objeto alvo (relação): Atributo do objeto *RuleType* (Repositório de Regras);
- Condições: Atributos do objeto *Condition* (Repositório de Regras);

A terceira etapa finaliza a conversão, montando os scripts de criação dos gatilhos e procedimentos utilizando os dados coletados, considerando as peculiaridades da implementação do SGBD utilizado. Tomando como referência o padrão PL/pgSQL utilizado pelo SGBD PostgreSQL como linguagem procedural, a sintaxe final de gatilhos e procedimentos é definida como ilustrado nos códigos-fonte 3.6.2 e 3.6.2.

Código Fonte 3.1: Sintaxe de um gatilho em PL/pgSQL.

```

1 CREATE [OR REPLACE] TRIGGER <ID do Gatilho>
2 {BEFORE|AFTER} {INSERT|DELETE|UPDATE} ON <Objeto alvo>
3 FOR EACH ROW EXECUTE PROCEDURE <ID do Procedimento>;

```

Código Fonte 3.2: Sintaxe de um procedimento ou função em PL/pgSQL.

```

1 CREATE OR REPLACE FUNCTION <ID do Procedimento>
2 BEGIN
3     IF <Data de Inicio> < now() AND
4        <Data de Fim> > now() AND
5        <Expressao> THEN
6         EXECUTE <Procedimento de Notificacao>(<Parametros>);
7     END IF;
8     END LANGUAGE plpgsql;

```

A *Expressao* descrita na linha 5 do código fonte 3.2 será composta pela conjunção de expressões que verificam cada uma das condições definidas pelo registro no Repositório de Regras. Já o *Procedimento de Notificacao* descrito na linha 6 realizará a inserção de um registro na base de notificações, a qual será posteriormente enviada ao módulo de disseminação pelo Gerente de Regras. O código fonte 3.3 descreve o processo de armazenamento desta notificação.

Código Fonte 3.3: Procedimento padrão de criação de notificações.

```

1 CREATE OR REPLACE FUNCTION <Procedimento de Notificacao>
2 (<Parametros>)
3 BEGIN
4     INSERT INTO

```

```

5      RULESUMMARY (    Data ,
6                      Tipo ,
7                      Id_objeto ,
8                      Condicao ,
9                      Descricao ,
10                     Flag_nova)
11
12      VALUES          (    now() ,
13                          <Conformidade|Violacao> ,
14                          <Objeto alvo> ,
15                          <Condicao> ,
16                          <Descricao> ,
17                          true);
END LANGUAGE plpgsql;

```

A divisão da conversão nessas três etapas permite uma maior modularidade do processo, com maior flexibilidade de escolha dos padrões de definição de regras e do banco de dados utilizado.

3.6.3 Detecção de violações

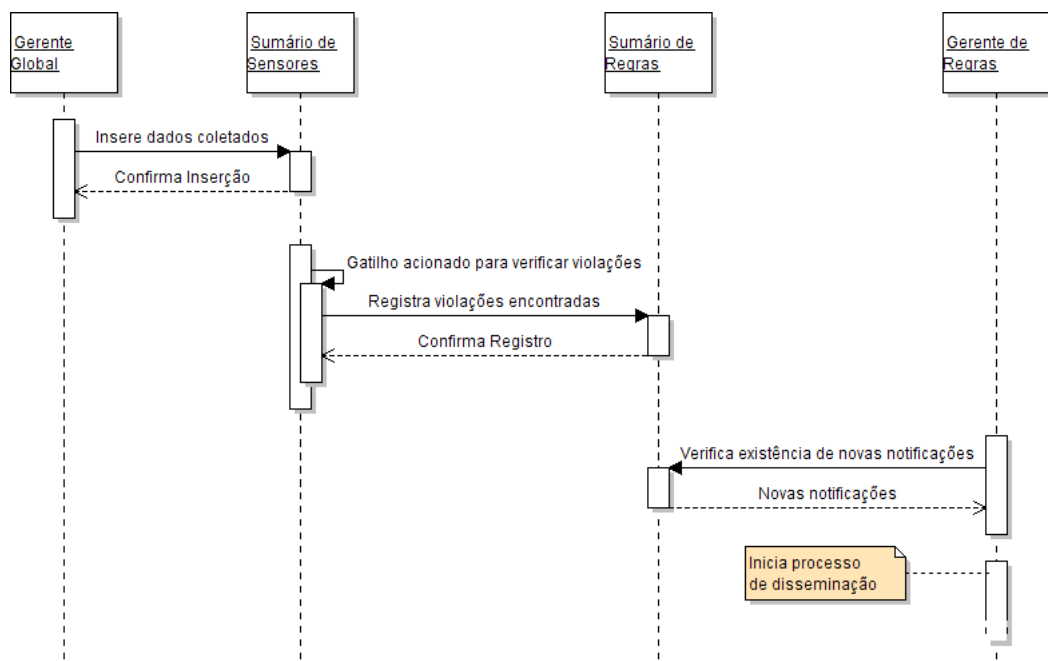


Figura 3.8: Diagrama de Seqüência de Detecção de Violações.

O diagrama de seqüência 3.8 ilustra as atividades envolvidas na detecção de violações por meio da análise de dados de sensores. O mesmo princípio se aplica para regras que

monitorem os dados de outros sumários, como os de produtos, serviços e processos. O processo de disseminação de notificações é descrito na seção 3.7.

Visando simplificar a implementação do módulo e reduzir a complexidade do sistema, o módulo de Gerenciamento de Qualidade foi totalmente concebido como sendo sem estado (*stateless*), ou seja, funciona baseado apenas em informações provenientes de outros módulos, não possuindo qualquer forma de armazenamento de dados específicos para seu funcionamento. O Sumário de Regras deve armazenar informações transacionais sobre quais regras e notificações já foram processadas pelo Gerente de Regras local através de sinalizadores (*flags*) em seus respectivos registros. Esta é a razão para a existência dos campos “flag_altera” e “flag_aplica” nos registros de instâncias de regras (Tabela 3.3(a)) e do campo “flag_nova” no registro de notificações (Tabela 3.3(b)), ambos elementos componentes do Sumário de Regras.

3.7 Modelo de Disseminação de Eventos

O controle de qualidade entre participantes de uma cadeia produtiva lógica e geograficamente distribuída requer um canal de comunicação confiável, que permita o intercâmbio de informações entre os sistemas de forma ágil e segura. Nosso trabalho propõe a existência de um dispositivo de disseminação de eventos baseado no paradigma *Publish/Subscribe* para a distribuição das notificações de qualidade originadas em diferentes repositórios de dados.

A figura 3.9 ilustra a seqüência de atividades necessárias para o registro de novas notificações no módulo de disseminação. Primeiramente o Gerente de Regras consulta o Sumário de Regras local sobre a existência de novas notificações armazenadas. Se existirem, o Gerente de Regras inicia a comunicação via serviço Web com o módulo de disseminação, requisitando o registro de cada uma delas. Ao receber a confirmação do registro, o Gerente de Regras determina que o Sumário de Regras marque a notificação registrada como processada, a fim de evitar sua posterior reavaliação.

O paradigma *Publish/Subscribe* exige a existência de “produtores” e “consumidores” de informação no canal de comunicação. Em nosso modelo, o papel de produtor é executado pelo Gerente de Regras, conforme descrito anteriormente. Os consumidores são os Gerentes Globais, responsáveis pela coordenação da cadeia e envio de informações para o módulo de interface com o usuário do sistema. A estes gerentes cabe a responsabilidade de buscar soluções para as violações identificadas, que podem variar desde uma simples notificação ao usuário até uma reconfiguração dos contratos e planos de coordenação que regem o funcionamento da cadeia produtiva afetada.

