

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**  
**CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA**  
**PROGRAMA DE PÓS-GRADUAÇÃO EM**  
**CIÊNCIA DA COMPUTAÇÃO**

**“Processo de Engenharia de Domínio para Aplicações em Cadeias  
de Suprimentos que Utilizam de Identificação por  
Radiofrequência (RFID)”**

**ORIENTADOR:** Sérgio Donizetti Zorzo  
**ALUNO:** Leonardo Barreto Campos

**São Carlos**  
**Setembro/2010**

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**  
**CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA**  
**PROGRAMA DE PÓS-GRADUAÇÃO EM**  
**CIÊNCIA DA COMPUTAÇÃO**

**“Processo de Engenharia de Domínio para Aplicações em Cadeias  
de Suprimentos que Utilizam de Identificação por  
Radiofrequência (RFID)”**

**Leonardo Barreto Campos**

**Orientação:**  
Prof. Dr. Sérgio Donizetti Zorzo

*Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos como requisito parcial exigido à obtenção do grau de **Mestre em Ciência da Computação**.*

**São Carlos**  
**Setembro/2010**

**Ficha catalográfica elaborada pelo DePT da  
Biblioteca Comunitária da UFSCar**

C198pe

Campos, Leonardo Barreto.

Processo de engenharia de domínio para aplicações em cadeias de suprimentos que utilizam de identificação por radiofrequência (RFID) / Leonardo Barreto Campos. -- São Carlos : UFSCar, 2010.

101 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2007.

1. Engenharia de software. 2. Software - desenvolvimento. 3. Reuso. 4. Cadeia de suprimentos. I. Título.

CDD: 005.1 (20<sup>a</sup>)

# Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

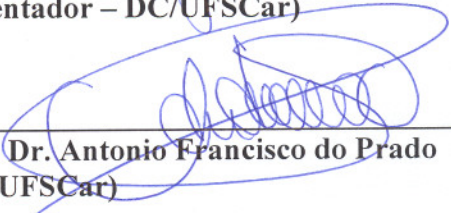
## *“Processo de Engenharia de Domínio para Aplicações em Cadeias de Suprimentos que utilizam de Identificação por RadioFrequência (RFID)”*

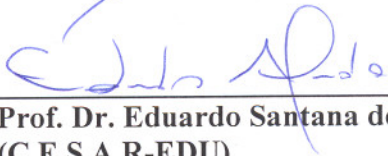
LEONARDO BARRETO CAMPOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

  
\_\_\_\_\_  
Prof. Dr. Sérgio Donizetti Zorzo  
(Orientador – DC/UFSCar)

  
\_\_\_\_\_  
Prof. Dr. Antonio Francisco do Prado  
(DC/UFSCar)

  
\_\_\_\_\_  
Prof. Dr. Eduardo Santana de Almeida  
(C.E.S.A.R-EDU)

São Carlos  
Agosto/2007

## **DEDICATÓRIA**

ao meu Filho,

Luis Eduardo César da Rocha Campos

## AGRADECIMENTOS

Agradeço primeiramente a DEUS por sempre ter me dado sabedoria, vitalidade, humildade e perseverança para finalizar tudo que comecei na vida.

À minha família (Minha Mãe, Meu Pai, Chico e Guila), minha fortaleza. Agradeço pela confiança deposita em mim e por eles terem ajudado a criar meu filho durante todo esse tempo que passei longe de casa, buscando um futuro melhor.

À minha namorada, Thais Silva Pereira, pelo companheirismo, amizade, confiança em meu potencial, palavras de carinho, amor, gratidão, solidariedade e pelo sorriso festivo a cada retorno à Vitória da Conquista.

Ao meu orientador, Sérgio Donizetti Zorzo, que soube exigir no momento certo, soube mudar o objetivo do estudo sem mudar o objeto de estudo e por ter me dado liberdade para guiar nossos trabalhos com bastante sutileza.

Aos membros do grupo Reuse in Software Engineering – RiSE (Alexandre, Vinícius, Lica, Martins, Bart, Fred, K. Brito, Sílvio Meira) especialmente ao Doutor Eduardo Santana de Almeida, por ter me orientado durante uma semana, ter cedido importantes referências bibliográficas e ter mostrado como coordenar um grupo e conseguir extrair o que cada um tem de melhor.

Aos amigos que fiz na República do Maiti (Cássio, Darley, Daniel, Raphael, André, Luiz, Ratto, Guto, Fernando, Marcos, Maycon, Hélio, Ricardo e Reginaldo) que de uma forma ou de outra me ensinaram a viver longe de casa, a respeitar as diferenças, a fazer churrasco e a entender um pouco mais o significado da palavra “amizade”.

Aos amigos que fiz no Departamento de Computação da UFSCar (Luanna, Rigolin, Daniele, Val, Pedro, Robson, Leandro, Cris, Edmilson, Leandro, Ricardo, Ruivo, Somera,

Perazzo, Bertoni, Laia, Patrick, Wilson, Muriel, Tatiane...) pelos diversos ensinamentos de como sobreviver a um mestrado na Federal.

Aos amigos que fiz na Igreja Nossa Senhora do Carmo (Silvinha, Lu, Zan, Tamiris, Érica, a outra Silvinha, aos coroinhas, Tony, Marcelo e todo pessoal do grupo de canto) que sentiam minha ausência e se alegravam com minha presença em cada celebração.

Aos membros da minha banca de Qualificação, o Dr. Antônio Francisco do Prado e o Dr. Carlos Luiz Trevelin, pela contribuição e direcionamento que deram ao meu trabalho em um momento importante dessa trajetória.

Aos Doutores do Departamento (Hélio, Trevelin, Célio, Saito, Rosângela) que solucionaram todas as dúvidas que tive e contribuíram para minha formação pedagógica dentro da sala de aula.

Aos professores do Enoy, CEFET e da UESB, especialmente ao Mestre Marlos André, por ter contribuído diretamente com minha inscrição no Mestrado em Ciência da Computação da UFSCar.

Aos funcionários do Departamento de Computação da UFSCar (Evelton, Mirian, Verinha, D. Ofélia, Cristina, Socorro e as meninas da limpeza) que mantiveram o Departamento operacional e atenderam prontamente todas as minhas solicitações de última hora.

## RESUMO

A Identificação por Radiofrequência abriu uma nova fase no Gerenciamento da Cadeia de Suprimentos que pode ser controlada com mais precisão e agilidade. Através dessa nova tecnologia é possível que os membros de uma Cadeia de Suprimentos troquem informações em tempo real sobre estoque, pedidos de reposição, localização dos produtos e planos de negócio. Por outro lado, todos esses avanços e novos conceitos trazidos pela Identificação por Radiofrequência não foram considerados pelo reuso de software e processos de desenvolvimento baseado em componentes. Nesse sentido, a Engenharia de Domínio apresenta-se como uma solução para coletar, organizar e armazenar experiências passadas na construção de novos sistemas e diminuir custos no desenvolvimento de aplicações baseadas na tecnologia RFID. Dessa forma, este trabalho apresenta um Processo de Engenharia de Domínio baseado em atividades e tarefas bem definidas para o desenvolvimento de sistemas que utilizam a tecnologia RFID na Cadeia de Suprimentos. São apresentadas como resultados três fases de um novo processo de Engenharia de Domínio e a aplicação dessas fases em um estudo de caso da Cadeia de Suprimentos de Supermercados.



## **ABSTRACT**

The Radio Frequency Identification – RFID has opened a new phase in Supply Chain Management that can be controlled with precision and agility. Through this new technology is possible that Supply Chain members exchange informations in real time about stock, replace request, location of the products, and business strategy. However, the Radio Frequency Identification do not considered by software reuse and component-based development processes. In this sense, the Domain Engineering presents one solution to collecting, organizing, and storing past experience in building news systems used to reduce costs in RFID-based software development. Thus, this work presents an Domain Engineering Process based on a well defined set of guidelines and steps to RFID-based software engineering Systems in Supply Chain. We showed as results three Domain Engineering phases and one study case of the Supermarket Supply Chain.

## SUMÁRIO

<b>LISTA DE FIGURAS.....</b>	<b>ix</b>
<b>LISTA DE TABELAS .....</b>	<b>x</b>
<b>LISTA DE ACRÔNIMOS .....</b>	<b>xi</b>
<b>1. Introdução .....</b>	<b>1</b>
<b>2. Sistemas RFID .....</b>	<b>4</b>
2.1 Código Eletrônico do Produto .....	6
2.2 Sistema de Identificação .....	8
2.2.1 Tags RFID .....	8
2.2.2 Leitores RFID .....	10
2.3 Middleware EPC .....	12
2.4 Serviço de Informação EPC .....	12
2.5 Serviço de Descoberta .....	13
2.6 Aplicações dos Sistemas RFID .....	13
2.6.1 Logística .....	14
2.6.2 Controle de Acesso .....	15
2.6.3 Saúde .....	15
2.7 Sistemas RFID e o Reuso de Software .....	16
<b>3. Processos de Reuso de Software.....</b>	<b>18</b>
3.1 Engenharia de Domínio .....	18
3.1.1 Draco .....	19
3.1.2 Feature-Oriented Domain Analysis (FODA) .....	20
3.1.3 Organization Domain Modeling (ODM).....	21
3.1.4 Feature-Oriented Reuse Method (FORM).....	22
3.1.5 RiDE: O Processo RiSE para Engenharia de Domínio .....	23
3.2 Linha de Produtos de Software.....	24
3.2.1 Family-Oriented Abstraction Specification and Translation Process (FAST) .....	25
3.2.2 Engenharia de Linha de Produto FORM .....	26
3.2.3 Komponentbasierte Anwendungsentwicklung (KobrA) .....	27
3.2.4 Pervasive Component Systems (PECOS).....	28
3.2.5 Product Line UML-Based Software Engineering (PLUS).....	28
3.2.6 Component-Oriented Platform Architecting Method (CoPAM).....	29
3.3 Considerações Finais .....	30
<b>4. O Processo de Engenharia de Domínio .....</b>	<b>31</b>
4.1. Visão Geral do Processo.....	31
4.2. Os Fundamentos .....	32
4.3. Passos do Processo de Engenharia de Domínio .....	35
<b>5. Análise do Domínio.....</b>	<b>37</b>
5.1. Planejamento do domínio .....	38
5.1.1 Descrição do domínio.....	38
5.1.2 Identificação dos <i>stakeholders</i> .....	39
5.1.3 Escopo do domínio .....	40
5.2 Requisitos do domínio .....	41
5.3 Modelagem dos requisitos .....	42
5.3.1 Requisitos comuns.....	43
5.3.2 Requisitos variáveis.....	44
5.4 Documentação dos requisitos .....	47

5.5 Considerações Finais .....	49
<b>6. Projeto do Domínio.....</b>	<b>50</b>
6.1 Mapeamento .....	50
6.2 Projeto dos Componentes .....	55
6.2.1 Identificação das Interfaces .....	55
6.2.2 Especificação dos Componentes .....	56
6.3 Visões da Arquitetura .....	56
6.4 Documentação da Arquitetura .....	60
6.5 Considerações Finais .....	62
<b>7. Implementação do Domínio.....</b>	<b>63</b>
7.1 Implementação dos Componentes .....	63
7.2 Documentação dos Componentes.....	64
7.3 Considerações Finais .....	68
<b>8. Estudo de Caso.....</b>	<b>69</b>
8.1 Análise do Domínio.....	69
8.1.1 Planejamento do Domínio .....	69
8.1.2 Requisitos do Domínio .....	72
8.1.3 Modelagem dos Requisitos.....	74
8.1.4 Documentação dos requisitos .....	75
8.2 Projeto do Domínio .....	76
8.2.1 Mapeamento .....	76
8.2.2 Projeto dos Componentes .....	78
8.2.3 Visões da Arquitetura .....	79
8.2.4 Documentação da Arquitetura .....	80
8.3 Implementação do Domínio .....	83
8.3.1 Implementação dos Componentes .....	83
8.3.2 Documentação dos Componentes.....	84
8.4 Considerações Finais .....	85
<b>9. Conclusão .....</b>	<b>86</b>
9.1 Trabalhos Relacionados.....	86
9.2 Trabalhos Futuros .....	87
<b>REFERÊNCIAS BIBLIOGRÁFIAS .....</b>	<b>88</b>

## LISTA DE FIGURAS

Figura 1: Arquitetura da Rede EPCglobal [HARRISON 2004].....	5
Figura 2: Relação entre a WWW e a Rede EPCglobal.....	6
Figura 3: Representação do número EPC dentro da tag RFID.....	7
Figura 4: Tag RFID .....	8
Figura 5: Prateleira inteligente .....	11
Figura 6: Funcionamento dos Sistemas RFID em um domínio .....	16
Figura 7: O Framework RiSE para Reuso de Software [Almeida et al. 2004].....	23
Figura 8: Desenvolvimento de Produto no conceito de linha de produto [NORTHROP 2002] .....	25
Figura 9: Modelo de processo do Processo de Engenharia de Domínio .....	34
Figura 10: Exemplos dos diagramas de <i>features</i> [CZARNECKI; EISENECKER 2000].....	46
Figura 11: Relações em UML para a visão de módulos.....	57
Figura 12: Exemplo da visão <i>Deployment</i> .....	59
Figura 13: Diagrama de Estados – Embarque de mercadorias.....	72
Figura 14: Diagrama de <i>features</i> para a Cadeia de Suprimentos de uma rede de supermercado .....	74
Figura 15: Interfaces do Componente Localização .....	78
Figura 16: Pacote de Especificação das Interfaces.....	79
Figura 17: Visão <i>Deployment</i> da Cadeia de Suprimentos.....	80
Figura 18: Componente Serviço de Localização.....	83

## LISTA DE TABELAS

Tabela 1: Estrutura do número EPC em cada versão .....	7
Tabela 2: Características dos leitores RFID .....	11
Tabela 3: Matriz de requisitos do domínio.....	44
Tabela 4: <i>Template</i> para Documentação de <i>Features</i> .....	48
Tabela 5: <i>Template</i> para Documentação da Arquitetura .....	60
Tabela 6: <i>Template</i> para Documentação de Componente .....	65
Tabela 7: Domínios da Cadeia de Suprimentos de uma rede de supermercados .....	70
Tabela 8: Níveis das <i>features</i> .....	73
Tabela 9: Matriz de requisitos do domínio baseada em prioridade.....	74
Tabela 10: Documentação da <i>feature</i> Produto .....	75
Tabela 11: Documentação da Arquitetura do Estudo de Caso .....	80
Tabela 12: Documentação do Componente Serviço de Localização .....	84

## LISTA DE ACRÔNIMOS

<b>AML</b>	Application Modeling Language
<b>CBD</b>	Component-Based Development
<b>CBSE</b>	Component-Based Software Engineering
<b>CoPAM</b>	Component-Oriented Platform Architecting Method
<b>DA</b>	Domain Analysis
<b>DD</b>	Domain Design
<b>DI</b>	Domain Implementation
<b>DSSA</b>	Domain-Specific Software Architecture
<b>EPC</b>	Eletronic Produt Code
<b>EPC-IS</b>	EPC Information Service
<b>FAST</b>	Family-Oriented Abstraction Specification and Translation Process
<b>FODA</b>	Feature-Oriented Domain Analysis
<b>FORM</b>	Feature-Oriented Reuse Method
<b>FORM</b>	Feature-Oriented Reuse Method
<b>ISSO</b>	International Standards Organization
<b>KobrA</b>	Komponentbasierte Anwendungsentwicklung
<b>MDD</b>	Model-Driven Development
<b>ODM</b>	Organization Domain Modeling
<b>ONS</b>	Object Naming Service
<b>PECOS</b>	Pervasive Component Systems
<b>PLUS</b>	Product Line UML-Based Software Engineering
<b>RFID</b>	Radio Frequency Identification
<b>RiDE</b>	RiSE Process for Domain EGINEERING
<b>SCM</b>	Supply Chain Management
<b>SEI</b>	Software Engineering Institute
<b>TDMA</b>	Time Division Multiple Access

# 1. Introdução

De acordo com a Supply-Chain Council (1997), a cadeia de suprimentos compreende todos os esforços envolvidos na produção e distribuição de um produto final ou de serviços, desde fornecedores até alcançar o cliente. O gerenciamento da cadeia de suprimentos (*Supply Chain Management – SCM*) inclui a gestão da oferta e da procura, o abastecimento de matérias-primas e peças, fabricação e montagem, armazenamento e monitoramento do produto em todos os canais de distribuição e entrega ao cliente.

Nesse contexto de várias fontes de informação exportando dados dinamicamente na cadeia de suprimentos, a Identificação por Radiofrequência (Radio Frequency Identification – RFID) aparece como uma tecnologia capaz de identificar objetos como produtos manufaturados, animais e pessoas. Assim, o objetivo da tecnologia RFID no gerenciamento da cadeia de suprimentos é garantir a interoperabilidade provendo, por exemplo, informações precisas e em tempo real sobre inventário das organizações, recalls de produtos e comunicações entre os participantes da cadeia de suprimentos.

Por outro lado, os sistemas baseados em RFID utilizados no gerenciamento da cadeia de suprimentos não foram considerados por um processo de desenvolvimento de software específico. Nesse cenário, um processo é importante e necessário para definir o modo como uma organização realiza suas atividades, bem como as pessoas trabalham e interagem de forma a atingir seus objetivos. Em particular, processos devem ser definidos para assegurar a eficiência, aplicabilidade e homogeneidade [ALMEIDA 2007].