O diagrama apresentado na figura 3.10 complementa a figura 3.9, descrevendo as atividades envolvidas na declaração de interesse de um Gerente Global em determinadas

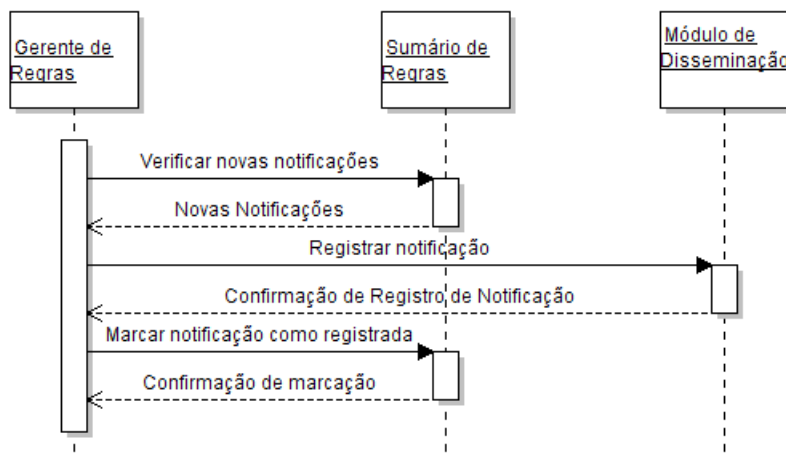


Figura 3.9: Diagrama de Seqüência: Registro de Notificações.

notificações, no envio destas notificações ao canal de disseminação pelo Gerente de Regras e na disseminação aos participantes cadastrados para recebê-las.

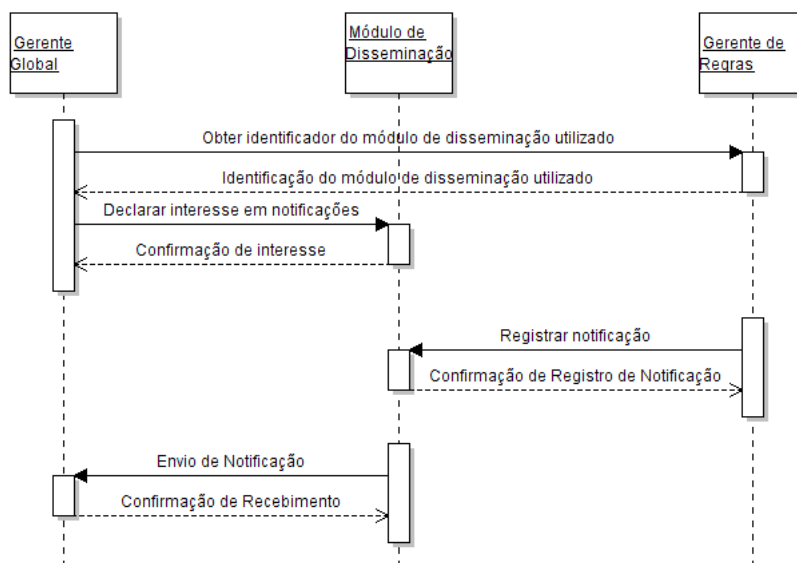


Figura 3.10: Diagrama de Seqüência: Subscrição e distribuição.

Cada participante deve especificar e disponibilizar em seu Gerente de Regras o endereço para o módulo de disseminação que irá utilizar, permitindo que suas notificações possam ser recebidas pelos demais participantes da cadeia. Por meio da arquitetura de rastreabilidade, os Gerentes Globais conseguem identificar automaticamente potenciais repositórios que podem vir a originar notificações de seu interesse, atribuindo assim

maior agilidade ao processo de gerenciamento de qualidade.

A filtragem de notificações neste modelo se restringirá apenas ao identificador do tipo da regra que originou a notificação, ao identificador da classe de objeto alvo da regra e ao identificador do objeto que originou a notificação.

Este trabalho propõe apenas a utilização do modelo de disseminação de eventos como uma ferramenta para o gerenciamento de qualidade em cadeias produtivas, não se aprofundando nos conceitos envolvidos nesta área de pesquisa.

3.8 Conclusões

Este capítulo apresentou o modelo proposto para o gerenciamento de qualidade em cadeias produtivas, baseado na combinação de repositório de regras, um mecanismo ativo de monitoramento destas regras sobre os demais dados da cadeia produtiva e um mecanismo de disseminação de eventos. O capítulo 4 apresenta detalhes de implementação.

Capítulo 4

Aspectos de implementação

Este capítulo apresenta a implementação do protótipo do Gerenciador de Qualidade em Cadeias Produtivas, que denominamos SPTracer. A seção 4.1 enumera as tecnologias utilizadas no desenvolvimento. A seção 4.2 descreve os detalhes do repositório de dados construído. A implementação dos serviços Web utilizados é elucidada na seção 4.3, enquanto as seções 4.4 e 4.5 descrevem, respectivamente, a construção do gerenciador de regras e do módulo de disseminação de eventos. A seção 4.6 detalha a implementação do módulo de gerenciamento global e das interfaces de usuário. A seção 4.7 conclui o capítulo.

4.1 Tecnologias Utilizadas

A linguagem de programação selecionada para implementar o protótipo foi Java 6.0 (*Java SE Development Kit 6*) [39], em virtude de sua simplicidade, modularidade e também nosso conhecimento prévio. O ambiente de programação escolhido foi o Eclipse IDE for Java EE Developers [19] com suporte ao servidor de aplicações JBoss e ferramentas de desenvolvimento Web WTP (*Web Tools Platform*).

Toda a arquitetura se baseia no uso de serviços Web. Assim, um servidor de aplicação JBoss 4.2.1.GA [34] foi utilizado para funcionar como container do protótipo, além de prover as funcionalidades de servidor Web, servidor de mensagens JMS e serviço de persistência de dados de forma integrada. As interfaces de usuários foram implementadas usando JSP (JavaServer Pages).

O JBoss utiliza em seu núcleo uma versão nativa do Apache Tomcat 6.0 [53] como servidor Web para disponibilização de conteúdo e dos pontos de conexão dos serviços Web. Provê ainda o JBoss Messaging, uma implementação compatível com o padrão JMS para troca de informações. Este é o padrão usado no mecanismo de notificação do protótipo.

O SGBD utilizado para persistência foi o PostgreSQL [46], em virtude de sua simplici-

dade de operação e principalmente de sua fácil integração com o mecanismo de persistência de dados utilizado (Hibernate 3.0 [29]).

A figura 4.1 ilustra em quais módulos do protótipo as tecnologias descritas foram aplicadas.

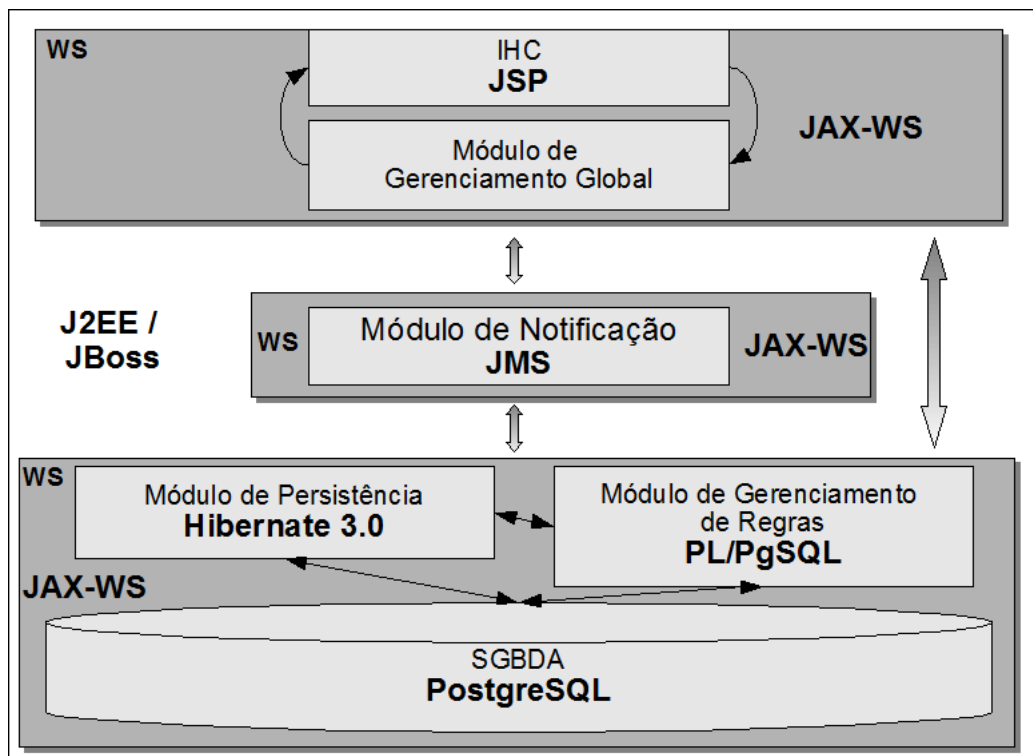


Figura 4.1: Tecnologias utilizadas.