Existem diversas definições para processos de software [OSTERWIEL 1987], [PRESSMAN 2005] e [SOMMERVILLE 2006]. De acordo com Ezran et al. (2002) processos de software referem-se a todas as tarefas necessárias para produzir e gerenciar software, enquanto que os processos são subconjuntos de tarefas necessárias para desenvolver e reusar

assets. A adoção de um novo e bem definido processo de software gerenciado e customizado é um facilitador para um eventual sucesso no reuso de programas [MORISIO et al. 2002]. No domínio da cadeia de suprimentos, muitos cenários e processos repetem entre os participantes da cadeia de suprimentos (sub-domínio), por exemplo, o gerenciamento do estoque, o recebimento e entrega das mercadorias e a localização de produtos.

Nesse sentido, o reuso de software – processo de criação de sistemas a partir de software existente em vez de construí-lo a partir do zero – é um aspecto fundamental para a melhoria da qualidade e produtividade no desenvolvimento de software. No contexto do reuso de software, uma importante pesquisa, incluindo relatórios de empresas [BAUER 1993], [ENDRES 1993], [GRISS 1994], [JOOS 1994], [GRISS 1995], pesquisa informal [FAKES; ISODA 1995], [FRAKES; KANG 2005] e estudos empíricos [RINE 1997], [MORISIO et al. 2002), [ROTHENBERGER et al. 2003] destacaram a relevância de um processo de reuso, uma vez que a forma mais comum de reuso de software implica desenvolver aplicações reutilizando assets pré-definidos.

O trabalho foca em duas direções para processos de reuso de software: Engenharia de Domínio e Linhas de Produto de Software. Dessa forma, motivado pela constante atualização da tecnologia RFID na cadeia de suprimentos e a falta de um processo específico para o desenvolvimento de sistemas baseados em RFID na cadeia de suprimentos, este trabalho define um processo sistemático de engenharia de domínio no qual inclui os passos de análise do domínio, projeto do domínio e implementação do domínio.

O texto desta dissertação está estruturado em nove capítulos, no Capítulo 2 é apresentado um levantamento bibliográfico dos Sistemas RFID padronizados pela EPCglobal Inc<sup>TM</sup>. A finalidade do capítulo é direcionar a adoção do processo de Engenharia de Domínio baseado em um padrão consolidado e conhecido da Identificação por Radiofrequência. No Capítulo 3 são apresentados os conceitos, os objetivos e as principais fases de onze processos



de reuso de software estudados e que contribuíram para a obtenção do novo processo de reuso apresentado pelo trabalho. O Capítulo 4, detalha uma visão geral do processo de engenharia de domínio e seus fundamentos.

O Capítulo 5 descreve a primeira fase do processo de Engenharia de Domínio apresentando passos, atividades e tarefas para a execução da Análise do Domínio nos domínios da Cadeia de Suprimentos. O Capítulo 6 tem como objetivos: orientar o arquiteto do domínio na escolha de padrões de projeto que servirão no mapeamento dos requisitos variáveis, obter múltiplas visões arquiteturais do Projeto do Domínio, identificar e especificar os componentes.

O Capítulo 7 apresenta a última fase do processo de Engenharia de Domínio descrevendo as atividades de implementação e documentação de componentes obtidas a partir das melhores literaturas da comunidade científica na área de Linhas de Produtos de Software e Engenharia de Domínio. No Capítulo 8, será apresentado a aplicação do processo descrito em um estudo de caso. Por fim, no Capítulo 9, serão apresentadas as principais contribuições do trabalho e as perspectivas para trabalhos futuros.

## 2. Sistemas RFID

Uma questão crítica das novas tecnologias é sua padronização. No caso dos Sistemas RFID, tanto a EPCglobal quanto a International Standards Organization (ISSO) tem adotado a RFID nos seus padrões. Porém, de acordo com [SABAGHI; VALDYANATHAN 2008], o mais promissor padrão industrial da RFID são as especificações e padrões adotados pela EPCglobal. A EPCglobal Inc é uma organização sem fins lucrativos que foi criada em 2003 pelo MIT Auto-ID Center em cooperação com outras universidades de pesquisa para estabelecer a Rede EPCglobal como o padrão global para a identificação automática e correta de qualquer item na cadeia de suprimentos.

A EPCglobal está estabelecendo um padrão de como as informações são passadas dos leitores RFID para várias aplicações, bem como de aplicação para aplicação e a cadeia de suprimentos. Esses padrões são especificados no Framework da Arquitetura EPCglobal, ou simplesmente, Rede EPCglobal. Ela é uma coleção de hardwares inter-relacionados, softwares e padrões de dados, juntamente com serviços essenciais que são operados pela EPCglobal e delegados, todos no serviço de objetivo comum de incrementar os fluxos de negócios e aplicativos de computador mediante a utilização de Códigos Eletrônicos do Produto (Electronic Product Code – EPC) [ARMENIO 2007]. A Rede EPCglobal é composta de cinco componentes: (i) Código Eletrônico do Produto, (ii) Sistema de Identificação, (iii) Middleware EPC, (iv) Serviços de Descoberta e (v) Serviço de Informação EPC. A Figura 1 apresenta os módulos, softwares e interfaces definidas para a Rede EPCglobal.

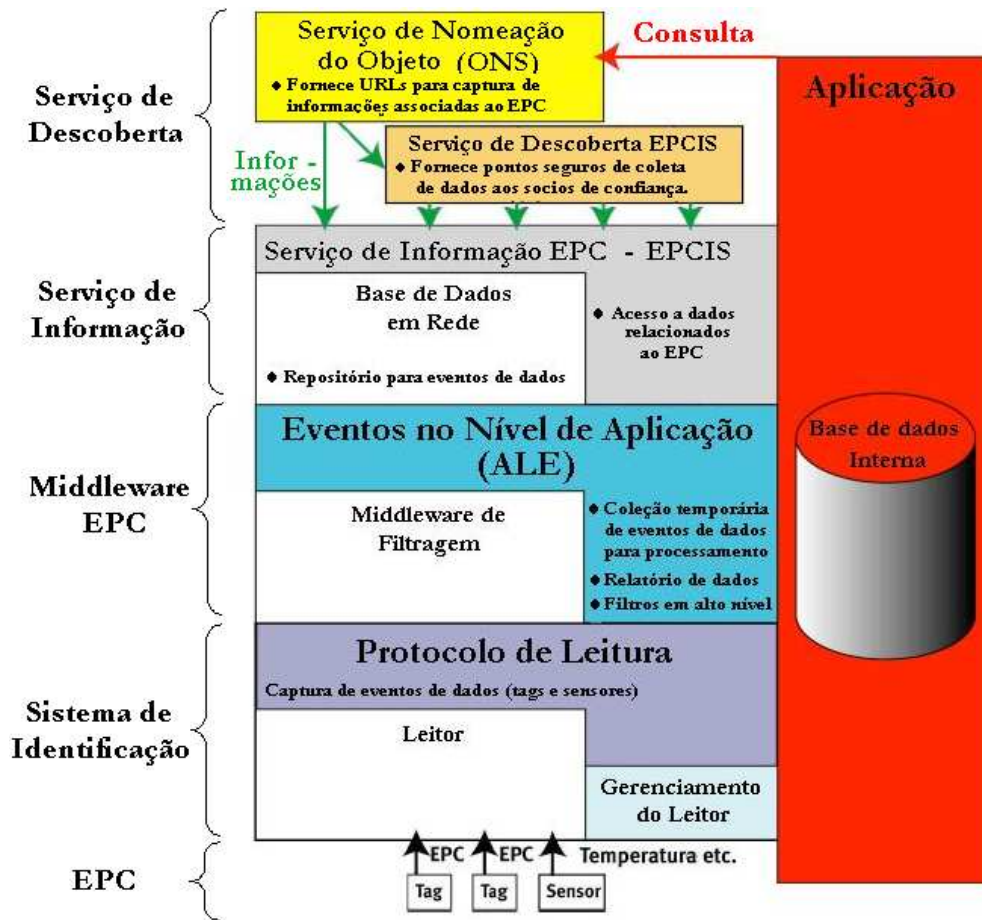
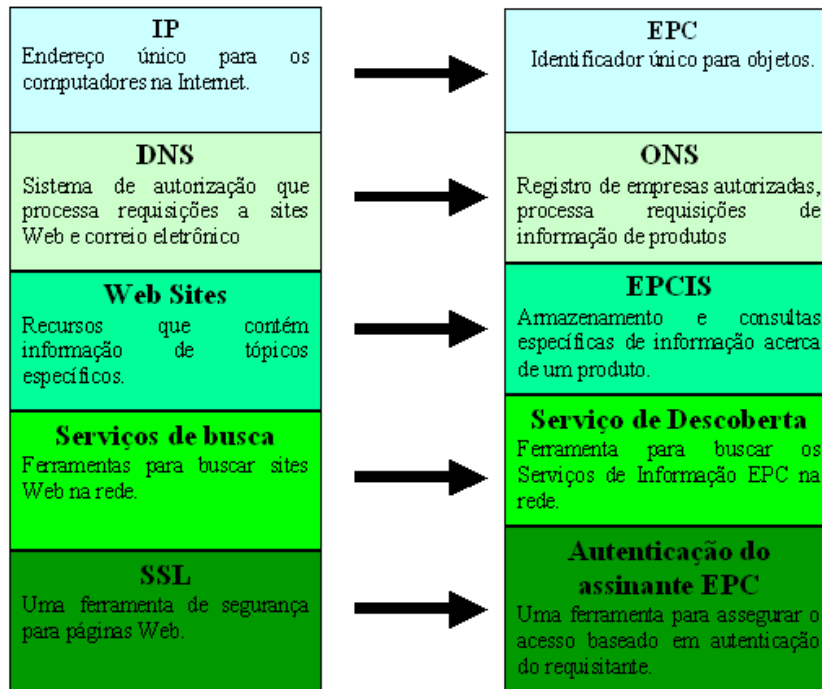


Figura 1: Arquitetura da Rede EPCglobal [HARRISON 2004]

A Arquitetura da Rede EPCglobal abrange as camadas responsáveis por tratar dos objetos físicos, da comunicação com leitores, do gerenciamento e filtragem dos dados, até a troca de informação entre membros da cadeia de suprimentos. Em comparação com a *World Wide Web* – *WWW* a Rede EPCglobal possui algumas relações entre seus módulos como mostra a Figura 2.



**Figura 2:** Relação entre a WWW e a Rede EPCglobal

As próximas seções do Capítulo 2 descreverão detalhadamente os padrões de hardware, software e interface de dados projetadas para a Rede EPCglobal.

## 2.1 Código Eletrônico do Produto

O Código Eletrônico do Produto (*Electronic Product Code – EPC*) é um esquema de identificação e nomeação projetado para identificar unicamente todos os objetos físicos e virtuais, grupos de objetos e não objetos tais como serviços [ENGELS 2003a]. O EPC é um número composto de um cabeçalho e três campos de dados e pode ser armazenado dentro de uma tag RFID, como mostra a Figura 3.



**Figura 3:** Representação do número EPC dentro da tag RFID

O primeiro campo de um número EPC é o cabeçalho, que guarda a versão utilizada por aquele EPC, podendo variar em códigos GS1 SSCC, GS1 GRAI entre outros [EANUCC 2006]. O primeiro campo de dados representa o número de identificação do Gerente do EPC, o segundo campo de dados, também chamado de Classe do Objeto, refere-se exatamente ao tipo do produto, sendo possível diferenciar classes de produtos como alimentos, móveis ou uma peça de vestuário, por exemplo. No terceiro e último campo de dados tem-se o Número de Série do produto que é único para cada produto dentro de sua classe.

O total de números utilizados em cada campo do EPC muda de acordo com a versão do número EPC que pode ser de 64 bits [BROCK 2001c], 96 bits [BROCK 2001b] ou 256 bits [ENGELS 2003b]. A versão EPC de 96 bits tem sido utilizada nas aplicações com os bits dispostos como mostra a Tabela 1 [ENGELS, 2003a].

**Tabela 1:** Estrutura do número EPC em cada versão

EPC		Número da Versão	Gerente do Domínio	Classe do Objeto	Número de Série
64 bits	Tipo I	2	21	17	24
	Tipo II	2	15	13	34
	Tipo III	2	26	13	23
96 bits	Tipo I	8	28	24	36
256 bits	Tipo I	8	32	56	192
	Tipo II	8	64	56	128
	Tipo III	8	128	56	64

Através do número Código Eletrônico do Produto o Serviço de Informação EPC (*EPC Information Service – EPC-IS*) conseguirá obter informações relevantes sobre o produto em um banco de dados local ou na Rede EPCglobal.

## 2.2 Sistema de Identificação

O sistema de identificação compreende as tags e os leitores RFID, através desses dois hardwares é possível armazenar números EPCs e recuperar essa informação a qualquer momento. Esses mecanismos são detalhadamente a seguir.

### 2.2.1 Tags RFID

Tags eletrônicas consistem de um pequeno *microchip* capaz de guardar um identificador único, uma antena que comunica com o leitor e portada opcionalmente de uma bateria como mostra a Figura 4. As tags permitem guardar dados de 96 bits, tais como um número de série ou um código do produto a ser transmitido aos leitores a partir da necessidade da aplicação.



**Figura 4:** Tag RFID

As tags diferem quanto ao seu sistema de alimentação e ao tipo de memória. Em relação ao sistema de alimentação, tags RFID que possuem bateria são chamadas tags ativas

enquanto que tags sem alimentação são denominadas tags passivas. Uma variação das tags passivas são as tags semi-passivas que usam a bateria interna somente para tornar ativo o circuito elétrico do chip, mas comunica-se pela potência extraída do leitor.

As etiquetas RFID também podem ser classificadas quanto à memória interna de cada tag. Essa classificação varia quanto ao modo de gravação de dados que pode ser do tipo leitura e escrita (*read-write*) ou somente leitura (*read-only*). As tags de leitura e escrita permitem adicionar ou remover informações existentes na memória da tag. Esse tipo de tag é útil em aplicações específicas, pois apresenta um custo elevado para ser incorporado em sistemas de rastreamento comum.

O Auto-ID Center classifica as tags RFID quanto a sua funcionalidade, as classes de tags definidas em [KABACHINSKI 2005] e [SARMA; ENGELS 2003] são:

- **Classe 0:** Essa classe representa o tipo mais simples de tag pois, corresponde às tags passivas do tipo somente leitura (*read-only*) que armazenam uma única vez os dados gravados pelo fabricante. Ao gravar um dado sobre a tag o fabricante impossibilita que a memória receba atualizações.
- **Classe 1:** Compreende as tags passivas somente leitura, que possibilitam a gravação de dados uma única vez. Sua diferença para a Classe 0 é que a gravação pode ser realizada tanto pelo fabricante da tag quanto pelo usuário final mas, a idéia principal de que os dados sejam escritos na tag uma única vez continua válida.
- **Classe 2:** São as tags passivas de leitura/gravação. Os usuários têm livre acesso para ler e escrever dados na memória da tag. Tags dessa classe normalmente tem memória expandida e podem ser usadas para guardar dados históricos, por exemplo, em uma linha de montagem a tag pode ser atualizada em cada etapa da produção.

- **Classe 3:** Considerada uma classe de tag semi-ativa, a classe 3 representa as tags de leitura e gravação com fonte própria de energia realimentada por leitores e podem conter ainda sensores. As tags dessa classe possuem mais circuitos eletrônicos a fim de guardar dados como temperatura, pressão, ou movimento captado pelos sensores associados a elas.
- **Classe 4:** Essa é a classe que compreende as tags com as maiores funcionalidades pois, abrange vantagens de outras classes e possui um transmissor integrado que possibilita a comunicação com outras tags sem auxílio de leitores.

### 2.2.2 Leitores RFID

Leitores RFID são hardwares capazes de captar o Código Eletrônico do Produto armazenado dentro de uma tag RFID, através de um protocolo de comunicação específico [HODGES; HARRISON 2003] e fornecê-lo ao Middleware EPC. Os leitores RFID podem ainda gravar dados em tags aptas para escrita e prover transmissão criptografada entre a tag e a interface de radiofrequência [KNOSPE et al. 2005].

O sinal emitido pelo leitor RFID é propagado para todas as direções e pode servir como fonte de energia para tags. A precaução que se deve ter com esse hardware é com possíveis interferências de sinais causadas pela emissão de ondas de outro leitor RFID. Para tanto, o esquema de múltiplos acessos por divisão de tempo (*Time Division Multiple Access – TDMA*) é recomendado para leitores que possuam a mesma área de cobertura.

Outro problema que o leitor RFID tem que resolver ocorre quando duas ou mais tags respondem ao sinal emitido por ele. O Auto-ID Labs adotou um método padrão onde o leitor pede que respondam somente às tags que possuam seus primeiros dígitos iguais aos dígitos comunicados pelo leitor, por exemplo, o leitor diz às tags: “Responda somente se seu EPC



começa com 0” se mais de uma etiqueta responder o leitor volta a perguntar: “Responda se seu EPC começar com 00” até que somente uma tag responda. Apesar de parecer um procedimento lento, um leitor RFID pode ler até 50 tags em menos de um segundo [AUTO-ID CENTER 2001] como mostra a Tabela 2[LOGICALCMG 2004].

**Tabela 2:** Características dos leitores RFID

Frequência	Taxa de Leitura (metros)	Velocidade (tags/seg)
125 – 134 KHz	0.45	1 – 10
13.56 MHz	< 1	10 – 40
868 – 870, 902 – 928 MHz	2 – 5	10 – 50

A Tabela 2 mostra também os diferentes alcances dos leitores para coletar dados de uma tag sem prejuízo de informação, esse alcance de leitura depende da frequência do leitor usado na comunicação que aumenta à medida que a frequência do leitor aumenta [LARAN RFID 2004]. A depender da utilização dos leitores, eles podem ser incorporados em dispositivos de mão, como *handholds* ou podem ser incorporados em locais fixos, como as “prateleiras inteligentes” (*smart shelves*) visualizadas na Figura 5.