A seguir exploraremos os detalhes de implementação em cada um dos componentes do protótipo elaborado.

4.2 Repositórios de dados

Os repositórios de dados foram implementados de maneira modular, visando possibilitar a utilização de cada um deles de maneira independente dentro de um mesmo sistema em um participante da cadeia. Isto significa que, apesar de todos os repositórios de dados estarem disponíveis dentro de um mesmo pacote, seus arquivos de configuração permitem selecionar apenas os módulos necessários para uma aplicação, conforme veremos a seguir.

Nosso modelo trabalha com classes anotadas com informações de persistência segundo o padrão EJB 3.0 (*Enterprise Java Beans*). A implementação do EJB 3.0 do servidor JBoss minimiza o esforço de persistência de objetos em bancos de dados relacionais. Este

modelo se baseia no uso de anotações da linguagem Java introduzidas a partir de sua versão 5.0 e utiliza o Hibernate 3.0 [29] como mecanismo de motor de persistência das classes anotadas. Desta forma é possível fazer a ligação entre classes e seus respectivos esquemas em um banco de dados utilizando apenas anotações no código das mesmas, bem como gerar automaticamente o esquema completo do banco de dados a partir de um conjunto de classes anotadas.

O código 4.1 apresenta um exemplo de classe anotada com os parâmetros de persistência. Nele, a classe *Product* é definida com alguns parâmetros básicos, como descrição do produto (*description*), conjunto de propriedades (*properties*) e tipo de produto *productType*.

Código Fonte 4.1: Exemplo do uso de anotações pelo Hibernate.

```
1 package br.unicamp.ic.lis.SPTracer.Summary;
2 import javax.persistence.*;
3 @Entity
4 public class Product extends SPTracerRegistry{
5     @Column
6     String description;
7
8     @OneToMany(cascade=CascadeType.ALL,fetch=FetchType.EAGER)
9     @JoinColumn(name="product_id")
10    Set<ProductProperty> properties = null;
11
12    @Column(nullable = false)
13    private String productType;
14
15    @OneToMany(cascade=CascadeType.ALL,fetch=FetchType.EAGER)
16    @JoinColumn(name="product_id")
17    Set<ProductSummary> summary = null;
18
19    public Product() {
20        super();
21    }
22
23    public Set<ProductProperty> getProperties() {
24        return properties;
25    }
26 }
```

A ligação entre as classes e suas respectivas tuplas na base de dados é realizada automaticamente pelo Hibernate em função de anotações adicionadas ao corpo da classe. Tais anotações são precedidas do símbolo “@” e fazem parte de um vocabulário específico do mecanismo de persistência. Os parâmetros que definem uma entidade (*@Entity*), seus atributos (*@Column*), relacionamentos (*@OneToMany*, *@JoinColumn*) e restrições

(*nullable = false*) são interpretados e utilizados pelo Hibernate tanto para criação quanto para a validação do esquema de dados e operações sobre ele executadas.

Tomemos como exemplo a utilização da anotação *@OneToMany* (linha 15) no atributo *properties*. Esta anotação define a cardinalidade um para muitos entre a entidade *Product* e outra entidade usando o atributo *product_id* para isto, além de conter parâmetros que especificam a forma como eventos sobre a entidade principal devem refletir sobre a entidade dependente (*cascade=CascadeType.ALL*) e se a inserção da classe principal na base de dados deve ou não acarretar na inserção das entidades dependentes ainda não persistidas (*fetch=FetchType.EAGER*). Por sua vez, a anotação *@JoinColumn* (linha 16) especifica o nome da coluna na tabela de propriedades que irá armazenar a chave estrangeira relacionada ao produto (*name= "product_id"*).

O reconhecimento da classe como uma entidade pelo Hibernate depende ainda de sua existência no arquivo de configuração do mesmo, conforme exemplo no código 4.2.

Código Fonte 4.2: Exemplo de configurações utilizadas pelo Hibernate.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate_
  Configuration_3.0//EN"
3 "http://hibernate.sourceforge.net/hibernate-configuration
  -3.0.dtd">
4 <hibernate-configuration>
5   <session-factory>
6     <property name="hibernate.connection.driver_class">org.
      postgresql.Driver</property>
7     <property name="hibernate.connection.url">
      jdbc:postgresql:public</property>
8     <property name="hibernate.connection.username">sptracer</
      property>
9     <property name="hibernate.connection.password">123</
      property>
10    <property name="hibernate.default_schema">public</property>
11    <property name="hibernate.dialect">org.hibernate.dialect.
      PostgreSQLDialect</property>
12    <mapping class="br.unicamp.ic.lis.SPTracer.Summary.Product
      "/>
13    <mapping class="br.unicamp.ic.lis.SPTracer.Summary.
      ProductSummary"/>
14  </session-factory>
15 </hibernate-configuration>

```

Nele é possível identificar os parâmetros de configuração da conexão com o banco de dados, como driver (linha 6), string de conexão (linha 7), o dialeto SQL a ser utilizado (linha 11) e a indicação de quais classes devem ser persistidas (linhas 12 e 13).

Mudanças nesta parte do código permitem selecionar quais tabelas devem ser materializadas em repositórios pelo Hibernate.

Operação	Descrição	Parâmetro
insertObjectOnDB()	Inserir objeto na base de dados	Objeto a ser inserido
updateObjectOnDB()	Atualiza objeto na base de dados	Objeto a ser atualizado
deleteObjectOnDB()	Remove objeto da base de dados	ID do objeto a ser removido
getObjectFromDB()	Instância um objeto a partir dos dados do banco de dados	ID do objeto desejado
query()	Executa uma consulta na base de dados	String de consulta

Tabela 4.1: Tabela de operações do *DataAccessControler*.

Todo o controle de acesso aos dados foi centralizado em apenas uma classe (*DataAccessControler*) que implementa as cinco operações básicas a serem realizadas sobre os objetos persistidos. Estas operações estão enumeradas na tabela 4.1. Estas operações são utilizadas diretamente pelas classes que implementam as funcionalidades dos serviços Web, conforme descrito na seção 4.3.

4.3 Serviços Web

A construção e disponibilização dos serviços Web utiliza o padrão JAX-WS 2.0[8] de especificação. JAX-WS (*Java API for XML Web Services*) é uma arquitetura de mais fácil compreensão para o desenvolvimento de serviços Web, e vem sendo adotada na substituição ao antigo padrão JAX-RPC (*Java API for XML-Based RPC*). Suas principais vantagens são uma separação mais clara entre a definição do serviço Web e seus componentes de interligação de dados, além de uma considerável simplicidade de uso.

JAX-WS suporta o uso extensivo de anotações no código JAVA que implementa as funcionalidades do serviço Web a ser disponibilizado. Isto possibilita a geração automática de toda a estrutura necessária para o funcionamento do serviço: o arquivo WSDL correspondente, as estruturas em XSD (*XML Schema*) das classes utilizadas como portadoras de informação em suas operações e os detalhes dos pontos de comunicação (*endpoint*) do serviço.

O código 4.3 mostra um trecho do código de uma das classes que implementam os serviços Web utilizados.

Código Fonte 4.3: Exemplo de implementação de um serviço Web.

```
1 package prodsum;
2 import javax.jws.WebMethod;
3 import javax.jws.WebService;
4 import javax.jws.soap.SOAPBinding;
5 @WebService(name = "ProdSumWS",
6             targetNamespace = "http://sptracer/ProdSumWS")
7
8 @SOAPBinding(style = SOAPBinding.Style.DOCUMENT,
9             use = SOAPBinding.Use.LITERAL,
10            parameterStyle = SOAPBinding.ParameterStyle.WRAPPED)
11
12 public class ProdSumWS {
13
14     @WebMethod()
15     public String insert(Product p) {
16         p.setHostURL(ServerInfo.getInstance()
17                 .getHostServerURL(this));
18         DataAccessControler.getInstance().insertObjectOnDB(p
19                 );
20         return p.getFullID();
21     }
22 }
```

No modelo de Kondo[35], os serviços Web disponibilizam métodos que traduzem operações a serem realizadas diretamente na base de dados. Nosso protótipo, por outro lado, utiliza uma abordagem baseada em objetos complexos para a transferência de dados, ou seja, ao invés dos métodos dos serviços Web receberem como parâmetro os dados explícitos que serão utilizados para manipulação no banco de dados, nosso protótipo utiliza objetos completos como parâmetros. Isso facilita a compreensão e implementação do sistema, pois modificações na estrutura de dados são imediatamente refletidas na interface dos serviços Web. Estes objetos são gerados automaticamente pelo JAX-WS a partir da análise das classes e subclasses utilizadas como parâmetros nas operações do serviço Web. Tais classes têm suas estruturas mapeadas em linguagem XSD e disponibilizadas junto ao arquivo WSDL do serviço correspondente.

Os esqueletos dos clientes dos serviços Web utilizados foram construídos usando a ferramenta *Web Tools Platform* do Eclipse, a qual gera toda a estrutura necessária para comunicação com um determinado serviço Web através da análise de seu descritor WSDL.

A figura 4.2 ilustra um modelo básico da comunicação através dos pacotes que com-

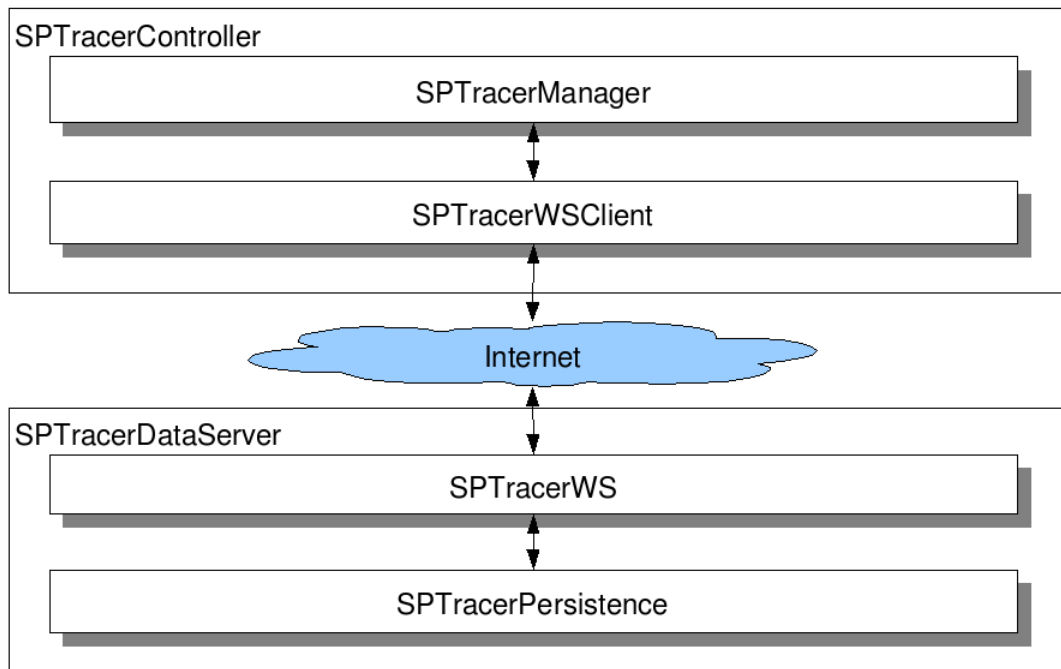


Figura 4.2: Esquema básico de comunicação do SPTracer via serviços Web

põem o protótipo. Nela observamos como a camada de serviços Web (clientes e servidores) atua como *middleware* de comunicação entre os componentes de persistência e lógica da arquitetura. O *SPTracerWsClient* é portanto o componente a ser utilizado por qualquer aplicação que deseje se comunicar com o repositório de dados de maneira simples e padronizada.

4.4 Módulo de Gerenciamento de Regras

O Módulo de Gerenciamento de Regras é o principal componente de nossa arquitetura, sendo responsável por todo o processo de controle das regras de qualidades aplicadas ao SGBD, conforme descrito na seção 3.6.

Seu funcionamento se baseia em consultas periódicas ao Sumário de Regras, verificando se existem regras ou notificações a serem processadas. É dividido em dois módulos principais, descritos a seguir.

4.4.1 Módulo de Controle de Regras

O Módulo de Controle de Regras (*RuleMgr*) é o responsável pela seleção, aplicação e remoção das regras no SGBD. Tomemos como exemplo a regra detalhada na tabela 4.3,

onde o formato dos identificadores foi simplificado. As etapas realizadas por este módulo ocorrem da seguinte forma:

Repositório de Regras - Classes			
id_regratp	descrição	origem	objeto
RuleType.1	Controle de temperatura do leite	Portaria CVS 15	ProductType.1

(a)

Repositório de Regras - Cláusulas					
id_regra	parametro	operador	valor	unidade	precisao
RuleType.1	Temperatura	<	10	Celsius	0.001
RuleType.1	Temperatura	>	1,5	Celsius	0.001

(b)

Sumário de Regras - Instâncias de Regras							
id_regra	tipo	objeto_tipo	objeto_id	início	fim	flag_altera	flag_aplica
Rule.1	RuleType.1	ProductType.1		12/01/08 00:00	12/01/09 00:00	true	false

(c)

Tabela 4.2: Exemplo de regra a ser processada pelo Módulo de Controle de Regras.

1. **Seleção:** Um consulta ao Sumário de Regras local seleciona todas as regras que tenham sido modificadas ($FLAG_ALTERA=TRUE$) ou que estejam aplicadas no SGBD e tenham expirado ($FLAG_APLICA=TRUE AND RULEEND < NOW()$).
2. **Aplicação:** As regras alteradas (incluídas ou modificadas no Sumário de Regras) e com prazo de aplicação válido são processadas e dão origem aos gatilhos e procedimentos necessários para o controle de violações realizado pelo SGBDA. O código 4.4 exibe o script em linguagem PL/pgSql (*PostgreSQL Procedural Language*) resultante da conversão da regra da tabela 4.3. Em uma mesma transação, os scripts são executados no SGBDA e os *flags* da regra inserida são alterados para que a regra não seja processada novamente ($FLAG_ALTERA=FALSE$).

Código Fonte 4.4: Exemplo do uso regra em formato PL/pgSQL

```

1 CREATE OR REPLACE FUNCTION "RULE.1"() RETURNS "TRIGGER" AS
  $BODY$
2 DECLARE
3   VIOLA BOOLEAN;
4   CONDITIONS TEXT = '';
5   TESTE BOOLEAN;
6   T2_SENSOR SENSOR%ROWTYPE;
7 BEGIN
8   IF NOT('2009-02-28_11:33:05-03' <= NOW() AND NOW() <= '
      2009-02-28_12:06:25-03') THEN RETURN NEW;
9   END IF;
10  SELECT * INTO T2_SENSOR FROM SENSOR WHERE SENSOR.ID = NEW.
      SENSOR_ID;
11  IF (T2_SENSOR.TARGETTYPE <> 'PRODUCT.1') THEN RETURN NEW;

```

```

12 END IF;
13 VIOLA = FALSE;
14 IF ( (NEW.ATTRIBUTE='TEMPERATURE') AND (NEW.VALUE <10 ) ) THEN
15     TESTE = TRUE;
16 ELSE    VIOLA = TRUE;
17     CONDITIONS = CONDITIONS || '2) TEMPERATURE <10';
18 END IF;
19 IF ( (NEW.ATTRIBUTE='TEMPERATURE') AND (NEW.VALUE >1 ) ) THEN
20     TESTE = TRUE;
21 ELSE    VIOLA = TRUE;
22     CONDITIONS = CONDITIONS || '1) TEMPERATURE >1';
23 END IF;
24 IF (VIOLA = TRUE) THEN PERFORM STORERULESUMMARY(1,NEW.
25     SENSOR_ID, CONDITIONS);
26 END IF;
27 RETURN NEW;
28 END;
29 $BODY$ LANGUAGE PLPGSQL;
30
31 CREATE TRIGGER "RULE.1" AFTER UPDATE ON SENSORSUMMARY FOR EACH
32     ROW EXECUTE PROCEDURE "RULE.1"();

```

A primeira verificação realizada pelo procedimento é o prazo de validade da regra (linha 8), a fim de evitar a avaliação de dados fora do prazo especificado pelo usuário enquanto a regra não é removida pelo módulo de controle de regras. O próximo passo é verificar se o tipo do objeto alvo desta regra é igual ao do sensor (linha 10). Além do tipo do objeto é possível especificar um identificador de um objeto específico, tanto na regra quanto no sensor, mas esta informação não consta na regra da tabela 4.3, tornando-a mais abrangente. Verificada a compatibilidade, é iniciada a análise dos predicados especificadas pelo modelo da regra (linha 13-21). A informação sobre a condição violada é armazenada em uma variável (*conditions*). Em havendo violações, um procedimento auxiliar é executado para criação dos registros de violação no Sumário de Regras (linha 22). Este procedimento é mostrado no código 4.5.

Código Fonte 4.5: Função que armazena o registro de violação no Sumário de Regras.