**Figura 5:** Prateleira inteligente

Essas prateleiras detectam quando seus produtos são adicionados ou removidos, atualizando automaticamente o estoque de cada produto.

### 2.3 Middleware EPC

O Middleware EPC é o responsável por controlar em tempo real a leitura de informações e eventos, fornecer alertas e controlar a informação vinda do leitor RFID para comunicação com o Serviço de Informação EPC. Conforme a Figura 1, é no Middleware que o número EPC é tratado e sua consistência é verificada antes de ser encaminhado ao Serviço de Informação EPC que irá para resolvê-lo localmente ou na Rede EPCglobal.

Como em todos os sistemas de camadas, a rede EPC é subdividida visando a entrega perfeita dos dados das camadas inferiores às camadas superiores, no caso do Middleware EPC para que o número EPC seja entregue corretamente ao EPC-IS, filtros e mecanismos de coleta de dados são configurados sobre esse Middleware. Os filtros agem para coordenar as atividades de um ou mais leitores RFID que ocupam o mesmo espaço de cobertura e com possibilidade de ocorrer interferência de ondas.

### 2.4 Serviço de Informação EPC

O Código Eletrônico do Produto é um identificador único que serve como chave de procura na base de dados que contém informações do objeto etiquetado. O Serviço de Informação EPC, mostrado na Figura 1, fornece uma interface padrão para acesso e armazenamento persistente de dados relacionados aos números EPCs que podem ser lidos ou modificados por partes autorizadas [BROCK; CUMMINS 2003].

O Serviço de Informação EPC apresenta três interfaces, são elas: a interface de captura de dados (*EPC Capture Interface*) que fornece um caminho para comunicação de eventos gerados por aplicações de EPC-IS associados, aplicações internas ou da própria base de dados do EPC-IS local.

A interface de consultas (*EPC Query Interface*) que prover meios pelo qual uma aplicação pode solicitar dados EPC-IS de uma base de dados e os meios para que o resultado retorne. Por fim, o Serviço de Informação EPC define um módulo de especificação de dados dentro do EPC-IS (*EPCIS Data Specification*) que descreve o produto através de um Schema XML.

## 2.5 Serviço de Descoberta

O Serviço de Descoberta permite aos usuários da Rede EPCglobal encontrar dados relacionados a um número EPC específico e solicitar acesso a este dado. Como mostra a Figura 1, o Serviço de Descoberta retorna diversos pontos que contenha dados sobre um número EPC além da entidade originalmente possuidora desse número. Por esse motivo, o serviço de descoberta classifica as bases de dados que disponibilizaram dados relacionados ao número EPC como confiáveis e não confiáveis.

O principal componente do Serviço de Descoberta é o Serviço de Nomeação de Objetos (*Object Naming Service – ONS*), responsável pela tradução de um número EPC em um endereço de rede do serviço que contém dados atuais pertencentes ao EPC em questão [EPCGLOBAL 2005]. Quando um número EPC é transmitido ao ONS, a função desse serviço é localizar na Internet o endereço de rede onde está armazenada a informação atual desse EPC e retornar ao serviço solicitante o endereço encontrado.

## 2.6 Aplicações dos Sistemas RFID

Desde a invenção da RFID até o final do século XX existem poucos registros de aplicações baseadas em RFID no contexto comercial [LANDT 2005]. Porém, quando as

primeiras indústrias perceberem o potencial de mercado da RFID e seus produtos diversos, a adoção dessa tecnologia cresce continuamente em seguimentos como, por exemplo, logística, controle de acesso, segurança, estoque de mercadores, etc. Em caso marcante no cenário mundial ocorreu em janeiro de 2004 quando a Wal-Mart passou a exigir da maioria dos seus fornecedores a identificação de todas pallets com tags RFID e até 2007 a adequação de todos os seus fornecedores.

Empresas como IBM, HP, SAP, Microsoft, Oracle, etc investem grande quantidade de dinheiro em pesquisas sobre tecnologias RFID. Segundo [GALLEN 2009] mesmo diante da desaceleração da economia, as receitas totais obtidas por produtos e serviços RFID somou um montante de US\$ 5,6 bilhões em 2009.

Diante desses indicadores, as próximas seções apresentarão as principais aplicações da Identificação por Radiofrequência.

### **2.6.1 Logística**

O Gerenciamento da Cadeia de Suprimentos (Supply Chain Management – SCM) é uma das mais comuns aplicações dos Sistemas RFID. Neste cenário, produtos e mercadorias são etiquetados com tags RFID e todos os itens são controlados por leitores RFID desde sua fabricação até os pontos de venda. Empresa como a Motorola inclui RFID na gestão de sua cadeia de suprimentos [MOTOROLA 2007] bem como, a DHL que iniciou o desenvolvimento de uma infraestrutura global de TI para controlar todos os pacotes com tags RFID até 2015 [BACHELDOR 2008].

### **2.6.2 Controle de Acesso**

Identificação pessoal onde tags RFID são incorporadas a cartões de identificação é outra grande aplicação da RFID [WEINSTEIN 2005]. Além de fornecer um armazenamento mais confiável de informações de identificação comparados com cartões magnéticos os cartões RFID possibilitam diferentes níveis de acesso de acordo com os diferentes níveis de segurança concedido no cartão do proprietário.

A tecnologia RFID também pode ser usada em cartões eletrônicos (e-tickets) para controlar o acesso a exposições, estádios, parques, etc. De acordo com [GUANGJIN 2008] nos Jogos Olímpicos de Pequim ocorridos em 2008 inseriu a tecnologia RFID nos seus bilhetes com a finalidade de associar dados pessoais dos espectadores ao ingresso, como por exemplo, fotografia do titular do bilhete, dados do passaporte, endereço, e-mail e telefone [LEE 2008].

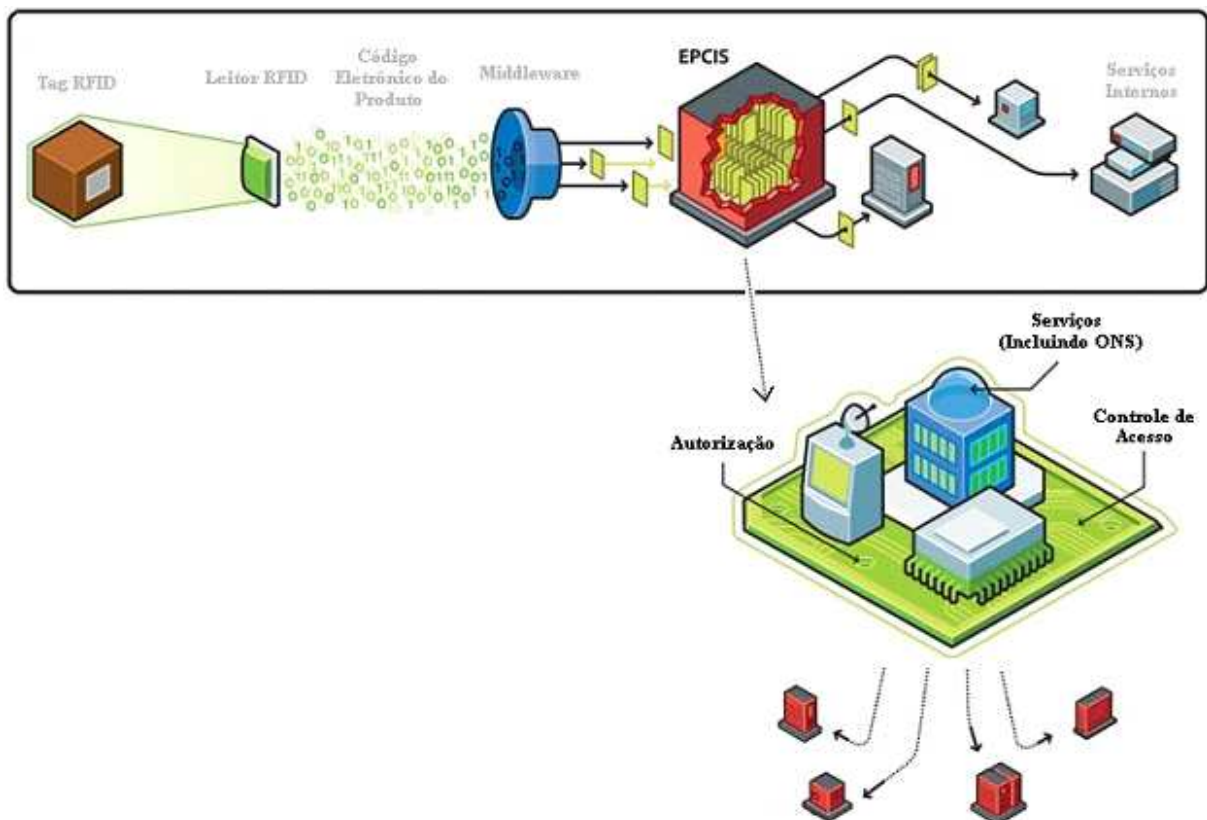
### **2.6.3 Saúde**

As principais aplicações da RFID nas demandas de saúde são para minimizar erros médicos. Nesse sentido a tecnologia RFID ajuda a automatizar a admissão, a triagem e o tratamento de processos [CANGIALOSI; MONALY; YANG 2007]. Outra aplicação importante da RFID é no controle de acesso de funcionários e pacientes, que possibilita, por exemplo, a localização de pessoal e de pacientes em leitos implantados em locais diferentes. Por fim, é comum encontrar hospitais que utilizam tags para identificar recém-nascidos para alertar funcionários em casos de seqüestros [WICKS; VISICH; LI 2006].

## 2.7 Sistemas RFID e o Reuso de Software

Os Sistemas RFID descritos nas seções anteriores contêm aspectos que podem ser considerados pelo reuso de software. De acordo com a Figura 1, os Hardwares e Softwares definidos para o funcionamento da Rede EPCglobal são gerenciados por uma aplicação com uma base de dados integrada.

Nesse sentido, o reuso de software pode ser usado no desenvolvimento das aplicações que irão gerenciar os Sistemas RFID de cada domínio a fim de diminuir custos e aumentar a qualidade das aplicações desenvolvidas. Na Figura 6 é possível visualizar como as partes que compõem os Sistemas RFID interagem.



**Figura 6:** Funcionamento dos Sistemas RFID em um domínio

Ao consultar a localização de um produto na cadeia de suprimentos, por exemplo, a aplicação executará as seguintes atividades: (i) leitura do número EPC armazenado na tag, (ii)

filtragem e tratamento do número EPC pelo Middleware EPC, (iii) busca de informações relacionadas ao produto pelo Serviço de Informação EPC nas bases de dados internas, (iv) autenticação na Rede EPCglobal, (v) busca de informações relacionadas ao produto pelo Serviço de Nomeação de Objetos nas bases de dados externas, (vi) entrega dos resultados ao usuário ou aplicação que efetuou a requisição.

O procedimento citado acima é comum a todos os domínios da Cadeia de Suprimentos. Portanto, o processo de reuso de software que será descrito nos capítulos 4, 5, 6 e 7 tem como principal objetivo facilitar a identificação de características, cenários e arquiteturas comuns para os analistas e projetistas do domínio bem como, auxiliar a implementação de componentes de softwares que associam conceitos dos Sistemas RFID.

No próximo capítulo serão apresentados os objetivos e as atividades de onze processos de reuso de software, distribuídos entre a Engenharia de Domínio e as Linhas de Produto de Software, que contribuirão na descrição do processo para aplicações da Cadeia de Suprimentos baseadas em RFID.

## 3. Processos de Reuso de Software

Desde o momento que o desenvolvimento de software começou a ser discutido na indústria, pesquisadores e profissionais tem procurado métodos, técnicas e ferramentas que permitam a melhoria de custos, tempo de mercado e de qualidade [ALMEIDA 2007]. Os processos de reuso de software compreendem conceitos, estratégias, técnicas e princípios que permitem aos desenvolvedores criar novos sistemas de forma eficaz usando arquiteturas e componentes desenvolvidos anteriormente.

Um processo de reuso de software, além de apresentar as questões relacionadas com aspectos não-técnicos (educação, cultura, aspectos organizacionais, etc), devem descrever duas atividades essenciais: o desenvolvimento para reutilização (Engenharia de Aplicação), que consiste na criação de aplicativos baseados em artigos produzidos em Engenharia de Domínio.

No estado da arte do reuso de software, processos apresentados em [ALMEIDA et al. 2005] e as discussões sobre o assunto em [FRAKES; KANG 2005] é possível notar que vários e importantes estudos de pesquisa foram desenvolvidos com vista a encontrar formas eficientes de desenvolver softwares reusáveis. Estes trabalhos focam em duas direções: Engenharia de Domínio e, atualmente Linhas de Produtos de Software, como pode ser visto nas próximas seções:

### 3.1 Engenharia de Domínio

A Engenharia de Domínio é definida em [CZARNECKI; EISENECKER 2000] como “uma atividade de coleção, organização e armazenamento de experiências passadas no desenvolvimento de sistemas ou partes dos sistemas em um domínio particular na forma de



artefatos reusáveis, bem como prover um meio adequado para reusar esses artefatos no desenvolvimento de novos sistemas”.

Nesse sentido, foram desenvolvidos diversos processos de Engenharia de Domínio como [NEIGHBORS 1980], [STARS 1993], [SIMOS et al. 1996], [JACOBSON; GRISS; JONSSON 1997] e [KANG et al. 1998] com o objetivo principal de viabilizar o desenvolvimento do software reusável.

### 3.1.1 Draco

A abordagem Draco é resultado do trabalho de Pós-Doutorado de James Neighbors (1980) e atualmente é considerada a primeira abordagem de Engenharia de Domínio. A idéia principal de Draco é organizar a construção do software em um número de domínios relacionados [NEIGHBORS 1980]. Cada domínio Draco encapsula as necessidades, requisitos e implementações diferentes de uma coleção de sistemas similares.

Especificamente, um domínio contém os seguintes elementos: (i) Linguagem Formal do Domínio: linguagem usada para descrever o sistema a ser implementado, (ii) Conjunto de Transformações de Otimização: representa regras de troca de fragmentos de programas equivalente na linguagem do domínio, (iii) Conjunto de Componentes transformacionais: cada componente consiste de um ou mais refinamentos capaz de traduzir os objetos e operações de uma linguagem do domínio fonte em uma ou mais domínios alvo, (iv) Procedimentos Específicos do Domínio: são usados sempre que um conjunto de transformações pode ser matematicamente executado, (v) Estratégias e Táticas de Transformação: regras que ajudam a determinar quando aplicar o refinamento de tática (independente do domínio) ou estratégia (dependente do domínio).

Draco apresentou uma primeira direção para desenvolver software usando Engenharia de Domínio, entretanto, sua abordagem é muito difícil de ser aplicada no ambiente industrial devido sua complexidade para executar atividades, bem como escrever transformações e a própria máquina Draco.

### 3.1.2 Feature-Oriented Domain Analysis (FODA)

Descrito por [KANG et al. 1990], o método de Análise do domínio Orientado à *Feature* (*Feature-Oriented Domain Analysis – FODA*) foi desenvolvido pelo Instituto de Engenharia de Software (*Software Engineering Institute – SEI*) e trouxe uma grande contribuição para os desenvolvedores de softwares reusáveis que foi o modelo de *features*. Para obter o modelo de *features* utilizando o método FODA, o analista do domínio precisa executar duas tarefas, descritas a seguir.

A primeira tarefa é a Análise do Contexto, sendo que seu objetivo é definir o escopo do domínio que possa fornecer produtos úteis ao domínio. Nesta fase os relacionamentos entre o foco do domínio e de outros domínios ou entidades também são estabelecidos e analisados para variações. Os resultados da análise do contexto, juntamente com fatores como disponibilidade de especialista do domínio e dificuldades no projeto são usados para limitar o escopo do domínio. Os resultados da Análise do Contexto possibilitam descrever o modelo do contexto que inclui um diagrama de estrutura e um diagrama de contexto.

A Modelagem do Domínio é a segunda tarefa, nela as principais associações e variações entre as aplicações e o domínio são identificadas e modeladas. Esta fase envolve os seguintes passos: (i) Análise da Informação: adquirir conhecimento do domínio em forma de entidades de domínio e o relacionamento entre eles, (ii) Análise de Características: capturar conhecimentos de clientes ou usuários finais das potencialidades gerais das aplicações em um

domínio e (iii) Análise Operacional: produzir o modelo operacional que representa como a aplicação trabalha, capturando os relacionamentos entre os objetos no modelo de informação e as características no modelo de *features*.

Em FODA, *feature* é definida como “um importante aspecto visível ao usuário, uma qualidade, ou uma característica do sistema de software ou dos sistemas” e são documentadas, sobretudo, através do diagrama de *features*. A fim de uma representação mais completa, o método FODA descreve três tipos de características: (i) *feature imperativa* em que cada aplicação no domínio deve ter, (ii) *feature alternativa* onde as aplicações podem possuir apenas uma de cada vez e (iii) *feature opcional* em que uma aplicação pode ou não ter.

### 3.1.3 Organization Domain Modeling (ODM)

O método de engenharia de domínio *Organization Domain Modeling – ODM* desenvolvido por Mark Simos consiste em três fases principais: (i) Planejamento do Domínio: fase destinada à delimitação do escopo do domínio, (ii) Modelagem do Domínio: fase onde o modelo do domínio é desenvolvido e (iii) Base de *Assets* do Engenheiro: fase que objetiva a produção da arquitetura dos sistemas do domínio e a implementação de *assets* reusáveis [SIMOS et al. 1996].