```

1 CREATE OR REPLACE FUNCTION STORERULESUMMARY(REGID BIGINT,
2     SENSOR_ID BIGINT, CONDIT TEXT)
3 RETURNS VOID AS $BODY$
4 DECLARE
5     T2_SENSOR SENSOR%ROWTYPE;
6 BEGIN
7     SELECT * INTO T2_SENSOR FROM SENSOR WHERE ID = SENSOR_ID;

```



```

7      INSERT INTO RULESUMMARY (RULE_ID ,HOSTURL ,TYPE ,CONDITION ,
      REG_CREATION ,SENSORFULLID ,FLAG_NEW) VALUES (REGID ,
      T2_SENSOR.HOSTURL , 'VIOLATION' ,CONDIT ,NOW() ,T2_SENSOR.
      FULLID ,TRUE) ;
8  END ;
9  $BODY$ LANGUAGE 'PLPGSQL' ;

```

3. **Remoção:** As regras aplicadas e expiradas têm seus respectivos gatilhos e procedimentos removidos do SGBD. Em uma mesma transação, o script de remoção é executado e as *flags* de controle da regra são modificadas para que a regra não seja processada novamente (FLAG_APLICA=FALSE e FLAG_ALTERA=FALSE). O código 4.6 remove o gatilho e procedimento da regra *Rule.1* do SGBDA.

Código Fonte 4.6: Remoção de uma regra no SGBDA.

```

1  DROP TRIGGER "RULE.1"() ON SENSORSUMMARY ;
2  DROP FUNCTION "RULE.1"() ;

```

Este processamento finaliza a etapa de detecção de violações de qualidade. A partir deste processamento, as violações detectadas já estarão disponíveis para eventuais consultas de atores externos a estas informações. No entanto, o modelo especifica uma ferramenta de disseminação de eventos para a distribuição das informações geradas, alimentado a partir da execução do módulo de controle de notificações descrito a seguir.

4.4.2 Módulo de Controle de Notificações

O Módulo de Controle de Notificações é responsável por enviar as notificações geradas pelo SGBDA ao Módulo de Disseminação de Eventos (seção 3.7). As notificações são automaticamente criadas pelo SGBDA a cada registro de violação armazenado no Sumário de Regras (código 4.5). No entanto, cada conjunto de leituras de um sensor processado pelo SGBDA pode dar origem a dezenas de registros de violações relacionadas a uma mesma regra. Assim, com o intuito de evitar que esses registros originem uma grande quantidade de notificações redundantes na arquitetura, apenas uma notificação é gerada para um par regra-sensor. O código 4.7 apresenta o script de criação de notificações. Apesar disto, caso seja de interesse, é possível consultar no banco de dados todos os registros de eventos de sensores. Desta forma, não há informação perdida.

Código Fonte 4.7: Criação de notificações a partir do registro de violações.

```

1  CREATE OR REPLACE FUNCTION STORERULENOTIFICATION() RETURNS "
      TRIGGER" AS $BODY$
2  DECLARE

```

```

3      T2_RULE RULE%ROWTYPE;
4      T2_NOTIFICATION RULENOTIFICATION%ROWTYPE;
5  BEGIN
6      SELECT ID INTO T2_NOTIFICATION FROM RULENOTIFICATION WHERE
          RULE_ID = NEW.RULE_ID AND SENSORFULLID = NEW.SENSORFULLID
          AND FLAG_NEW = TRUE;
7      IF (T2_NOTIFICATION.ID > 0 ) THEN RETURN NEW;
8      ELSE SELECT * INTO T2_RULE FROM RULE WHERE RULE.ID=NEW.RULE_ID;
9      INSERT INTO RULENOTIFICATION(RULE_ID,HOSTURL ,RULEFULLID ,
          SENSORFULLID ,TARGETTYPE ,TARGETID ,FLAG_NEW ,REG_CREATION)
10     VALUES (NEW.RULE_ID ,T2_RULE.HOSTURL ,T2_RULE.FULLID ,NEW .
          SENSORFULLID ,T2_RULE.TARGETTYPE ,T2_RULE.TARGETID ,TRUE ,NOW())
          ;
11     END IF;
12     RETURN NEW;
13 END;
14 $BODY$ LANGUAGE 'PLPGSQL';
15
16 CREATE TRIGGER STORENEWRULENOTIFICATION AFTER INSERT ON RULESUMMARY
          FOR EACH ROW EXECUTE PROCEDURE STORERULENOTIFICATION();

```

Um vez criadas, as notificações são armazenadas como registros no Sumário de Regras, conforme exemplificado na tabela 4.3. O envio das notificações ao Módulo de Disseminação de Eventos ocorre em 3 etapas distintas:

Sumário de Regras - Instâncias de Regras							
id_regra	tipo	objeto_tipo	objeto_id	início	fim	flag_altera	flag_aplica
Rule.1	RuleType.1	ProductType.1		12/01/08 00:00	12/01/09 00:00	true	false

(a)

Sumário de Regras - Resultados das análises							
id_regra	tipo	data	id_objeto	condição	sensor_id	medida_id	flag_nova
Rule.1	Violação	12/01/08 00:00		Temperatura < 10	Sensor.1	SensorSummary.545	true
Rule.1	Violação	12/01/08 00:00		Temperatura < 10	Sensor.1	SensorSummary.623	true

(b)

Sumário de Regras - Notificações					
id_regra	data	objeto_tipo	objeto_id	sensor_id	flag_nova
Rule.1	12/01/08 00:00	ProductType.1		Sensor.1	true

(c)

Tabela 4.3: Exemplo de notificação a ser processada pelo Módulo de Controle de Notificações.

1. **Seleção:** Um consulta ao Sumário de Regras local seleciona todas as notificações que tenham sido adicionadas e não processadas ($FLAG_NOVA=TRUE$).
2. **Preparação:** Conforme descrito na seção 3.7, a interface do Módulo de Disseminação de Eventos com os demais módulos do sistema é baseada em serviços Web.

A etapa de preparação consistem em identificar qual endereço do módulo de disseminação a ser utilizado e configurar a conexão com este serviço para o envio da notificação.

3. **Envio:** Após instanciada, a classe que representa o Módulo de Disseminação publica a notificação no módulo remoto. Em caso de sucesso, o Módulo de Controle de Notificações altera o *flag* de inclusão da notificação e finaliza sua execução (*FLAG_NOVA=FALSE*).

O ciclo de gerenciamento apenas termina quando o Módulo de Gerenciamento Global recebe as notificações geradas, distribuídas pelo Módulo de Disseminação de Eventos, cujos detalhes de implementação são descritos na seção 4.5.

4.5 Módulo de Disseminação de Eventos

O Módulo de Disseminação de Eventos funciona como um canal de comunicação comum entre diversos participantes da cadeia, conforme definido na seção 3.7. Sua implementação utiliza como base o padrão JMS (*Java Message Service*) de intercâmbio de mensagens, cuja infraestrutura de organização, controle e persistência de informações é toda fornecida pelo servidor de aplicações utilizado neste protótipo (JBoss 4.2.1.GA [34]). Assim, o papel do Módulo de Disseminação de Eventos se restringe à disponibilização dos dois serviços Web responsáveis, respectivamente, pela publicação (*PublishWS*) e pelo recebimento (*SubscribeWS*) de notificações, os quais detalharemos a seguir.

A forma de disseminação de eventos entre os módulos que interagem com o este módulo foi especificado em nosso modelo (seção 3.7) seguindo o padrão *push* de subscrição. Isto significa que os eventos são disponibilizados no canal de comunicação e posteriormente enviados aos participantes interessados sem que haja a solicitação explícita de cada entrega. No entanto, visando a simplificação de implementação, o protótipo utilizou o modelo *pull* na entrega de notificações aos Gerenciadores Globais interessados em recebê-las. Ou seja, após fazerem a subscrição, o Gerenciadores devem verificar periodicamente a existência de notificações relacionadas à suas subscrições.

4.5.1 Serviço de Publicação

O serviço Web de publicação de mensagens denominado *PublishWS* é o responsável pelo recebimento, classificação e disponibilização das notificações aos demais participantes da cadeia. Sua especificação é composta pela interface de comunicação com os publicadores de notificações (Módulos de Gerenciamento de Regras) e pela implementação das rotinas que processam e entregam a mensagem à infraestrutura JMS do servidor de aplicações.

Uma mensagem JMS é composta basicamente por um cabeçalho, propriedades e pelo seu corpo, que pode ser uma simples cadeia de caracteres ou um objeto complexo, como no caso das notificações. As propriedades devem ser especificadas como pares “nome da propriedade” e “valor”, e são utilizadas para a seleção das mensagens através de filtros baseadas em linguagem SQL. Esta característica é de grande importância para nosso modelo, pois permite que os participantes da cadeira interessados em notificações específicas (oriundas de determinada regra, tipo de objeto, objeto específico ou sensor) possam filtrar diretamente no canal de comunicação apenas as informações que desejarem receber.

A publicação é realizada no servidor de aplicações em um canal de comunicação denominado “tópico”, configurado para ser durável. Ser durável significa manter as mensagens disponíveis no canal enquanto houver uma subscrição apta a recebê-la (cujos parâmetros de seleção sejam satisfeitos pelas propriedades da mensagem), mesmo que o receptor não esteja conectado ao serviço de mensagens no instante da publicação. O código 4.8 mostra as principais informações contidas no cabeçalho de uma mensagem JMS criada a partir de uma notificação.

Código Fonte 4.8: Mensagem JMS criada a partir de uma notificação.