Ao descrever as três fases do método ODM, algumas características inovadoras para os processos de engenharia de domínio existentes são apresentadas, como o foco nos *stakeholders*, tipos de domínio, notação mais geral para as *features*, análise das combinações de *features*, modelagem conceitual, flexibilidade da arquitetura, etc. Por outro lado, as fases não apresentam detalhes específicos de como executar muitas atividades, pois o método ODM fornece “uma direção geral de alto nível de como aplicar o método dentro de um projeto ou organização particular” [SIMOS et al. 1996].

### 3.1.4 Feature-Oriented Reuse Method (FORM)

O *Feature-Oriented Reuse Method – FORM* foi desenvolvido pela Universidade Pohang de Ciência e Tecnologia e constitui uma extensão do método *Feature-Oriented Domain Analysis – FODA*. FORM é um método sistemático com foco na captura dos pontos comuns e nas diferenças de aplicações em um domínio baseado em *features* [KANG et al. 1998]. O resultado da análise é utilizado para desenvolver arquiteturas dos domínios e componentes. Em FORM, o uso de *features* é motivado pelo fato de que consumidores e engenheiros muitas vezes falam de características do produto em termo de *features* que o produto tem ou fornece.

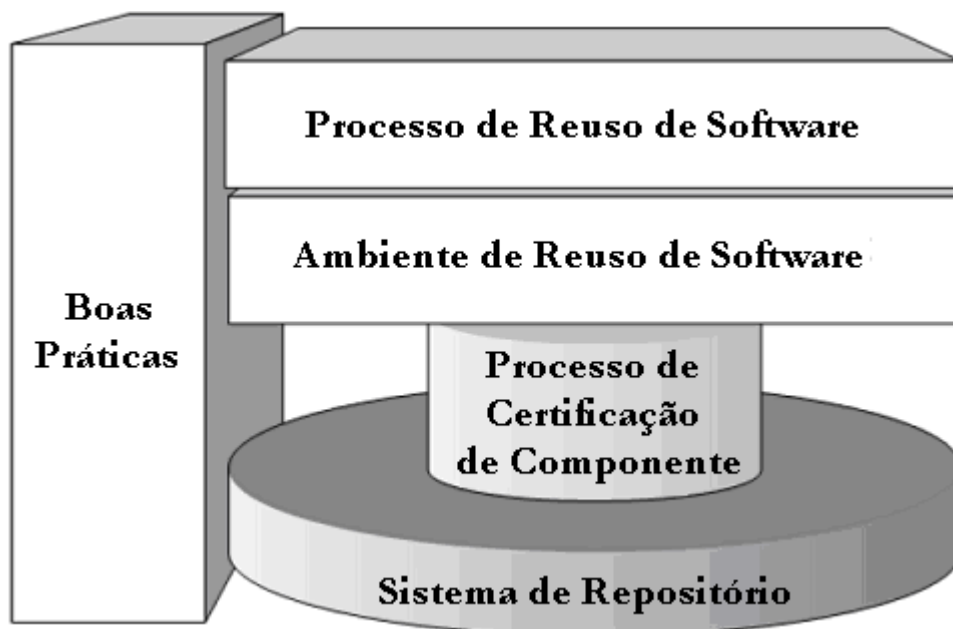
O método FORM consiste de dois processos principais: a Engenharia de Domínio e a Engenharia de Aplicação. A Engenharia do Domínio consiste em analisar sistemas em um domínio, criar arquiteturas referências e reusar componentes baseados nos resultados dessa análise. A arquitetura referência e componentes reusáveis são esperados para acomodar os pontos em comum e as variações dos sistemas do domínio. No processo de Engenharia de Domínio apresentado por FORM, são definidos três passos: análise do contexto, modelagem do domínio e modelagem da arquitetura.

A Engenharia de Aplicação é um processo de desenvolvimento de uma aplicação específica que faz uso do conhecimento do domínio obtido durante a engenharia de domínio, por exemplo, a arquitetura referência encontrada e os componentes de software reusáveis. Esse processo ocorre primeiramente analisando exigências do usuário e selecionando *features* apropriadas e válidas do domínio para o modelo de *feature*. Em seguida, é identificado o modelo referência correspondente e, por fim, é desenvolvida a aplicação por meio de componentes reusáveis de uma forma *bottom-up*.

### 3.1.5 RiDE: O Processo RiSE para Engenharia de Domínio

Baseado em um *survey* do estado da arte dos processos de reuso de software, o Processo RiSE para Engenharia de Domínio (*RiDE: The RiSE Process for Domain Engineering*) descreve passos bem definidos para a análise do domínio, projeto do domínio e implementação do domínio [ALMEIDA 2007]. Inicialmente o método RiDE mostra pontos forte, fracos e brechas deixadas pelos processos de reuso de software existentes. Em seguida, apresenta o processo de Engenharia de Domínio através de atividades, sub-atividades, entradas, saídas, princípios, *guidelines* e papéis.

Outra importante contribuição de RiDE é a apresentação do projeto *Reuse in Software Engineering – RiSE* descrito em [ALMEIDA et al. 2004]. O objetivo do projeto RiSE é “desenvolver um framework robusto para reuso de software a fim de permitir a adoção de um programa de reuso”. O framework proposto tem dois níveis, como mostra a Figura 7:



**Figura 7:** O Framework RiSE para Reuso de Software [Almeida et al. 2004]

No primeiro nível, formado pelas boas práticas, são considerados aspectos não técnicos como educação, treinamento, incentivos, programa para introduzir reuso e

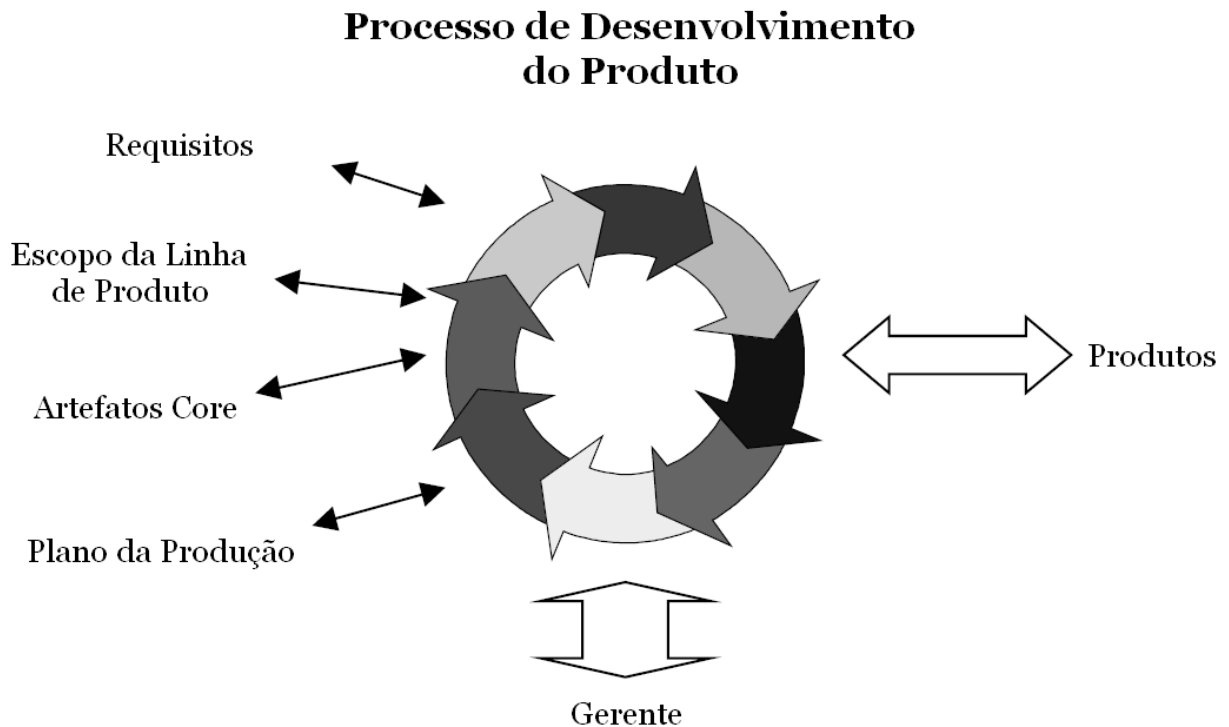
gerenciamento organizacional. O segundo nível é formado por importantes aspectos técnicos relacionados ao reuso de software como processos, ambiente e ferramentas.

Dessa forma, o processo RiSE para Engenharia de Domínio considera diversos aspectos do reuso de software como por exemplo, ambientes de reuso, gerenciamento de componentes e certificação de componentes. Entretanto, o processo RiSE foca na Engenharia de Domínio que é considerado o aspecto chave para o sucesso no reuso.

### **3.2 Linha de Produtos de Software**

Uma nova tendência iniciada para explorar o processo de reuso de software é a Linha de Produto de Software. De acordo com [CLEMENTS; NORTHROP 2001] uma linha de produto de software é “um conjunto de sistemas de software associados em comum, gerenciando um conjunto de características que satisfaça as necessidades específicas de um segmento de mercado particular ou destino e que são desenvolvidas a partir de um conjunto comum de artefatos principais em uma forma prescrita”.

A Figura 8 descreve o processo de desenvolvimento de produto com elementos ao redor. Vários elementos afetam o processo, assim como os requisitos, o escopo da linha de produto, os artefatos core e o plano do produto [NORTHROP 2002].



**Figura 8:** Desenvolvimento de Produto no conceito de linha de produto [NORTHROP 2002]

A proposta das Linhas de Produtos de Software é explorar pontos que a Engenharia de Domínio não mostrou eficiência. Segundo [BAYER et al. 1999] existem basicamente três aspectos que evidenciam falhas na Engenharia de Domínio, são eles: desvio da área de aplicação, falta de direção operacional e foco que ultrapassa as questões organizacionais. Dessa forma, as seções a seguir apresentam abordagens de Linhas de Produtos de Software que complementam ou apresentam alternativas aos processos de Engenharia de Domínio.

### 3.2.1 Family-Oriented Abstraction Specification and Translation Process (FAST)

No final dos anos 90 David Weiss apresentou um prático processo de produção de software orientado à família, o processo é conhecido como *Family-Oriented Abstraction Specification and Translation Process – FAST*. O processo FAST é aplicável em organizações que criam múltiplas visões de um produto que compartilha importantes atributos comuns, assim como o comportamento, interfaces ou código.

O principal objetivo do processo FAST é fazer um processo de engenharia de software mais eficiente pela redução de múltiplas tarefas, descrição dos custos de produção e pela redução do tempo de mercado [WEISS; LAI 1999]. No processo de engenharia de linha de produto definido por FAST são definidos três sub-processos: O primeiro deles é a Qualificação do Domínio que recebe as necessidades gerais da linha de negócios e produz um econômico modelo para estimar o número e o valor dos membros da família e o custo para produzi-lo.

O segundo sub-processo é a Engenharia do Domínio, seus objetivos são: gerar uma linguagem específica para especificação dos membros da família, um ambiente para a geração dos membros da família da especificação e um processo de produção dos membros da família usando o ambiente. O terceiro e último sub-processo é a Engenharia de Aplicação que gera os membros da família em resposta aos requisitos do consumidor.

O método FAST nasceu na indústria e teve um alto respaldo prático. Portanto, FAST é mais recomendado aos engenheiros de software e projetistas que trabalham na indústria. O uso do método FAST é motivado pelos problemas que faz da tarefa de desenvolver software uma atividade longa e custosa. Porém, algumas atividades no processo assim com na Engenharia de Domínio não são tão simples de serem executadas, por exemplo, a especificação de uma Linguagem de Modelagem de Aplicação (Application Modeling Language – AML) bem como projetar o compilador para gerar os membros da família.

### **3.2.2 Engenharia de Linha de Produto FORM**

O processo de Engenharia de Linha de Produto FORM representa uma extensão do *Feature-Oriented Reuse Method – FORM* descrito em [KANG et al. 1998]. Ele foi desenvolvido com a finalidade de suportar o desenvolvimento de linhas de produto de



software [KANG; LEE; DONOHOE 2002]. As extensões ao método FORM consistem no Desenvolvimento de Asset e no Desenvolvimento do Produto.

Desenvolvimento de *Asset*, possui três processos: (i) análise da linha de produto: entre outros aspectos tem-se o marketing, o desenvolvimento do plano do produto, refinamento, modelagem das *features* e a análise dos requisitos, (ii) desenvolvimento das arquiteturas: compreende a identificação dos componentes em alto-nível, a especificação dos dados, controle e a suas dependências (iii) resultado da análise baseado nos componentes reusáveis: refinamento dos componentes arquiteturais em componentes concretos. O Desenvolvimento do Produto inclui a análise dos requisitos, a seleção das *features*, seleção e adoção da arquitetura, adoção dos componentes e a geração de código para o produto.

A maior contribuição do processo de Engenharia de Linha de Produto FORM trata dos aspectos de negócio assim como o *Marketing and Product Plan – MPP* descrito na fase de desenvolvimento dos *assets*. Segundo [KANG; LEE; DONOHOE 2002] “com um MPP o reuso não é oportunista, ele é cuidadosamente planejado para uma linha de produto específica. Nossos consumidores têm aplicado esse método para vários domínios de aplicação industrial com a finalidade de criar ambientes de engenharia de software e *assets* para uma linha de produto”.

### 3.2.3 Komponentbasierte Anwendungsentwicklung (KobrA)

Desenvolvida por [ATKINSON; BAYER; MUTHING 2000], a abordagem KobrA é um processo de engenharia de linha de produto baseada em componentes, que descreve uma forma para desenvolver *assets* genéricos que podem acomodar variações de uma linha de produto através da engenharia de *framework*.

A engenharia de *framework* é a principal atividade definida por Kobra, iniciando com o projeto em um contexto no qual produtos de uma linha de produto são usados. O contexto inclui informação do escopo, dos pontos em comum e variações da linha de produto. Então, os requisitos da linha de produto são analisados e as especificações do componente Kobra, conhecido como *Komponent*, são desenvolvidas. Baseado nessas especificações, as realizações de *Komponent*, o qual descreve o projeto que satisfaça os requisitos, são desenvolvidos.

A abordagem Kobra ainda fornece um modelo de decisão que contém a seleção de variações para a configuração válida dos produtos. Kobra inclui ambos processos e técnicas para a engenharia de linha de produto.

### **3.2.4 Pervasive Component Systems (PECOS)**

O processo *Pervasive Component Systems – PECOS* representa um esforço para aplicar os conceitos de reuso no domínio dos sistemas embarcados [WINTER; ZEIDLER; STICH 2002]. Apresenta como motivação para essa tarefa a dificuldade para manter, atualizar, customizar e portar os sistemas embarcados em outras plataformas.

PECOS foca em dois aspectos centrais, o primeiro é como permitir o desenvolvimento de famílias de dispositivos PECOS e o segundo aspecto diz respeito à composição de um dispositivo PECOS, de seus *assets*, os componentes pré-fabricados e como esses componentes devem ser acoplados, não somente no nível funcional mas também no nível não-funcional.

### **3.2.5 Product Line UML-Based Software Engineering (PLUS)**

O método *Product Line UML-Based Software Engineering – PLUS* é uma abordagem de modelagem baseado na UML para o desenvolvimento de sistemas simples com suporte às linhas de produto de software [GOMAA 2005]. PLUS corresponde à abordagem mais recente de linhas de produto e fornece várias técnicas de modelagem e notações para a engenharia de linha de produto.

Na atividade de engenharia de requisitos, a modelagem dos casos de uso e a modelagem de *features* são fornecidas. Em seguida, para a atividade de análise, a modelagem estatística, a modelagem de interação dinâmica, a modelagem de máquina de estados dinâmica e a modelagem de dependência *feature*/classe são introduzidas. Por fim, para a atividade de projeto, os padrões de arquitetura de software e o projeto de software baseado em componentes são propostos.

### **3.2.6 Component-Oriented Platform Architecting Method (CoPAM)**

A abordagem *Component-Oriented Platform Architecting Method – COPAM* foi inicialmente desenvolvida pelos laboratórios de pesquisa da Philips. COPAM é um dos resultados do projeto Gaudi [PHILIPS 2000] que tem como objetivo “emergir uma metodologia de arquitetura de sistema mais acessível e transferir esse saber fazer e habilidade para uma nova geração de arquitetos de sistema”.

O objetivo principal do método CoPAM é conseguir adequar da melhor forma possível negócios, arquitetura, processos e organização. O objetivo específico do projeto da arquitetura é encontrar um meio termo entre abordagens baseada em componentes e centrada em arquitetura. A abordagem baseada em componentes é uma abordagem *bottom-up* dependendo da composição, enquanto que a abordagem centrada na arquitetura é *top-down* dependendo da variação [AMERICA et al. 2000].

O processo CoPAM é formado por duas atividades a engenharia da plataforma que consiste no desenvolvimento de uma plataforma com um número de componentes reusáveis e a engenharia do produto que desenvolve produtos usando os componentes da plataforma, adicionando novos componentes quando necessário.

### **3.3 Considerações Finais**

A evolução dos processos de reuso de software mostra novas tendências no levantamento de requisitos através de *features*, obtenção da arquitetura referência do sistema através de padrões de projeto e a implementação de componentes considerando componentes de negócio. Deve-se ressaltar ainda, o direcionamento de processos de reuso para domínios específicos e o grande número de abordagens focadas na indústria. Diante desses cenários, os capítulos 4, 5, 6 e 7 mostrarão atividades, passos e tarefas que compõem o Processo de Engenharia de Domínio para o desenvolvimento de aplicações baseadas na tecnologia RFID inseridas no domínio das Cadeias de Suprimentos.