```
1  org.jboss.mq.SpyObjectMessage {
2  Header {
3      jmsDestination    : TOPIC.notifierTopic
4      jmsDeliveryMode  : 2
5      jmsExpiration    : 0
6      jmsPriority       : 4
7      jmsMessageID     : ID:1-12358629982101
8      jmsTimeStamp     : 1235862998210
9      jmsCorrelationID : null
10     jmsReplyTo       : null
11     jmsType          : null
12     jmsRedelivered  : false
13     jmsProperties    : {SensorFullId=Sensor.1, RuleFullId=Rule.1,
                        TargetType=ProductType.1}
14     jmsPropReadWrite: true
15     msgReadOnly     : false
16     producerClientId: ID:1
17 }}
```

É possível observar na linha 13 as propriedades desta notificação. O conteúdo da mensagem é um objeto da classe *RuleNotification*, que poderá ser utilizado pelo receptor da mesma forma como foi originalmente publicada.

4.5.2 Serviço de Recebimento

O serviço Web de subscrição e recebimento de notificações denominado *SubscribeWS* é a interface responsável pela entrega de mensagens aos participantes interessados. Essa entrega é realizada de forma passiva, ou seja, depende da verificação periódica, por parte dos receptores, da existência de novas notificações a serem recebidas. O processo de recebimento de notificações ocorre em três etapas distintas:

1. **Subscrição:** Os participantes interessados em receber determinado tipo de notificação devem realizar sua subscrição no Módulo de Disseminação de Eventos, informando um identificador único para a subscrição e os parâmetros de seleção das mensagens desejadas. Estes parâmetros devem considerar as propriedades descritas no cabeçalho da mensagem JMS, e são especificados como cláusulas SQL. Tomando como exemplo a mensagem cujo cabeçalho é descrito no código 4.8, exemplos de filtagem válidas que selecionariam esta notificação são:
 - `RULEFULLID='RULE.1'`
 - `TARGETTYPE='PRODUCTTYPE.1' AND RULEFULLID='RULE.1'`
 - `TARGETTYPE IN ('PRODUCTTYPE.1','PRODUCTTYPE.5')`
 - `TARGETID='PRODUCT.3' OR TARGETTYPE='PRODUCTTYPE.6'`

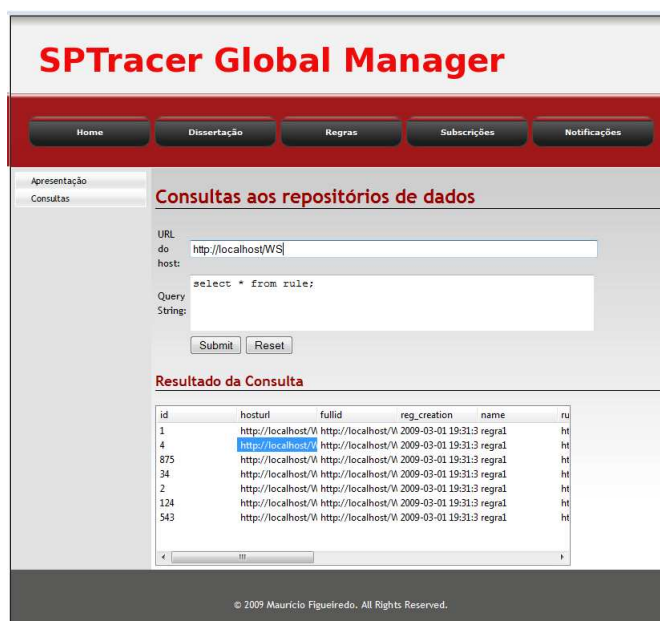
É importante ressaltar que as subscrições não são retroativas, ou seja, não permitem o recebimento de mensagens enviadas anteriormente à sua criação. Como analogia, podemos considerar as subscrições como caixas postais seletivas, que armazenam apenas o conteúdo de interesse do destinatário e são periodicamente consultadas por ele.

2. **Recebimento:** Uma vez realizada a subscrição, as mensagens que satisfizerem seus critérios de seleção serão mantidas no tópico até que o interessado em recebê-las se conecte novamente ao Módulo de Disseminação e informe o código da subscrição da qual deseja receber mensagens. Todas as mensagens selecionadas serão entregues apenas uma vez ao destinatário, mas a subscrição será mantida ativa para futuras notificações.
3. **Cancelamento da Subscrição:** O cancelamento da subscrição deve ser realizado pelo subscritor sempre que não houver mais interesse em receber notificações oriundas de determinada subscrição.

4.6 Módulo de Gerenciamento Global

O Módulo de Gerenciamento Global é o responsável por prover a interface de interação do usuário com o sistema de gerenciamento de qualidade. Sua implementação se baseia no uso de clientes de serviços Web que interagem diretamente com os sumários e repositórios da arquitetura de rastreabilidade e com o módulo de disseminação de eventos para a subscrição e recebimento de notificações.

A figura 4.3 ilustra a tela de consulta a dados de repositórios. Nela, o usuário deve informar a URL do serviço que deseja acessar e o texto de consulta em linguagem SQL. Estas informações são enviadas ao serviço Web do repositório indicado e as informações recuperadas são apresentadas na tabela de resultado.



id	hosturl	fullid	reg_creation	name	ru
1	http://localhost/V	http://localhost/V	2009-03-01 19:31:3	regra1	ht
4	http://localhost/V	http://localhost/V	2009-03-01 19:31:3	regra1	ht
875	http://localhost/V	http://localhost/V	2009-03-01 19:31:3	regra1	ht
34	http://localhost/V	http://localhost/V	2009-03-01 19:31:3	regra1	ht
2	http://localhost/V	http://localhost/V	2009-03-01 19:31:3	regra1	ht
124	http://localhost/V	http://localhost/V	2009-03-01 19:31:3	regra1	ht
543	http://localhost/V	http://localhost/V	2009-03-01 19:31:3	regra1	ht

Figura 4.3: Interface de consulta a dados de repositórios.

4.7 Conclusões

Este capítulo apresentou os principais aspectos relacionados à implementação do protótipo. Foram abordadas as particularidades de cada tecnologia utilizada, bem como suas aplicações em cada um dos componentes principais da arquitetura proposta.

Capítulo 5

Conclusões e Trabalhos Futuros

5.1 Conclusões

Esta dissertação contribuiu para a solução dos problemas relacionados ao controle de qualidade de produtos em cadeias produtivas a partir da análise de dados de rastreabilidade, propondo e implementando mecanismos para o gerenciamento e monitoramento de regras de qualidade em processos e serviços que atuam sobre estes produtos. A hipótese básica é que especialistas do domínio tenham especificado todos os critérios a serem aplicados, correspondendo às regras que definem qualidade. Para o desenvolvimento da pesquisa, foram utilizados como base trabalhos desenvolvidos na UNICAMP em rastreabilidade, serviços Web e documentação de processos, em particular à proposta de Bacarin [2] e ao modelo de rastreabilidade proposto por Kondo [35]. O trabalho envolve pesquisa em serviços Web, sensores, bancos de dados ativos e sistemas baseados em eventos, provendo mecanismos que possibilitem análises espaço-temporais de violação de restrições pré-estabelecidas.

Partindo do modelo de Kondo [35], foram propostos novos repositórios e sumários para o armazenamento de regras de qualidade e dados oriundos de sensores. Foi especificado, ainda, todo o mecanismo de controle e conversão de das regras em procedimento executáveis pelo SGBDA, além do modelo de distribuição das notificações geradas pela avaliação das restrições. A arquitetura foi totalmente implementada e testada para alguns cenários. Os testes utilizaram dados fictícios, e analisando o comportamento da arquitetura ao se inserir informações que provocassem o registro de violações; isto permitiu validar a execução de vários outros tipos de cenários.

Nosso trabalho apresenta uma nova abordagem em relação aos trabalhos correlatos por fazer uso de mecanismos ativos para a detecção de violações de qualidade e disseminação de notificações aos diversos atores que interagem com a cadeia produtiva.

A principal dificuldade enfrentada neste trabalho foi a necessidade de se encontrar uma arquitetura de gerenciamento de regras de qualidade genérica o suficiente para abranger

a maior variedade possível de modelos de rastreabilidade em cadeias produtivas, ou que ao menos demandasse um baixo esforço de adaptação em virtude de peculiaridades de um determinado tipo de cadeia. Novamente a escolha do modelo de rastreabilidade de Kondo [35] como ponto de partida foi de grande valia, já que sua generalidade foi uma das premissas básicas de seus autores.

É importante destacar ainda as dificuldades encontradas na implementação do protótipo em virtude de limitações da tecnologia adotada e da complexidade da arquitetura, haja visto o grande número de repositórios, sumários e módulos de gerenciamento necessários para seu completo funcionamento.

As principais contribuições desta dissertação são:

- Um estudo detalhado referente ao uso de regras de qualidade associado à rastreabilidade;
- Proposta de um modelo capaz de gerenciar a elaboração, aplicação e análise dessas regras de qualidade, com auditoria dos eventos registrados;
- A especificação e desenvolvimento de algoritmos que transformem tais regras em gatilhos executados pelo SGBD;
- Implementação e validação da arquitetura proposta por meio de simulações.

5.2 Extensões

Esta dissertação oferece diversas possibilidades de extensão, algumas das quais listadas a seguir.

5.2.1 Complexidade da especificação das regras

Neste trabalho propomos uma abordagem inicial de mapeamento das regras de qualidade do modelo relacional para uma linguagem procedural baseada em gatilhos e procedimentos que seja processável pelo mecanismo ativo do SGBD. No entanto, a especificação das restrições poderia ser bastante simplificada, ou ainda o poder de expressividade das restrições ser incrementado, por meio do uso de uma linguagem mais elaboradas e de fácil assimilação ao usuário, como, por exemplo, o OCL *Object Constraint Language* com mapeamento para linguagem procedural.

Além disso, consideramos apenas regras com predicados analisados de forma aditiva (AND). Uma extensão seria considerar predicados conjuntivos ou com negação.

5.2.2 Uso de ontologias

Uma vez que as regras de qualidade se baseiam completamente na comparação entre atributos para a identificação de violações, é necessário garantir a compatibilidade entre as informações analisadas. O uso de ontologias permite identificar corretamente a semântica das informações de rastreabilidade e das restrições, assegurando a compatibilidade dos fatores analisados.

5.2.3 Integração entre regras e workflows

Seria interessante estender o modelo para que fossem considerados não apenas fatores mensuráveis para a avaliação das restrições, mas também características dos métodos de produção adotados na cadeia. Combinar a controle de regras aos workflows de produção possibilitariam o controle sobre práticas prejudiciais à qualidade do produto final. Analisar, por exemplo, se o processo de beneficiamento de café de determinada indústria adiciona conservantes aos grãos é de suma importância para uma segunda indústria que deseje adquirir aquele insumo para produção de café “orgânico”.

5.2.4 Outros tipos de regras de qualidade

Nosso trabalho foca a análise de restrições de integridade baseando-se em características mensuráveis via sensores. Uma extensão relevante seria modificar a especificação das regras no intuito de atender a situações que influenciem na qualidade dos produtos, e onde sensores não possam ser utilizados. Um exemplo seria controlar quantas vezes o produto trocou de veículo durante um determinado transporte, ou se foi respeitada a altura máxima das pilhas de produtos armazenados em um determinado local.

Referências Bibliográficas

- [1] A. AILAMAKI, C. FALOUTOS, P. S. FISCHBECK, M. J. SMALL, AND J. VAN-BRIESEN. An environmental sensor network to determine drinking water quality and security. *SIGMOD Rec.* **32**(4), 47–52 (2003).
- [2] E. BACARIN, C. B. MEDEIROS, AND E. MADEIRA. A collaborative model for agricultural supply chains. In “Proc. OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2004 - LNCS 3290”, pp. 319–336. Springer Berlin / Heidelberg (October 2004).
- [3] L. L. BELLO, O. MIRABELLA, AND N. TORRISI. Modelling and evaluating traceability systems in food manufacturing chains. In “WETICE '04: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04)”, pp. 173–179, Washington, DC, USA (2004). IEEE Computer Society.
- [4] G. BOFF AND J. L. DE OLIVEIRA. Modelagem de restrições de integridade utilizando ocl com mapeamento para sql. *VII Workshop de Teses e Dissertações em Bancos de Dados, XXIII Simpósio Brasileiro de Banco de Dados* (Outubro 2008).
- [5] D. BOX, D. EHNEBUSKE, G. KAKIVAYA, A. LAYMAN, N. MENDELSON, H. F. NIELSEN, S. THATTE, AND D. WINER. Simple object access protocol (soap) 1.1. Technical Report, World Wide Web Consortium (2000).
- [6] A. BUCHMANN, C. BORNHÖVD, M. CILIA, L. FIEGE, F. GÄRTNER, C. LIEBIG, M. MEIXNER, AND G. MÜHL. “DREAM: Distributed Reliable Event-based Application Management”, pp. 319–350. Springer (May 2004).
- [7] D. CHAPPELL AND T. JEWELL. “Java Web Services”. O'Reilly, german ed. (2002).
- [8] R. CHINNICI, M. HADLEY, AND R. MORDANI. The java api for xml-based web services (jax-ws) 2.0. Technical Report, Java Community Process (2006).

- [9] R. CHINNICI, A. RYMAN, S. WEERAWARANA, AND J.-J. MOREAU. Web services description language (WSDL) version 2.0 part 1: Core language. W3C recommendation, W3C (June 2007). <http://www.w3.org/TR/2007/REC-wsdl20-20070626>.
- [10] M. CILIA, M. ANTOLLINI, C. BORNHÖVD, AND A. BUCHMANN. Dealing with heterogeneous data in pub/sub systems: The Concept-Based approach. In “International Workshop on Distributed Event-Based Systems (DEBS’04)”, Edinburgh, Scotland (May 2004).
- [11] M. A. CILIA. Active database system support for topological constraints in geographical information systems. Master’s thesis, Instituto de Computação - Unicamp (March 1996).
- [12] M. G. C. A. CIMINO, B. LAZZERINI, F. MARCELLONI, AND A. TOMASI. Cerere: an information system supporting traceability in the food supply chain. In “CECW ’05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology Workshops”, pp. 90–98. IEEE Computer Society (2005).
- [13] L. CLEMENT, A. HATELY, C. VON RIEGEN, AND T. ROGERS. Uddi spec technical committee draft 3.0.2. Oasis committee draft, OASIS (2004).
- [14] D. CULLER, D. ESTRIN, AND M. SRIVASTAVA. Overview of sensor networks. *IEEE Computer* **37**(8), 41–49 (2004).