## 4. O Processo de Engenharia de Domínio

De acordo com o Capítulo 3, o reuso de software pode ser uma importante forma de desenvolver software, oferecendo benefícios relacionados à qualidade, produtividade e custos, principalmente, através de um processo de reuso bem definido. No entanto, os processos de reuso de software atuais apresentam brechas e falta de detalhes nas atividades chave, como, por exemplo, a engenharia de domínio. Outros requisitos, como a engenharia de aplicação, métricas, aspectos econômicos, reengenharia, adaptação e qualidade também são importantes, mas o foco nesta dissertação é a engenharia de domínio. Neste contexto, conforme [CAMPOS et al. 2008], esta seção apresenta uma visão geral do processo de engenharia de domínio proposto, seus fundamentos e as etapas.

### 4.1. Visão Geral do Processo

Conforme definido no capítulo anterior, a engenharia de domínio é uma atividade de coleção, organização e armazenamento de experiências passadas no desenvolvimento de sistemas ou partes dos sistemas em um domínio particular na forma de artefatos reusáveis (ou seja, produto de trabalhos reusáveis), bem como prover um meio adequado para reusar esses artefatos (ou seja, recuperação, capacitação, divulgação, adaptação, montagem e assim por diante) no desenvolvimento de novos sistemas [CZARNECKI; EISENECKER 2000]. Um processo de engenharia de domínio deve definir três etapas principais: Análise do Domínio (Domain Analysis – DA), Projeto do Domínio (Domain Design – DD), e Implementação do Domínio (Domain Implementation – DI). Em geral, o principal objetivo da Análise do Domínio é o escopo do domínio e a definição de um conjunto de requisitos reusáveis para os sistemas no domínio. Em seguida, o Projeto do Domínio desenvolve uma arquitetura comum

para o sistema no domínio e elabora um plano de produtos. Finalmente, a Implementação do Domínio implementa os artefatos reusáveis, por exemplo, componentes reusáveis, linguagens específicas do domínio, geradores, e um processo de produção [CZARNECKI; EISENECKER 2000].

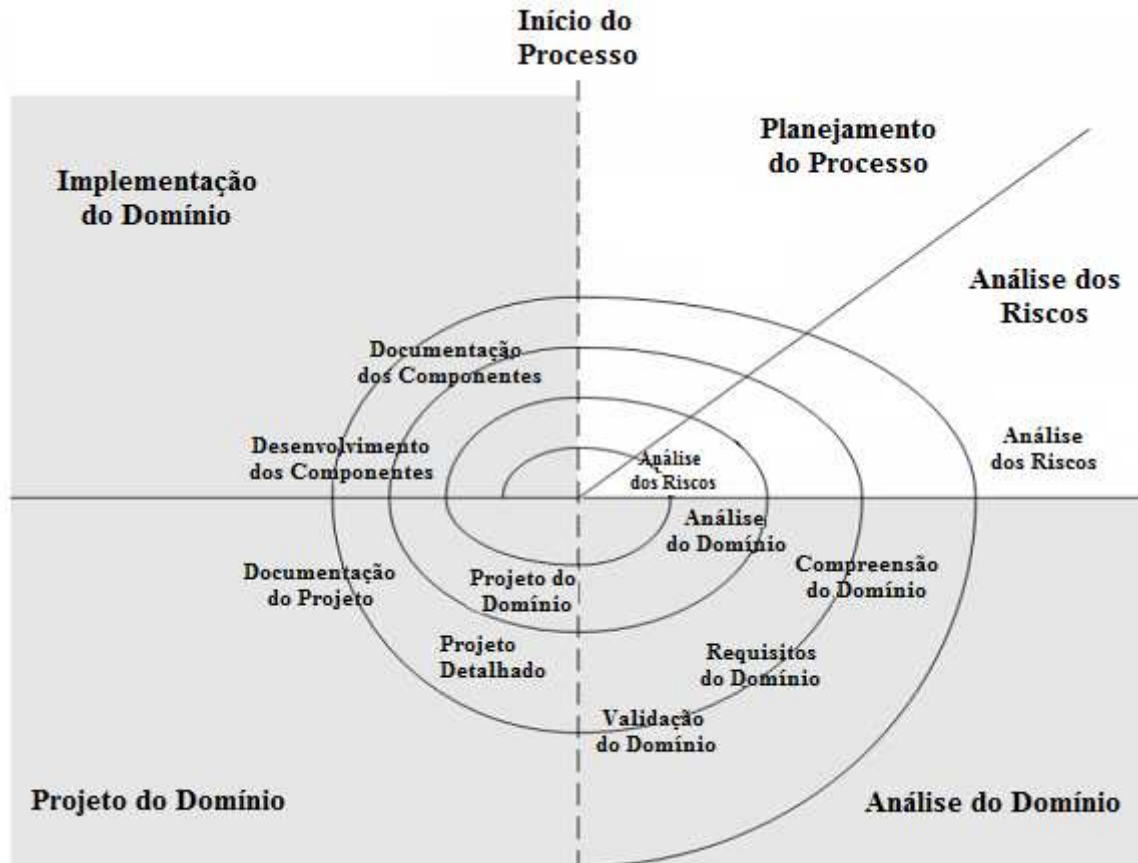
Antes de apresentar mais detalhes sobre o processo de engenharia de domínio proposto, é importante discutir dois conceitos básicos que serão utilizados a seguir: (i) Domínio: compreende não somente o conhecimento do mundo real em uma área de um determinado problema, mas também o conhecimento sobre como construir sistemas de software nesta área correspondente ao domínio de um conjunto de sistemas de ponto de vista, (ii) Característica: é amplamente utilizado na análise do domínio para capturar os pontos comuns e variabilidades dos sistemas em um domínio. Em geral, existem duas definições de características (features) encontradas na literatura de reuso. A primeira diz que uma característica é visível ao usuário final de um sistema, a definição FODA. A segunda definição sobre característica diz que é um recurso distinguível de um conceito (por exemplo, sistema, componente, etc) que são relevantes para alguns participantes do conceito, a definição ODM [SIMOS et al. 1996]. Neste trabalho, a primeira definição será utilizada, pois é a base para a área de análise do domínio. Outros conceitos importantes, mais específicos para cada etapa do processo, serão apresentados mais tarde, juntamente com o processo de descrição detalhada.

### **4.2. Os Fundamentos**

Um processo de desenvolvimento de software pode ser entendido como o conjunto de atividades necessárias para transformar requisitos de um usuário para um sistema de software. A forma como ele é feito pode mudar de processo para processo. Por exemplo, os processos podem ser focados em engenharia de domínio, serviços [PAPAZOGLU;

GEORGAKOPOULOS 2003], ou Desenvolvimento Orientado a Modelos (Model-Driven Development – MDD) [SCHMIDT 2006], ou utilizar os diferentes modelos de processos, porém, todo o tipo de processo é baseado em alguns fundamentos. Nesta seção, os fundamentos para o processo de engenharia de domínio, serão apresentados.

**Modelo de Processo.** Um modelo de processo de software é uma representação abstrata de um processo de software [SOMMERVILLE 2006]. Cada modelo de processo representa um processo a partir de uma perspectiva particular, e, portanto, fornece apenas informações parciais sobre esse processo. Existem modelos diferentes modelos de processo publicados na literatura de engenharia de software, tais como, o Modelo Waterfall, o Desenvolvimento Evolucionário, Engenharia de Software Baseado em Componentes (Component-Based Software Engineering – CBSE), a entrega incremental, o desenvolvimento espiral, entre outros [PRESSMAN 2005]. Estes modelos de processo são amplamente utilizados na prática atual de desenvolvimento de software. O processo de engenharia de domínio definido nesta dissertação é baseada no modelo de processo em espiral [BOEHM 1988], porém, apresenta algumas características do modelo de processo CBSE, uma vez que artefatos reusáveis são usados para desenvolver aplicações. O modelo em espiral proposto originalmente por Boehm (1988), ao invés de representar o processo de software como uma seqüência de atividades com algum retrocesso de uma atividade para outra, consiste em uma espiral, onde cada loop representa uma fase do processo de desenvolvimento de software. A Figura 9 mostra uma visão geral do processo de acordo com o modelo de processo espiral.



**Figura 9:** Modelo de processo do Processo de Engenharia de Domínio

**Domain Driven.** Em vez de processos de desenvolvimento de software tradicionais como, por exemplo, o RUP, que é orientado aos casos de uso, o processo de engenharia de domínio é orientado ao domínio, onde o foco está em um conjunto de aplicações para um domínio particular. Neste domínio, os pontos cruciais são: identificar as características comuns e específicas existentes nas futuras e potenciais aplicações; organizar essas informações em um modelo de domínio; em seguida, projetar a arquitetura de software específica do domínio (Domain-Specific Software Architecture – DSSA) e, finalmente, implementar componentes reutilizáveis para esse domínio. Mesmo no processo orientado ao domínio, casos de uso também serão usados.

**Arquitetura de Software.** A arquitetura de software envolve a estrutura e organização, através da qual os componentes do sistema moderno e subsistemas interagem para formar



sistemas; e as propriedades de sistemas que podem ser melhor concebidas e analisadas ao nível do sistema [KRUCHTEN et al. 2006].

**Desenvolvimento baseado em componentes (Component-Based Development – CBD).**

técnicas de desenvolvimento baseada em componentes são importantes porque, no projeto do domínio, por exemplo, é interessante modularizar a arquitetura nos componentes bem definidos, o que pode ser facilmente alterado, sem afetar outras partes da arquitetura. Além disso, na implementação do domínio, cujo objetivo é desenvolver ativos reusáveis, uma forma importante de fazer isso é através de um conjunto de componentes específicos do domínio, aumentando o potencial de reutilização.

**Padrão de Projeto.** Um padrão de design é uma forma de maior granularidade no reuso de classe, pois envolve mais de uma classe e a interligação entre objetos de diferentes classes. Do ponto de vista do processo de engenharia de domínio, padrões de projeto são importantes porque eles podem ser usados para encapsular as variabilidades existentes na análise do domínio e executar o mapeamento para o projeto.

### **4.3. Passos do Processo de Engenharia de Domínio**

O processo de Engenharia de Domínio definidos no presente trabalho é composto de três etapas: Análise do Domínio, Projeto do Domínio e Implementação do Domínio. Devido à amplitude, cada passo é, respectivamente, divididos em atividades e sub-atividades. Os passos são definidos para ser usado em seqüência. No entanto, mesmo com resultados menos ideais, elas podem ser usadas separadamente. Portanto, neste capítulo, cada etapa será tratada como uma abordagem para uma parte diferente do ciclo de vida do domínio da engenharia. Nesse sentido, há uma abordagem para a Análise do domínio, descritas no Capítulo 5, uma

abordagem para o Projeto do Domínio, descrito no Capítulo 6 e, finalmente, uma abordagem para a Implementação do Domínio, descrita no Capítulo 7.

## 5. Análise do Domínio

A análise do domínio é vista como uma importante fase no desenvolvimento de softwares reusáveis, pois busca capturar os pontos comuns e as variações dos sistemas de um domínio específico. A primeira definição para análise do domínio foi dada por Neighbors em 1980, para ele a análise do domínio “é a atividade de identificação dos objetos e operações de classe de sistemas similares em um domínio particular” [NEIGHBORS 1980]. Neighbors certificou que a identificação de operações comuns em um domínio diminuía o esforço nas demais fases de desenvolvimento do sistema.

Apesar de inovadora, a tese de Neighbors não apresentou passos consistentes de como executar a análise do domínio. Na tentativa de suprir essa necessidade deixada por Neighbors, trabalhos como [PRIETO-DIAZ 1990] e [ALMEIDA et al. 2005] vieram mostrar formas de executar a análise do domínio. Para Prieto-Diaz o objetivo da análise do domínio é: “encontrar formas para extrair, organizar, representar, manipular e compreender informações reusáveis, para formalizar o processo de análise do domínio e desenvolver tecnologias e ferramentas para suportá-las” [PRIETO-DIAZ 1990].

Nesse sentido, o presente trabalho definiu uma abordagem para análise do domínio com o objetivo de identificar e modelar pontos em comum e variações presentes nas aplicações de uma cadeia de suprimentos que incorpora a tecnologia RFID. A abordagem foi desenvolvida a partir do estudo dos onze processos de reuso de software apresentados no Capítulo 3 que buscou identificar objetivos, pontos positivos e pontos negativos de cada processo de reuso.

A Análise do Domínio está dividida em quatro tarefas: **(i)** Planejamento do domínio, **(ii)** Requisitos do domínio, **(iii)** Modelagem dos requisitos e **(iv)** Documentação dos requisitos. Apresentaremos a seguir a descrição dessas quatro tarefas e em seguida uma análise global da

Análise do Domínio para as aplicações que incorporam a tecnologia RFID na Cadeia de Suprimentos.

## 5.1. Planejamento do domínio

O planejamento do domínio é a primeira tarefa da análise do domínio, busca-se alcançar com essa tarefa uma completa descrição do domínio, a identificação dos *stakeholders* do domínio e a delimitação do escopo do domínio. Ao término do planejamento do domínio deve ser gerado um documento contendo as tarefas específicas de cada domínio envolvido na cadeia de suprimentos. O planejamento do domínio envolve as seguintes atividades: (i) descrição do domínio, (ii) identificação dos *stakeholders* e (iii) escopo do domínio.

### 5.1.1 Descrição do domínio

A descrição do domínio deve ser realizada observando, primeiramente, qual é a cadeia de suprimentos que deseja aplicar o processo de engenharia de domínio. Por exemplo, o domínio será a cadeia de suprimentos da carne bovina, de bens duráveis, bens de consumo, etc. Cada cadeia de suprimentos terá seu escopo e será a partir da descrição do domínio que surgirão os *stakeholders*, o conjunto de requisitos, entre outros.

A cadeia de suprimentos é definida por [LAMBERT; STOCK; VANTINE 1998] como “a integração do negócio desde os fornecedores até o usuário final que proporcionam os produtos, serviços e informações agregarem valores ao cliente”. Assim como Lambert, os estudos de [FLEURY 2000] e [ROSS 1998] não tornam explícito como a cadeia de suprimentos está segmentada, portanto, este trabalho considera a integração da cadeia de suprimentos composta três domínios: fornecedores, cadeia interna de suprimentos e clientes.

A cadeia interna de suprimentos compreende os distribuidores, atacadistas, varejistas, etc e, dessa forma, a cadeia de suprimentos não se limita a um número específico de domínios entre o fornecedor e o cliente. Independente do número de domínios entre o início e o fim da cadeia de suprimentos todos eles devem ser descritos em função da (i) Atividade: o objetivo do domínio dentro da cadeia de suprimentos, (ii) Entrada: qual(is) domínio(s) da cadeia de suprimentos recebe informações, (iii) Saída: para qual(is) domínio(s) da cadeia de suprimentos passa informações e (iv) Tecnologia: onde são aplicados os Sistemas RFID e quais os objetivos em utilizar-se dos Sistemas RFID no domínio.

### **5.1.2 Identificação dos *stakeholders***

A atividade de identificar os *stakeholders* compreende em analisar as pessoas que têm interesse definido sobre o resultado do projeto. Vários *stakeholders* podem ser identificados no desenvolvimento e utilização dos Sistemas RFID em uma cadeia de suprimentos, mas o analista do domínio não deverá definir *stakeholders* com o mesmo objetivo no projeto.

O *stakeholder* mais comum de uma cadeia de suprimentos que utiliza os Sistemas de Identificação por Radiofrequência é o Especialista da Tecnologia RFID. Ele deve ter experiência com a Rede EPCglobal, com os leitores, as tags, seus atributos, a compra, a instalação e o uso dos hardwares. Outros *stakeholders* comuns, mas não obrigatórios, que podem fazer parte dessa lista são: Analista do domínio, o Gerente do domínio e o Especialista de mercado.

O Analista do domínio é uma pessoa que conduz o processo de análise do domínio descrito no capítulo 5 e em [CAMPOS; ZORZO 2007]. O Gerente do domínio consiste em todas as pessoas responsáveis por seus domínios na cadeia de suprimentos. Finalmente, o Especialista de mercado reúne características como inteligência dos negócios, estudos de

competitividade e estimativa, segmentação do mercado, planos dos consumidores e a integração dessas informações em uma estratégia e plano de negócio coesivo.

### **5.1.3 Escopo do domínio**

Os domínios identificados e descritos na primeira atividade do planejamento do domínio precisam ser delimitados para evitar modelagem e implementação de módulos desnecessários futuramente. O primeiro passo para a delimitação do domínio é a identificação e eliminação dos domínios que foram descritos, mas não mantiveram ligação com qualquer outro domínio da cadeia de suprimentos. A identificação desses domínios será a partir das entradas e saídas geradas por eles, ou seja, domínios que não recebem e não produzem dados para outros domínios estarão na lista das exclusões.

O segundo passo é analisar o escopo do domínio de forma horizontal, esse tipo de análise tem o objetivo de responder a seguinte pergunta: Quantos sistemas diferentes estão no domínio? Por exemplo, em uma cadeia de suprimentos de automóveis a composição do produto final irá interagir com a cadeia de suprimentos de pneus, portanto, os sistemas do segundo domínio não convêm descrever na cadeia de suprimentos de automóveis. Por outro lado, o sistema da concessionária de automóveis fará parte do domínio, pois as solicitações de reposição de estoque, a contratação de pessoal, o gerenciamento dos Sistemas RFID desse domínio, etc estarão associados a esse sistema que faz parte da cadeia de suprimentos de automóveis.

Outro passo do escopo do domínio é a análise vertical dos sistemas do domínio, essa análise busca responder quais partes dos sistemas estão no domínio. Enquanto a análise horizontal identifica os diferentes sistemas, a análise vertical busca eliminar da abordagem as partes dos sistemas que não correspondem ao domínio. Por exemplo, em uma cadeia

produtiva de carne bovina, as aplicações responsáveis por coordenar a venda nos frigoríficos estarão fora da modelagem.

A atividade de determinar o escopo do domínio não é uma tarefa fácil, por esse motivo, as organizações que não tiverem experiência com o reuso de software e a engenharia de domínio, devem determinar o menor escopo possível para o domínio. Posteriormente novos domínios podem ser adicionados e maximizar o reuso de modelos e componentes no domínio.

## 5.2 Requisitos do domínio

A segunda tarefa da análise do domínio definida no trabalho é a definição dos requisitos do domínio. O objetivo dessa tarefa é descrever as características do domínio e dos sistemas que refletem o domínio. O processo de identificação dos requisitos do domínio envolve gerentes, engenheiros, usuários finais, etc que foram identificados na atividade de identificação dos *stakeholders*.

Entretanto, a atividade de identificar os requisitos do domínio a partir das necessidades dos *stakeholders* não é uma atividade das mais fáceis. Isso ocorre por vários motivos (**M**), como: **M1**. ambigüidade: *stakeholders* que não sabem o que eles desejam no sistema e passam informações que podem ser interpretadas de forma distinta. **M2**. redundância: requisitos de diferentes *stakeholders* interpretados da mesma forma. **M3**. conflito de requisitos: diferentes *stakeholders* com requisitos conflitantes. **M4**. fatores externos: requisitos do domínio influenciados por fatores políticos ou organizacionais. **M5**. evolução dos *stakeholders*: novos *stakeholders* podem surgir e mudar os requisitos do domínio. **M6**. evolução dos requisitos: mudança dos requisitos durante o processo de análise do domínio.

Na tentativa de minimizar os erros que ocorrem no levantamento dos requisitos do domínio, o trabalho faz a identificação dos requisitos através de *features*. A definição de

*feature* adotada neste trabalho está em concordância com [SIMOS et al. 1996] que considera *feature* como “características distintas do conceito (por exemplo, sistema, componente, etc) que é relevante para alguns *stakeholders* do conceito”.

Uma vez definida a forma para extrair os requisitos do domínio, faz-se necessário definir como extraí-los. Para essa etapa do levantamento dos requisitos o trabalho propõe a análise dos possíveis cenários do domínio, ou seja, como os sistemas do domínio são usados na prática.

Os passos (**P**) para o levantamento dos requisitos a partir de cenários são: **P1.** estado inicial: analisar e descrever como estava o estado dos sistemas no início do cenário. **P2.** eventos: verificar o fluxo normal dos eventos no cenário. **P3.** eventos alternativos: verificar a existência de fluxos concorrentes ao fluxo normal e a ocorrência de fluxos que venham gerar erros. **P4.** estado final: analisar e descrever o estado dos sistemas ao término do cenário. **P5.** *stakeholders*: listar os *stakeholders* que tiveram participação no evento.

Com os cenários do domínio descritos, o analista do domínio terá as *features* mais facilmente e poderá identificar os requisitos comuns e variáveis no domínio. Um artefato que pode ser gerado nessa atividade é o diagrama de estados finitos para cada cenário analisado. A notação *Unified Modeling Language – UML* define o diagrama de estados que pode ser utilizado pelo analista do domínio como ferramenta para a composição do artefato.

### 5.3 Modelagem dos requisitos

A modelagem dos requisitos corresponde à segunda tarefa da análise do domínio, nela o analista do domínio busca identificar e modelar os requisitos comuns e variáveis do domínio. Como discutido na atividade de requisitos do domínio, as *features* constituem uma grande



contribuição da engenharia de domínio em relação à engenharia de sistemas convencionais, pois facilita a identificação dos requisitos e facilita o reuso de software.

Nos Sistemas RFID, a identificação dos requisitos comuns e dos requisitos variáveis terá como base a Rede EPCglobal, ou seja, as primeiras características serão extraídas com base nas definições trazidas pela arquitetura da Rede EPCglobal. Por exemplo, o número EPC, o Sistema de Informação, o Serviço de Descoberta etc. Nesse sentido, a modelagem dos requisitos será realizada obedecendo as seguintes atividades: requisitos comuns e requisitos variáveis, que são detalhados a seguir.

### **5.3.1 Requisitos comuns**

A definição dos requisitos comuns é uma importante atividade no processo de análise do domínio, pois quanto mais requisitos comuns menor serão os requisitos variáveis e maior será o reuso de software no domínio. O primeiro passo para a identificação dos requisitos comuns do domínio é a exploração dos requisitos de todas as aplicações previstas para o domínio. Os requisitos que forem comuns para essas aplicações são candidatas para serem requisitos comuns.

A abordagem definida neste trabalho para identificar requisitos comuns é através da matriz de requisitos associada à prioridade, como mostra a Tabela 3. A matriz de requisitos do domínio tem a finalidade de auxiliar a identificação de requisitos comuns aos domínios do sistema a partir de sua prioridade para os *stakeholders*.

A coluna mais à esquerda da Tabela 3 representa os requisitos identificados na atividade de requisitos do domínio para os diversos domínios. Na primeira linha estão listados os domínios descritos na atividade de descrição do domínio.

**Tabela 3:** Matriz de requisitos do domínio

	Domínio 1	Domínio 2	Domínio 3	...	Domínio n
Req. 1	P2	P1	P2	-	-
Req. 2	X	P2	P3	-	-
Req. 3	P1	P1	P1	-	-
...				-	-
Req. n	-	-	-	-	-

O corpo da Tabela 3 deve ser preenchido com a prioridade (*P*) dos requisitos em cada domínio. As prioridades seguem a seguinte classificação: *P1*: *Alta*: requisito indispensável para o domínio, *P2*: *Média*: requisito que auxilia os requisitos de alta prioridade a manter a funcionalidade do sistema, *P3*: *Baixa*: requisito de baixa importância para o sistema.

Após o preenchimento da tabela, o analista do domínio poderá definir a prioridade satisfatória dos requisitos comuns e classificá-los como requisitos comuns do domínio. Na Tabela 3, por exemplo, caso o analista do domínio tenha definido a prioridade *P2* satisfatória para os requisitos comuns, os requisitos 1 e 3 seriam selecionados. Para os requisitos que não se aplicarem ao domínio, a tabela deve ser preenchida com um X, como ilustrado na Tabela 3 para o Requisito 2 do Domínio 1.

### 5.3.2 Requisitos variáveis

Após identificar os requisitos comuns do sistema, o analista do domínio deve analisar os pontos de variação do domínio e identificar os requisitos variáveis. Estudo como o de [SVAHNBERG et al. 2001] mostra que, quanto mais requisitos variáveis forem preservados no processo de desenvolvimento dos sistemas, maior será a flexibilidade e o reuso no domínio.

Nesse sentido, o levantamento dos requisitos variáveis do domínio é considerado uma importante tarefa, pois possibilita aos sistemas desenvolvidos adaptar-se as novas situações.

O processo mais simples para identificação dos requisitos variáveis do domínio é aplicar a operação lógica XOR (ou exclusivo) de todos requisitos levantados no domínio com os requisitos comuns. Dessa forma, os requisitos que não forem comuns serão requisitos variáveis do domínio.

Outra forma de identificar e gerenciar as variações em um domínio é através das *features* do domínio. Em [KANG et al., 1990] e [KANG et al., 1998] é apresentado um modelo de *features* (*feature models*) no qual, as *features* são agrupadas hierarquicamente estabelecendo relacionamentos do tipo: *composed-of*, *generalization/specialization* e *implemented by*. A notação utilizada neste trabalho seguirá a proposta pelo trabalho de [CZARNECKI; EISENECKER 2000] que classifica as *features* de quatro formas: *mandatory*, *optional*, *alternative* e *or-feature*.

As *mandatory features* são características que identificam o domínio, ou seja, àquelas características presentes em todas as instâncias do domínio. As *optional features* são características que adicionam algum valor ao domínio quando habilitada. *Alternative features* é o tipo de característica que desabilita as demais *alternative feature* quando habilitada, semelhante à operação lógica XOR (ou exclusivo). Por fim, as *or-features* são características que podem ser habilitadas em grupo, diferentemente das *alternative feature* permite a ativação de uma característica por grupo. Os diagramas definidos por Czarnecki & Eisenecker para representar as *features* são apresentados na Figura 10.

De acordo com [CZARNECKI; EISENECKER 2000], as *features* *f1*, *f2*, *f3* e *f4* da Figura 10a são *mandatory features* do conceito *C*. A figura simboliza ainda que as *features* *f1* e *f2* são instâncias do conceito *C* e todas as instâncias de *C* na qual tem *f1* também terá *f3* e *f4*. Portanto, as instâncias de *C* podem ser descritas pelo conjunto de *features* {*C*, *f1*, *f2*, *f3*, *f4*}.

Para as *optional features*, a associação entre os nós é finalizada com um círculo vazio, como mostra a Figura 10b. Dessa forma, as *features*  $f_1$ ,  $f_2$  e  $f_3$  são *optional feature* do conceito  $C$  e uma instância de  $C$  pode ter uma das seguintes descrições:  $\{C\}$ ,  $\{C, f_1\}$ ,  $\{C, f_1, f_3\}$ ,  $\{C, f_2\}$ ,  $\{C, f_1, f_2\}$ , ou  $\{C, f_1, f_2, f_3\}$ .

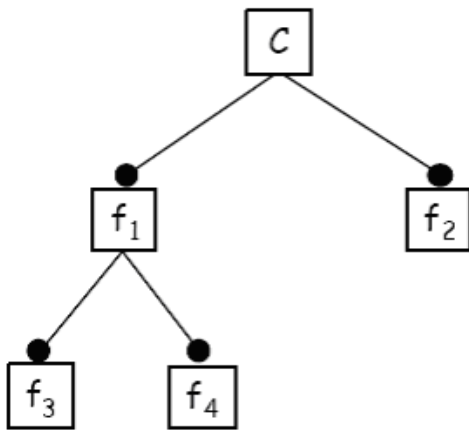


Figura 10a: Mandatory feature

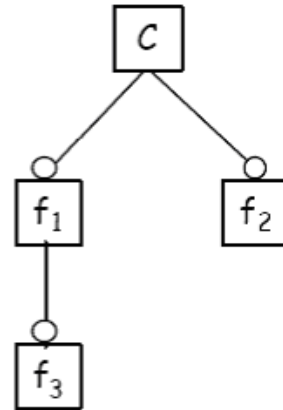


Figura 10b: Optional feature

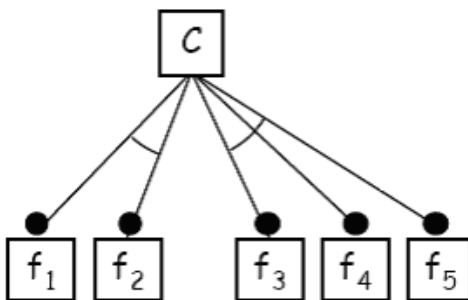


Figura 10c: Alternative feature

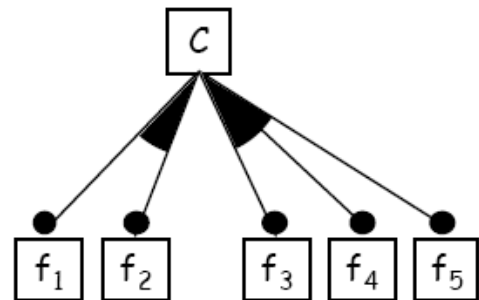


Figura 10d: Or-feature

Figura 10: Exemplos dos diagramas de *features*[CZARNECKI; EISENECKER 2000]

O terceiro tipo dos diagramas de *features* representada na Figura 10c corresponde às *alternative features*. Na representação gráfica, os nós do conjunto de *alternative features* são conectados por um arco, assim tem-se que o conceito  $C$  possui dois conjuntos de *alternatives features*: um conjunto com  $f_1$  e  $f_2$  e outro conjunto com  $f_3$ ,  $f_4$  e  $f_5$  que deriva as seguintes descrições:  $\{C, f_1, f_3\}$ ,  $\{C, f_1, f_4\}$ ,  $\{C, f_1, f_5\}$ ,  $\{C, f_2, f_3\}$ ,  $\{C, f_2, f_4\}$  ou  $\{C, f_2, f_5\}$ .

Finalmente, a Figura 10d mostra a representação gráfica das *or-features*, onde os nós de um conjunto de *or-features* são conectados por um arco cheio. Como o conceito  $C$  tem dois conjuntos de *or-features*: um conjunto com  $f_1$  e  $f_2$  e outro conjunto com  $f_3$ ,  $f_4$ , e  $f_5$  as

diferentes descrições possíveis são:  $\{C, f1\}$ ,  $\{C, f2\}$ ,  $\{C, f1, f2\}$ ,  $\{C, f1, f3\}$ ,  $\{C, f1, f4\}$ ,  $\{C, f1, f5\}$ ,  $\{C, f1, f3, f4\}$ ,  $\{C, f1, f3, f5\}$ ,  $\{C, f1, f4, f5\}$ ,  $\{C, f1, f3, f4, f5\}$ ,  $\{C, f2, f3\}$ ,  $\{C, f2, f4\}$ ,  $\{C, f2, f5\}$ ,  $\{C, f2, f3, f4\}$ ,  $\{C, f2, f3, f5\}$ ,  $\{C, f2, f4, f5\}$ ,  $\{C, f2, f3, f4, f5\}$ ,  $\{C, f1, f2, f3\}$ ,  $\{C, f1, f2, f4\}$ ,  $\{C, f1, f2, f5\}$ .

As raízes das árvores são chamadas de conceitos, segundo [CZARNECKI; EISENECKER 2000] são elementos e estruturas no domínio de interesse inteiramente subjetivo, ou seja, suas informações dependem não somente de pessoas mas também do tempo, do contexto e outros fatores. Para os Sistemas RFID, o analista do domínio também poderá classificar um nó da árvore como uma *feature* assim como as demais. Isso ocorre porque pode existir nós considerados objetivos e que possibilitam a agregação de outras *features*. A definição de conceito trazida por [CZARNECKI; EISENECKER 2000] aplicar-se-á, obrigatoriamente, apenas na raiz principal.

#### 5.4 Documentação dos requisitos

A quarta e última tarefa da análise do domínio é a documentação dos requisitos, nela os requisitos podem ser documentados usando texto em linguagem natural ou linguagem de modelagem de requisitos. Apesar da ambigüidade inerente à documentação usando linguagem natural, o trabalho usa essa opção em concordância com [CZARNECKI; EISENECKER 2000], pois os requisitos foram modelados em forma de *features* na atividade de requisitos variáveis. O *template* definido por Czarnecki & Eisenecker para a documentação de *features* pode ser visualizado na Tabela 4 com as alterações que os Sistemas RFID exigem:

**Tabela 4:** *Template* para Documentação de *Features*

<b>Nome da Feature:</b>
<b>Descrição Semântica</b>
Cada <i>feature</i> deve ter uma pequena descrição semântica contextualizada com os Sistemas RFID.
<b>Razão</b>
As <i>features</i> devem ter uma nota explicando porque ela está incluída no modelo e se essa <i>feature</i> é exclusiva dos Sistemas RFID.
<b>Stakeholders e programas clientes</b>
Cada <i>feature</i> deve ser relacionada com <i>stakeholders</i> que tenham interesse na <i>feature</i> e o programa cliente que tem necessidade por esta <i>feature</i> .
<b>Aplicações exemplos</b>
A documentação pode descrever <i>features</i> com conhecimento das aplicações que vão implementá-las. Opcionalmente, o analista do domínio pode exemplificar aplicações baseadas na tecnologia RFID fora do domínio da Cadeia de Suprimentos.
<b>Restrições</b>
Dois importantes tipos de restrições são: a exclusão mútua e as restrições requeridas.
<b>Ponto de Variação</b>
Pontos de variação podem ser marcados com “abrir” se novas sub- <i>features</i> (ou <i>features</i> ) forem esperadas. Por outro lado, marca-se um ponto de variação com “fechado” quando não se espera sub- <i>features</i> .
<b>Prioridades</b>
Prioridades possivelmente são atribuídas à <i>features</i> com a finalidade de gravar sua relevância

no processo.

### **5.5 Considerações Finais**

A Análise do Domínio é uma importante fase nos processos de engenharia de domínio, pois delimita o domínio, orienta o levantamento dos requisitos, modela e documenta as características a serem projetadas e implementadas. Dessa forma, este capítulo identificou particularidades das aplicações baseadas na tecnologia RFID inseridas na Cadeia de Suprimentos e apresentou uma abordagem para a Análise do domínio específica e para estas aplicações. O próximo capítulo apresentará a segunda fase do processo, o Projeto do Domínio, com o objetivo de fornecer visões distintas da arquitetura do domínio e meio para a identificação e especificação dos componentes de software.

## 6. Projeto do Domínio

O Projeto do Domínio é a segunda fase no processo de Engenharia de Domínio definido neste trabalho. O principal objetivo desta fase está de acordo com [BOSCH 2000] que busca “a produção de uma arquitetura referência ou específica do domínio, definindo seus elementos e suas interlocuções”.

A definição para arquitetura de software adotada neste trabalho considera duas definições. A primeira é apresentada por [BASS; CLEMENTS; KAZMAN 2003] que considera arquitetura de software como “uma estrutura ou estruturas de sistema, no qual compreende elementos, suas propriedades visíveis externamente e o conjunto de seus relacionamentos”.