- [15] A. M. G. DE CASTRO ET AL. Cadeias produtivas e sistemas naturais - prospecção tecnológica. In “*”, p. 564, Brasília, Brazil (1998). EMBRAPA/Serviço de Produção de Informação.
- [16] C. DE VIGILÂNCIA SANITÁRIA DE SÃO PAULO. Portaria cvs 15 de 7.11.1991 (1991). Disponível em: <http://www.cvs.saude.sp.gov.br/download.asp?tipo=zip&arquivo=91pcvs15.zip>.
- [17] H. M. DEITEL, P. J. DEITEL, R. NIETO, AND E. AL. “XML: Como Programar”. Bookman (2003).
- [18] J. V. F. S. E LUIS MAURÍCIO MARTINS DE RESENDE E LUIZ ALBERTO PILATTI. Rastreabilidade: Uma exigência da cadeia agroindustrial para produtos especiais. *Revista Produção Online* (2006).
- [19] ECLIPSE. Eclipse ide for java ee developers. URL: <http://www.eclipse.org/downloads/moreinfo/jee.php> (2008). (Acessado em: 01/04/2008).

- [20] M. A. FIGUEIREDO AND C. B. MEDEIROS. Gerenciamento de qualidade em cadeias produtivas agrícolas. *VII Workshop de Teses e Dissertações em Bancos de Dados, XXIII Simpósio Brasileiro de Banco de Dados* (Outubro 2008).
- [21] I. GÊNESIS. Sistema gênese de certificação sisbov - sgcs. Technical Report, Instituto Gênese (2007).
- [22] M. GOLFARELLI, S. RIZZI, AND I. CELLA. Beyond data warehousing: what's next in business intelligence? In "DOLAP '04: Proceedings of the 7th ACM international workshop on Data warehousing and OLAP", pp. 1–6. ACM (2004).
- [23] O. M. GROUP. Corba notification service specification. Technical Report, Object Management Group (2004).
- [24] O. O. M. GROUP. Object constraint language (ocl). URL: <http://www.omg.org/spec/OCL/2.0/> (2009). (Acessado em: 03/01/2009).
- [25] F. M. GRZYNA. Manufacturing planning. In "In Juran, J. M. and Gryna, F. M. - Quality Control Handbook", pp. 38–40, Washington, DC, USA (1988). McGraw-Hill.
- [26] M. HAPNER, R. BURRIDGE, R. SHARMA, J. FIALLI, AND K. STOUT. Java message service specification version 1.1. Technical Report, Sun Microsystems, Inc (2002).
- [27] J. M. HELLERSTEIN, W. HONG, AND S. R. MADDEN. The sensor spectrum: technology, trends, and requirements. *SIGMOD Rec.* **32**(4), 22–27 (December 2003).
- [28] M. HENDRICKS, S. CABLE, J. BASHA, B. GALBRAITH, R. IRANI, J. MILBURY, T. MODI, A. TOST, A. TOUSSAINT, AND R. KRAAI. "Professional Java Web Services". Alta Books (2002).
- [29] HIBERNATE. Hibernate - relational persistence for java and .net. URL: <http://www.hibernate.org/> (2008). (Acessado em: 01/04/2008).
- [30] J. E. HOBBS. A transaction cost analysis of quality, traceability and animal welfare issues in uk beef retailing. *British Food Journal* **98**, 16–26(11) (1 October 1996).
- [31] E. INC. Discoverrfid. URL: <http://www.discoverrfid.org/> (2009). (Acessado em: 25/01/2009).
- [32] E. INC. Epcglobal standards overview. URL: <http://www.epcglobalinc.org/> (2009). (Acessado em: 25/01/2009).

- [33] M. H. JANSEN-VULLERS, C. A. VAN DORP, AND A. J. M. BEULENS. Managing traceability information in manufacture. In “International Journal of Information Management” (2003).
- [34] JBOSS.ORG. Jboss application server. URL: <http://www.jboss.org/jbossas/> (2008). (Acessado em: 01/04/2008).
- [35] A. A. KONDO. Gerenciamento de rastreabilidade em cadeias produtivas agropecuárias. Master’s thesis, Instituto de Computação - Unicamp (Abril 2007).
- [36] R. T. M. MACHADO. Signals of food quality and traceability. In “In IV Congresso Internacional de Economia e Gestão de Redes Agroalimentares” (2003).
- [37] K. MARTINEZ, J. K. HART, AND R. ONG. Environmental sensor networks. *IEEE Computer* **37**(8), 50–56 (2004).
- [38] C. B. MEDEIROS AND M. CILIA. Maintenance of binary topological constraints through active databases. In “Proceedings of the 3rd ACM Workshop on Advances in GIS”, pp. 127–134, Baltimore, USA (December 1995). ACM.
- [39] S. MICROSYSTEMS. Java se development kit 6. URL: <http://java.sun.com/javase/6/> (2008). (Acessado em: 01/04/2008).
- [40] T. MOE. Perspectives on traceability in food manufacture. *Trends in Food Science and Technology* **9**, 211–214(4) (May 1998).
- [41] K. MOHAN AND B. RAMESH. Managing variability with traceability in product and service families. In “HICSS ’02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS’02)-Volume 3”, p. 76, Washington, DC, USA (2002). IEEE Computer Society.
- [42] R. NAGAPPAN, R. SKOCZYLAS, R. P. SRIGANESH, AND R. P. SRINGANESH. “Developing Java Web Services”. John Wiley & Sons, Inc., New York, NY, USA (2002).
- [43] A. OLSSON AND C. SKJÖLDEBRAND. Risk management and quality assurance through the food supply chain case studies in the swedish food industry. *The Open Food Science Journal* pp. 49–56 (Agosto 2008).
- [44] G. Z. PASTORELLO, C. B. MEDEIROS, AND A. SANTANCHÈ. Providing homogeneous access for sensor data management. Technical Report IC-07-012, Institute of Computing, State University of Campinas (May 2007).

- [45] D. B. PINTO, I. CASTRO, AND A. A. VICENTE. The use of tics as a managing tool for traceability in the food industry. *Food Research International* p. 772781 (Agosto 2006).
- [46] POSTGRESQL. Postgresql. URL: <http://www.postgresql.org/> (2008). (Acessado em: 01/04/2008).
- [47] P. R. F. SAMPAIO. Dynamic constraints in active object-oriented databases. Master's thesis, Instituto de Computação - Unicamp (December 1994).
- [48] E. E. SANTO AND J. X. MEDEIROS. Coordenação de qualidade na cadeia da carne bovina: O caso da exigência da rastreabilidade. In “In III Congresso Internacional de Economia e Gestão de Redes Agroalimentares” (2001).
- [49] W. SHIROU, D. YUSUKE, O. SATOSHI, AND I. ATSUSHI. Extendable product traceability system from small start. In “SAINT-W '06: Proceedings of the International Symposium on Applications on Internet Workshops”, pp. 76–79, Washington, DC, USA (2006). IEEE Computer Society.
- [50] J. R. STOCK. “Food Supply Chain Management”, ch. 14. The US Food Supply Chain. Blackwell Publishing (2004).
- [51] R. SZEWCZYK, E. OSTERWEIL, J. POLASTRE, M. HAMILTON, A. M. MAINWARING, AND D. ESTRIN. Habitat monitoring with sensor networks. *Commun. ACM* **47**(6), 34–40 (2004).
- [52] Y. TANIGUCHI AND N. SAGAWA. IC Tag Based Traceability: System and Solutions. In “Proc. ICDE '05”, pp. 13–17 (2005).
- [53] TOMCAT. Apache tomcat. URL: <http://tomcat.apache.org/> (2008). (Acessado em: 01/04/2008).
- [54] T. P. WILSON AND W. R. CLARKE. Food safety and traceability in the agricultural supply chain: using the internet to deliver traceability. *Supply Chain Management: An International Journal* **3**(3), 127–133 (1998).