Outra definição para Arquitetura de Software considerada é a de [BUSCHMANN et al. 1996] que diz que: “uma arquitetura de software é uma descrição de subsistemas e componentes de um sistema de software e os relacionamentos entre eles. Subsistemas e componentes são tipicamente especificados em diferentes visões para mostrar a relevância funcional e propriedades não-funcionais de um sistema de software. A arquitetura de um sistema é um artefato. Ela é o resultado da atividade de desenvolvimento de software”.

Nesse sentido, o Projeto do Domínio descrito em [CAMPOS et al. 2008], mostrará como definir uma arquitetura baseada na tecnologia RFID e os passos a serem seguidos para a identificação e especificação de componentes. A abordagem do Projeto do Domínio está dividida em quatro tarefas: (i) Mapeamento, (ii) Projeto dos Componentes, (iii) Visões da Arquitetura e (iv) Documentação da Arquitetura, descritas nas seções a seguir.

### 6.1 Mapeamento



A primeira tarefa definida no Projeto do Domínio é o mapeamento entre os requisitos identificados na fase de Análise do Domínio para a arquitetura referência. Uma importante questão considerada nesta tarefa são os requisitos variáveis. Para [SVANHNBERG; GRUP; BOSH 2001], quanto mais requisitos variáveis forem preservados no processo de desenvolvimento dos sistemas, maior será a flexibilidade e a customização do sistema.

As Cadeias de Suprimentos apresentam diferentes particularidades, tanto na utilização da tecnologia RFID quanto na distribuição de sua linha de produção. Por esse motivo, o mapeamento dos requisitos deve preservar as variações a fim de possibilitar a aplicação do processo em diferentes domínios.

Outra questão que o projetista do domínio deve levar em consideração é na especificação dos componentes. Inevitavelmente, o projetista toma decisões que podem restringir o reuso dos componentes. Decisões que vão desde algoritmos específicos utilizados para implementar os componentes até os objetos e os tipos nas interfaces dos componentes. Quando essas decisões entram em conflito com requisitos específicos o reuso dos componentes será impossível ou o sistema ficará ineficiente.

Uma forma usada nos últimos anos para contornar essa situação é por meio dos padrões de projeto (*Design Pattern*). De acordo com [ALEXANDER 1977] “Cada padrão descreve um problema que ocorre várias vezes em um ambiente e descreve o núcleo da solução do problema, de tal forma que você possa usar essa solução milhões de outras vezes, sem fazer sempre da mesma forma duas vezes”. Dessa forma, o padrão é uma solução bem testada para um problema comum, mas que não descreve uma especificação detalhada.

Com Padrões de Projeto é possível reusar facilmente projetos e arquiteturas, tornar projetos mais acessíveis para desenvolvedores de novos sistemas, auxiliar a mudança de projeto de sistemas reusáveis mantendo o compromisso de reusabilidade, bem como prover documentação e manutenibilidade dos sistemas existentes.

Geralmente os padrões têm quatro elementos, um nome único para o padrão, o problema que o padrão está tentando resolver, a solução para o problema no contexto em que ele aparece e as conseqüências que a utilização do padrão acarreta.

Em concordância com [SVANHNBERG; GRUP; BOSH 2001], esse trabalho trata requisitos variáveis como importantes fontes de variação para a arquitetura referência e aplica padrões de projeto sobre essa classe de requisitos. Portanto, para o projeto das variações da arquitetura serão levadas em consideração as *features* alternativas, *features* opcionais e *or-features*. Os padrões adotados para serem usados nas *features* alternativas são *Abstract Factory*, *Chain of Responsibility*, *Factory Method*, *Observer*.

O padrão *Abstract Factory* que fornece uma interface para a criação de família de objetos relacionados ou dependentes sem especificar suas classes concretas. Ao especificar o nome da classe quando um objeto é criado o arquiteto do domínio corre o risco de forçar uma execução particular do objeto em vez uma simples relação particular com a classe. Dessa forma, o método *Abstract Factory* cria objetos independentemente e possibilita que apenas uma *factory* concreta seja criada para uma das *features* alternativas.

Na Cadeia de Suprimentos que utiliza RFID as leituras simultâneas de produtos realizadas por leitores RFID são situações comuns que mostram a aplicação do padrão *Abstract Factory*. O Middleware EPC ao receber dados provenientes de diversos leitores RFID deve indicar qual número EPC passar para o Serviço de Informação e não sobrecarregar as bases de dados com informações desnecessárias.

Outro padrão é o *Chain of Responsibility*, pois evita acoplar o remetente de uma requisição a seu receptor, dando a mais de um objeto a possibilidade de receber o pedido. Objetos no método *Chain of Responsibility* podem ter um tipo comum, mas usualmente eles não compartilham da mesma implementação. Nesse sentido, uma mesma requisição realizada em domínios distintos poderá ser solucionada por objetos diferentes. Por exemplo, a consulta

por um número EPC é encaminhada por meio de uma “cadeia de responsabilidades” até o Serviço de Descoberta que resolverá a busca por bases de dados localmente ou em bases de dados compartilhadas de outros domínios.

O terceiro padrão adotado é o *Factory Method* que define uma interface para criação de um objeto, mas permite subclasses decidirem qual classe instanciar. O *Factory Method* possibilita realizar um projeto mais customizado com um pouco mais de dificuldade. Outros padrões de projeto requerem novas classes, enquanto que no *Factory Method* apenas é requerida uma nova operação. O número EPC pode ser atribuído a qualquer objeto, produto, animal, entre outros, dessa forma, uma subclasse chamada produto poderá definir qual objeto instanciar.

Temos ainda o padrão *Observer* que define uma dependência um-para-muitos entre objetos, dessa forma, quando um dos objetos muda de estado, todas suas dependências são notificadas e atualizadas automaticamente. Usando esse padrão, as *features* alternativas selecionadas irão atualizar automaticamente as outras *features* vinculadas a ela. Nas Cadeias de Suprimentos, as tags RFID são utilizadas independentemente da aplicação por outro lado, caso o padrão do número EPC inserido nesta tag seja alterado de 96 bits para 256 bits as aplicações que tratam deste número devem se adequar para ao novo padrão de comunicação.

Para as *features* opcionais, este trabalho aplica padrões direcionados para as características adicionais do sistema, são eles: *Decorator*, *Prototype* e *Observer*.

O padrão *Decorator* é usado porque adiciona responsabilidades para um objeto dinamicamente. Como o próprio nome sugere, o método *Decorator* pode ser usado para as *features* opcionais principalmente porque elas são características adicionais. Dessa forma, se uma *feature* está presente, o *ConcreteDecorator* é responsável pelo gerenciamento e chama a execução. Nos Sistemas RFID está previsto um módulo de consulta aos produtos para

empresas ou pessoas autorizadas (gerentes, fornecedores, etc), dessa forma o padrão *Decorator* pode ser aplicado na criação de um módulo de consulta aos consumidores.

Outro padrão que pode ser utilizado nas *features* opcionais é o *Prototype* que especifica as qualidades dos objetos que serão usadas na criação de uma instância que servirá como um protótipo para criar novos objetos. O padrão *Prototype* especifica também como a interação com a *feature* deverá ocorrer que pela definição estabelece um protótipo concreto para cada *feature*. Assim, quando o Serviço de Informação EPC solicitar ao Serviço de Nomeação de Objetos que realize uma consulta a um produto desconhecido, uma cópia dos dados retornados (protótipo) será criada localmente. Por fim, o padrão *Observer* pode ser usado nas *features* opcionais da mesma forma que na *feature* alternativas.

Para as *or-features*, que representam as variações do domínio, têm-se três padrões de projeto que podem ser utilizados: o *Bridge*, o *Builder* e o *Director*. O padrão *Bridge* reduz a abstração de suas implementações de modo que os dois possam variar independentemente. Ele fornece uma interface estável aos clientes mesmo quando as classes que o implementam variam. No Serviço de Nomeação do Objeto – ONS existem consultas que podem retornar fontes de dados confiáveis ou não-confiáveis, entretanto, independente do tipo da base de dados existirá uma ponte (*bridge*) para efetuar a consulta a essas bases.

O padrão *Builder* que separa a construção de um objeto complexo de sua representação de modo que o mesmo processo de construção possa criar diferentes representações. O padrão *Builder* pode ser usado para desenvolver *features* compostas. Dessa forma, somente o *Director* está disponível, sendo responsável decidir qual *feature* estará na aplicação ou não. No transporte das *pallets* em uma Unidade de Transporte por exemplo, a aplicação decide qual será o tipo da Unidade de Transporte (caminhão, navio, avião, etc) e cria o objeto automaticamente considerando suas particularidades (tamanho, quantidade de produto transportada, responsáveis pelo transporte, etc).

Por fim, o padrão *Singleton* assegura uma classe ter somente uma instância, e fornece um ponto global para acessá-lo. Esse padrão também é fortemente recomendado para as *or-features* que se relacionam com *features* obrigatórias (*mandatory*). Dessa forma, o caminho a seguir será único e a classe deverá ser instanciada apenas uma única vez. Na Cadeia de Suprimentos o padrão *Singleton* será utilizado juntamente com o *Builder* após a representação do objeto estiver definida.

## 6.2 Projeto dos Componentes

O objetivo da fase de Projeto dos Componentes é criar um conjunto inicial de interfaces e especificações de componentes, juntamente com uma primeira visão da arquitetura de componentes.

### 6.2.1 Identificação das Interfaces

Esta seção descreve o projeto das interfaces de um componente baseado na UML 2.0. Uma interface é definida por [BOOCH; RUMBAUGH; JACOBSON 2005] como “uma coleção de operações que são usadas para especificar um serviço de uma classe ou componente”. Similarmente, [SZYPERSKI 2002] define interface como “um conjunto de operações, com cada operação definindo algum serviço ou função que o componente irá executar para o cliente”.

Em concordância com [CHEESMAN; DANIELS 2000] e [ALMEIDA 2007], as interfaces identificadas neste trabalho são do tipo *business* e de sistema. O conceito *business* ajuda a focar sobre a informação e o processo associado que o sistema necessitará gerenciar. Para [ALMEIDA 2007] o processo que identifica as interfaces do tipo *business* consiste em:

analisar o modelo de *feature* para identificar classes (para cada módulo e componente); representar as classes baseadas em *features* com atributos e multiplicidade; e refinar as regras de negócio usando linguagem formal.

Na identificação das interfaces do sistema o arquiteto do domínio deve, para cada caso de uso, considerar se não existem responsabilidades do sistema que devem ser modeladas. Nesse caso, são representadas como uma ou mais operações das interfaces.

### **6.2.2 Especificação dos Componentes**

Após a especificação das interfaces, as informações adicionais de especificação que o desenvolvedor do componente necessita ter consciência são as dependências de um componente em relação às outras interfaces. Os passos apresentados neste trabalho para a especificação dos componentes estão de acordo com a abordagem UML Components apresentada por Cheesman & Daniels (2000).

O primeiro passo consiste em especificar as interfaces usadas e oferecidas. Para cada especificação de componente é necessário dizer quais interfaces e realizações esse componente deve suportar. Assim como, é necessário confirmar qualquer restrição o qual outras interfaces são usadas para a realização.

Em seguida, deve-se especificar as restrições de interação entre componentes. Nessa fase são definidas as restrições de como uma operação particular deve ser implementada. Ao contrário das tradicionais interações no nível de implementação, interações de componente define restrições no nível de especificação.

### **6.3 Visões da Arquitetura**

Assim como a arquitetura de uma construção é representada usualmente usando diferentes visões, por exemplo: visão estática, visão dinâmica, especificação de materiais, etc a descrição adequada de uma arquitetura de software também requer múltiplas visões.

Diversas visões da arquitetura são importantes, pois descrevem diferentes aspectos do mesmo sistema. Dessa forma, as categorias de visões definidas nesse trabalho são: (i) visão de módulos, (ii) visão de processos, (iii) visão de *deployment* e (iv) visão de dados.

A Visão de Módulos mostra a decomposição do sistema em partes, cada parte representa um subsistema, componente, classes de objetos, interfaces e suas diferentes formas de se relacionarem. O objetivo principal na descrição e interligação dos módulos de um domínio é a identificação das relações entre os módulos. A visão de módulos define três tipos de relações, como mostra a Figura 11.

Apesar da Rede EPCglobal estabelecer relações entre módulos dos Sistemas RFID as aplicações em Cadeias de Suprimentos podem conter diferentes módulos e estabelecer suas próprias relações.

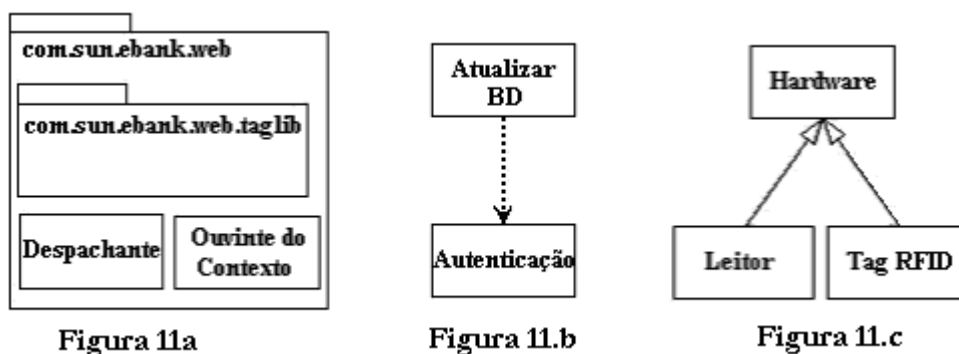


Figura 11: Relações em UML para a visão de módulos

O primeiro tipo de relação entre os módulos, visualizado na Figura 11a, é a “parte de” usada quando um pacote contiver sub-pacotes ou classes. A segunda relação, visualizada na Figura 11b é a “depende de” que estabelece uma relação de dependência entre os módulos mostra uma dependência definida pela Rede EPCglobal que é autenticação na rede para que uma aplicação efetue consultas. Por fim, a Figura 11c mostra uma relação do tipo “é um” com

a finalidade de representar os módulos que herda características de um módulo mais geral. A Figura 11c representa um módulo comum nos Sistemas RFID que são as heranças passadas pelo hardware aos leitores e tags RFID.

A notação mais apropriada para representar a visão de módulos é através de diagramas UML 2.0 [OMG 2005] entre eles, o diagrama de pacotes, o diagrama de componentes, o diagrama de classes e o diagrama de objetos. O diagrama de pacotes é o principal para a arquitetura, pois incorpora a decomposição de alto nível do sistema em componentes e seus relacionamentos. Cada componente no diagrama de pacotes é usado para descrever um projeto interno ou parte de um subsistema.

A Visão do Processo descreve o comportamento dos sistemas durante sua execução, ou seja, a decomposição do sistema em termos das atividades requisitadas, seus relacionamentos e o uso dos recursos em tempo de execução. A finalidade dessa visão é mostrar o funcionamento dos sistemas e analisar as propriedades que se manifestam em tempo de execução. Uma amostra de como é o funcionamento dos Sistemas RFID pode ser vista na Figura 6, ela mostra o processo de consulta à localização de um produto.

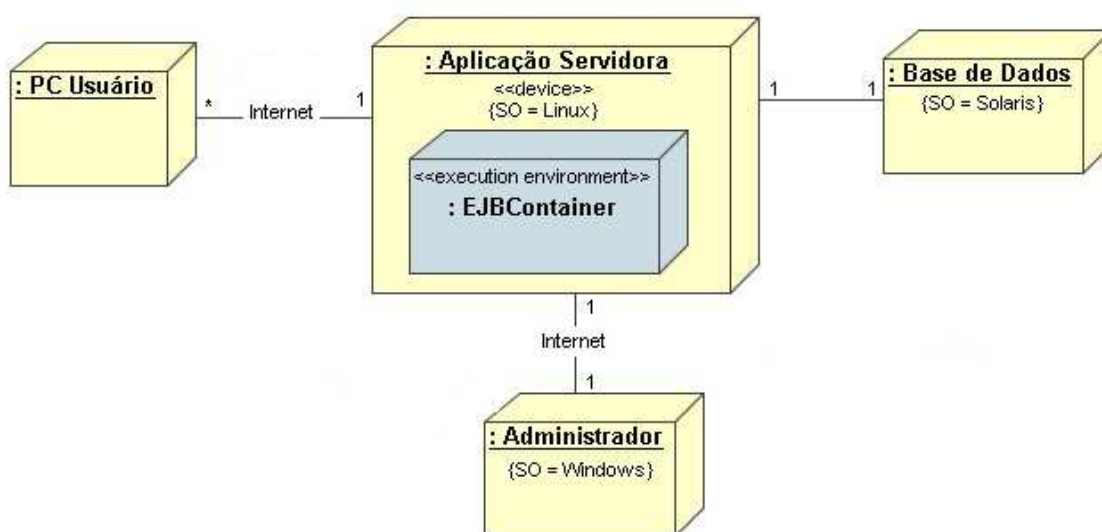
Para representar a visão do processo utilizando a UML 2.0 tem-se os seguintes diagramas: Diagrama de Interação, Diagrama de Tempo, Diagrama de Máquina de Estados, Diagrama de Atividades, Diagrama de Comunicação e o Diagrama de Seqüência. Juntamente com o diagrama de atividades, o diagrama de estados é o mais recomendado para descrever os processos que ocorrem na Cadeia de Suprimentos baseada na RFID, pois capturam o comportamento completo do sistema com mais detalhes e não levam em consideração aspectos de pouca relevância como o tempo por exemplo.

A Visão *Deployment* pretendida neste trabalho visa descrever a estrutura de hardwares na qual o sistema está sendo executado. Dessa forma é possível verificar a interligação dos



Serviços de Informação EPC, analisar o desempenho da Rede EPCglobal, questões de segurança e acesso de controle às bases de dados.

A UML 2.0 define o diagrama *deployment* com o propósito de modelar os hardwares usados na implementação do sistema, os componentes utilizados sobre os hardwares e as associações entre esses componentes. Dessa forma recomenda-se a utilização do diagrama *deployment* da UML 2.0 para representar graficamente a visão *deployment* desse trabalho, como mostra a Figura 12:



**Figura 12:** Exemplo da visão *Deployment*.

No diagrama *deployment* instâncias de artefatos são colocados em um nó, visualizado na Figura 12 como um cubo. Artefatos por sua vez, podem ser arquivos, bibliotecas, base de dados ou programas construídos ou modificados no projeto. Esses artefatos implementam coleções de componentes.

Por fim, a Visão dos Dados descreve a modelagem das bases de dados e o relacionamento existente entre elas. O objetivo dessa visão é melhorar o desempenho e a adaptabilidade dos sistemas, bem como, identificar redundância e inconsistência entre os dados. Sua aplicabilidade na Cadeia de Suprimentos está na representação das diversas bases de dados que armazenam informações sobre as tags RFID.

A visão dos dados inicia com um modelo conceitual e deve ser refinado até conter toda informação necessária para a criação da base de dados física. As notações mais indicadas para essa visão são os diagramas de entidade-relacionamento e o diagrama de classes da UML 2.0.

#### 6.4 Documentação da Arquitetura

Uma vez que as visões foram definidas e executadas, o arquiteto do domínio fará a documentação da arquitetura em termos das informações que se aplicam a mais de uma visão. Nesse sentido, o *template* apresentado na Tabela 5 foi definido a fim de auxiliar a documentação da arquitetura do sistema.

**Tabela 5:** *Template* para Documentação da Arquitetura

<b>Documento da Arquitetura</b>
<b>1. Guidelines de Utilização</b>
Descrever a forma que o documento da arquitetura está organizado, a melhor forma de utilização dos campos bem como cenários de utilização do documento na Cadeia de Suprimentos que está sendo projetada.
<b>2. Informações do Projeto</b>
Mostrar informações do projeto, como por exemplo, descrição da Cadeia de Suprimentos, versão dos Sistemas RFID utilizados, existência de documentos auxiliares ou anteriores, membros do projeto e seus objetivos em linhas gerais.
<b>3. Informações do Domínio</b>
Descrever o contexto do domínio a ser projetado, atributos de qualidade, requisitos funcionais e não-funcionais de maior relevância para os projetistas da Cadeia de

Suprimentos.
<b>4. Documentação das View:</b>
<b>4.1 Nome da View</b>
Além de tornar público o nome da <i>view</i> , este campo é reservado para mostrar que tipo de <i>view</i> será documentada e importância dessa <i>view</i> na Cadeia de Suprimentos.
<b>4.2 Representação Gráfica</b>
Descrever a arquitetura utilizando uma notação padronizada, reconhecida e de fácil adaptabilidade para futuros projetistas.
<b>4.3 Descrição dos Elementos</b>
Relação dos elementos utilizados na representação gráfica da <i>view</i> , as propriedades das interfaces, relacionamentos e comportamento dos Sistemas RFID de acordo com a <i>view</i> .
<b>4.4 Views Relacionadas</b>
Descreve o relacionamento da <i>view</i> documentada com outras <i>views</i> . Por exemplo, as bases de dados descritas na <i>view deployment</i> podem ser melhores visualizadas na <i>view</i> de dados.
<b>4.5 Outras Informações</b>
Observações importantes da <i>view</i> que merecem ser ressaltadas pelo arquiteto, bem como justificativas para decisões tomadas na descrição da <i>view</i> .
<b>5. Relações de Análise e Projeto</b>
Mostrar que todos os requisitos relacionados na fase de análise foram satisfeitos nos elementos da arquitetura.
<b>6. Glossário</b>
Mostrar o glossário do sistema e a lista de acrônimos.

### **6.5 Considerações Finais**

A fase de Projeto do Domínio visa descrever uma arquitetura referência para o domínio definindo seus elementos e suas interlocuções. A literatura mostra que os métodos existentes apresentam dificuldade em apresentar uma forma sistemática e completa para a descrição da Arquitetura referência. Dessa forma, este capítulo mostrou como aplicar padrões de projeto nos requisitos variáveis dos domínios e como representar graficamente a arquitetura do domínio em visões. O próximo capítulo apresentará a terceira e última fase do processo descrito neste trabalho com a finalidade de orientar a melhor forma para implementar e documentar componentes.

## 7. Implementação do Domínio

A Implementação do Domínio é a terceira e última fase no Processo de Engenharia de Domínio para o desenvolvimento de aplicações em Cadeias de Suprimentos baseadas na RFID. Em concordância [POHL; BOCKLE; LINDEN 2005] a finalidade dessa fase é fornecer o projeto detalhado, a implementação e a documentação de *assets* reusáveis, baseado na arquitetura descrita na fase do Projeto do Domínio.

As atividades definidas para a Implementação do Domínio estão de acordo com respeitados métodos de desenvolvimento baseado em componentes e processos de reuso de software, entre eles estão o UML Components [CHEESMAN; DANIELS 2000], Catalysis [D'SOUZA; WILLS 1998] e [ALMEIDA 2007]. As seções a seguir detalham as atividades da Implementação do Domínio descritas em [CAMPOS et al. 2008].

### 7.1 Implementação dos Componentes

Na atividade de implementação dos componentes o engenheiro do domínio deve ter os artefatos gerados na fase de Análise do Domínio (requisitos, modelo de *features* e cenários) e Projeto do Domínio (arquitetura do domínio e especificação dos componentes). A partir daí são executadas quatro tarefas, que serão descritas a seguir

A primeira tarefa é descrever o componente, o objetivo é fornecer informação da proposta geral do componente, assim como o fornecedor do componente, versão, tecnologia RFID e pacote. Ao descrever a tecnologia RFID o desenvolvedor deve ressaltar quais as influências e contribuições deixadas pelo componente para os Sistemas RFID. Essas informações podem importantes questões quando componentes são armazenados em um repositório.

A especificação das interfaces é a segunda tarefa, essa fase depende dos artefatos desenvolvidos na análise do domínio e no projeto do domínio, assim como a arquitetura e as especificações do componente. A tarefa de especificação das interfaces é descrita na seção 5.2 com maiores detalhes. Em seguida, tem-se a terceira tarefa, a implementação dos serviços que objetiva implementar os serviços definidos na fase anterior, usando qualquer tecnologia de implementação, assim como o código para registrar esses serviços para serem usados por outros componentes, se um ambiente de execução dinâmico é usado.

A quarta tarefa consiste no desenvolvimento e instalação do componente. De acordo com a tecnologia de implementação usada, isso envolve compilação e empacotamento do componente em uma forma que é adequado desenvolvê-lo no ambiente de produção. Por fim, na última tarefa, a reutilização de outros serviços, o engenheiro de software deve descrever o componente que irá reusar outros serviços.

### **7.2 Documentação dos Componentes**

Quando um componente é projetado e implementado, os desenvolvedores têm em mente alguns cenários de utilização e pode também ter especificado em função de um conjunto de casos de uso. Dessa forma, caso os clientes não compreendam a finalidade do componente e como o desenvolvedor esperava que ele fosse usado, farão a utilização errada do componente. No pior caso, os componentes serão descartados por não ter executado a tarefa como solicitado.

Nesse sentido, a documentação de componentes é apresentada por [KOTULA 1998] como “um caminho para a troca eficiente de informações e conhecimento entre o autor e o cliente. Ela é uma das mais importantes formas para melhorar a compreensão do programa e reduzir os custos com o software”. Fica evidente que, embora os desenvolvedores de

componentes se esforcem para criar interfaces fáceis de usar, alguns detalhes ainda devem ser comunicados aos clientes.

Como solução para a criação de uma boa documentação de componentes Kotula (1998) apresenta trinta e nove padrões inter-relacionados, agrupados em seis categorias: (i) *Generative Patterns*: descrevem em alto nível padrões criando padrão, (ii) *Content Patterns*: descreve o material que deve ser incluído na documentação, (iii) *Structure Patterns*: descreve como a documentação deve ser organizada, (iv) *Search Patterns*: discute as facilidades necessárias para encontrar uma informação específica, (v) *Presentation Patterns*: descreve como a documentação deveria ser apresentada graficamente e (vi) *Embedding Patterns*: fornece *guidelines* de como inserir o conteúdo da documentação no código fonte.

Por outro lado [TAULAVUORI; NIEMELA; KALLIO 2004] diz que “a definição de um padrão não é suficiente para a adoção de uma nova prática de documentação”. Dessa forma, Taulavuori et al., (2004) descreve um sistema de documentação de componente que foi desenvolvido para suportar o desenvolvimento de documentação de componente.

Após analisar padrões definidos por Kotula (1998) e a documentação de componentes, no contexto de linhas de produto de software, descrita em [TAULAVUORI; NIEMELA; KALLIO 2004], esse trabalho chegou ao seguinte *template* para a documentação dos componentes implementados mostrado na Tabela 6.

**Tabela 6:** *Template* para Documentação de Componente

<b>Documento de Componente</b>
<b>1. Informações Básicas</b>
Define a identificação, propriedades e informações gerais dos componentes.
<b>1.1 Nome</b>
Deve ser único, bem definido e descrever o componente.

<b>1.2 Tipo</b>
Deve expressar a forma de utilização pretendida para o componente. Por exemplo, um subsistema, um componente ator, um bloco de função ou um procedimento.
<b>1.3 Finalidade</b>
Descreve a proposta de um componente e sua relação com os Sistemas RFID inseridos na Cadeia de Suprimentos.
<b>1.4 História</b>
Descreve o ciclo de vida do componente, incluindo sua versão, a data de revisão, as pessoas que tem feito o trabalho e as modificações feitas no componente.
<b>2. Informações das Interfaces</b>
Define como um componente pode ser usado e interconectado com outros componentes.
<b>2.1 Interfaces Requeridas</b>
As interfaces requeridas devem ser documentadas em termos do nome, tipo, descrição, comportamento e finalidade da interface.
<b>2.2 Interfaces Providas</b>
As interfaces providas seguem os mesmos atributos das interfaces requeridas.
<b>3. Informações de Padrões</b>
Descreve os protocolos e os padrões definidos para o componente ser aplicado.
<b>3.1 Protocolo</b>
Descreve a interação entre dois componentes necessários para alcançar um objetivo específico.
<b>3.2 Padrão</b>
Deve informar quais os padrões da Rede EPCglobal (Versão de tag, leitor, ONS, etc) o componente está utilizando e quais os padrões que o componente necessita para seu correto funcionamento. Esses padrões podem restringir a compatibilidade e funcionalidade do



componente.
<b>4. Informações Técnicas</b>
Apresenta informações detalhadas sobre o projeto e a implementação do componente.
<b>4.1 Ambiente de Desenvolvimento</b>
Define o ambiente no qual o componente foi desenvolvido, por exemplo, Jini, Java, etc.
<b>4.2 Interdependências</b>
Descreve a dependência do componente com outros componentes.
<b>4.3 Pré-Requisitos</b>
Define todos os outros requisitos que o componente deve ter para operar.
<b>4.4 Implementação</b>
Descreve informações da estrutura interna do componente, os pontos de variação onde o comportamento do componente pode ser alterado e detalhes da implementação das interfaces.
<b>4.5 Restrições</b>
Descreve todos os itens que limitam as opções fornecidas para o projeto ou implementação do componente.
<b>5. Informações Não-Funcionais</b>
Descreve a qualidade de um componente e termos de:
<b>5.1 Adaptabilidade</b>
Define como o componente pode ser adaptado em novas Cadeias de Suprimentos com o menor custo.
<b>5.2 Confiança</b>
Define as vezes que o componente opera ou sua habilidade para executar as funções requeridas sob circunstâncias indicadas.

<b>5.3 Desempenho</b>
Define a medida de qualidade do componente. As medidas são: linhas de código, <i>throughput</i> , detecção de erros, tempo de alocação dos recursos, priorização de eventos e a utilização nos Sistemas RFID.
<b>5.4 Segurança</b>
Define as estratégias contra códigos maliciosos e hackers. Até mesmo em nível físico, ou seja, nas tags e leitores RFID.
<b>6. Informações Diversas</b>
Descreve informações gerais sobre o componente e que ajudam a compreensão do <i>template</i> para documentação do componente.
<b>6.1 Guia de Instalação</b>
Define as operação que devem ser executadas antes do componente ser usado.
<b>6.2 Suporte</b>
Define o contato das pessoas que podem ajudar na instalação e no suporte técnico dos componentes.

### 7.3 Considerações Finais

A Implementação do Domínio tem como seu principal objetivo apresentar meios para a implementação e documentação dos componentes desenvolvidos. Apesar das poucas bibliografias o capítulo se baseou em métodos consolidados para o estabelecimento da última fase do processo de Engenharia do Domínio. O próximo capítulo apresenta as considerações finais deste trabalho e apresenta os trabalho futuros proporcionados no estabelecimento do processo.

## 8. Estudo de Caso

A abordagem de engenharia de domínio proposta nos capítulos 4, 5, 6 e 7 está dividida em três fases: Análise do Domínio, Projeto do Domínio e Implementação do Domínio. Neste capítulo será mostrada a aplicação desta abordagem em um estudo de caso da cadeia de suprimentos para uma rede de supermercados, conforme [CAMPOS; ZORZO 2008].

### 8.1 Análise do Domínio

Vista como uma importante fase nos processos de engenharia de domínio, a análise do domínio descrita no capítulo 5 e em [CAMPOS; ZORZO 2007] consiste em quatro passos: Planejamento do domínio, Requisitos do domínio, Modelagem dos requisitos e Documentação dos requisitos. As próximas seções apresentarão, em detalhes, cada passo da análise do domínio aplicado-os em uma cadeia de suprimentos de uma rede de supermercado.

#### 8.1.1 Planejamento do Domínio

A primeira atividade corresponde ao planejamento do domínio. Essa fase é baseada em três sub-atividades: Descrição do domínio, Identificação dos *stakeholders* e Escopo do domínio.

A Descrição do domínio visa descrever qual cadeia de suprimentos será aplicada a análise do domínio. Neste caso, a cadeia de suprimentos analisada será a de uma rede de supermercados. Os domínios identificados são: (i) fornecedores de matéria-prima, (ii) indústria, (iii) centros de distribuição, (iv) supermercados e (v) consumidores. Todos

domínios da cadeia de suprimentos foram descritos em função de quatro aspectos, veja a Tabela 7:

**Tabela 7:** Domínios da Cadeia de Suprimentos de uma rede de supermercados

<b>Domínio</b>	<b>Atividade</b>	<b>Entrada</b>	<b>Saída</b>	<b>Tecnologia</b>
Fornecedores de Matéria-prima	Extrair a matéria-prima da natureza e fornecê-la às indústrias.	Fornecedores de Matéria-prima	Indústria	Recebimento da matéria-prima, Processamento, Empacotamento e Transporte
Indústria	Transformar matéria-prima em mercadorias para os consumidores	Fornecedores de Matéria-prima e Centros de Distribuição	Centros de Distribuição	Recebimento da matéria-prima, Processamento, Check-points, Empacotamento e Transporte
Centros de Distribuição	Satisfazer a demanda criada por respectivos varejistas	Indústria e Supermercados	Supermercados	Recebimento das mercadorias, Armazenamento, Localização dos produtos e Transporte
Supermercados	Monitorar o estoque e automaticamente iniciar a encomenda	Centrais de Distribuição e Consumidores	Consumidores	Recebimento dos produtos, Prateleiras inteligentes e Estoque
Consumidores	Aquisição de produtos e consulta de informações destes produtos	Supermercados	-	Não se aplica

Apesar da troca de informações lateral entre os domínios da cadeia de suprimentos, os Sistemas RFID, através do Serviço de Informação EPC, possibilitam a troca de informações entre todos os domínios envolvidos. Como mostra [BROCK; CUMMINS 2003] os dados armazenados em um EPC-IS ficam disponíveis para toda organização ou pessoa que se autenticar na Rede EPCglobal.

A partir dos dados contidos na Tabela 7 é possível identificar situações similares em que os Sistemas RFID são usados, por exemplo, no recebimento de mercadorias, processamento dos produtos. São situações como essas que aumentam o reuso do software produzido ao final do processo. É possível também verificar as relações entre os domínios, neste caso, os domínios foram ordenados em uma seqüência onde as relações são

estabelecidas um após o outro. Por fim, observa-se a atividade exercida por cada domínio na cadeia de suprimentos que contribuirá na delimitação do escopo da Cadeia de Suprimentos.

A segunda sub-atividade é a Identificação dos *Stakeholders*, nela são identificadas as pessoas, ou alguém, que tenha um interesse definido no resultado do projeto. Os *stakeholders* identificados para esse estudo de caso foram: (i) Especialista da tecnologia RFID, (ii) Analista do domínio, (iii) Gerente do domínio, (iv) Desenvolvedor e (v) Especialista de mercado.

O Especialista da tecnologia RFID é um *stakeholder* com interesse identificado em todos os domínios da Cadeia de Suprimentos. Por outro lado, o Analista do domínio, o Gerente do domínio e o Desenvolvedor são *stakeholders* presentes, sobretudo, nos domínios das Indústrias, dos Centros de Distribuição e dos Supermercados. Isso se explica porque a maior parte das informações que transitam em uma Cadeia de Suprimentos são provenientes desses domínios. Para o domínio dos Consumidores não foram identificados *stakeholders* uma vez que os consumidores não participam do processo ativamente da Cadeia de Suprimentos.

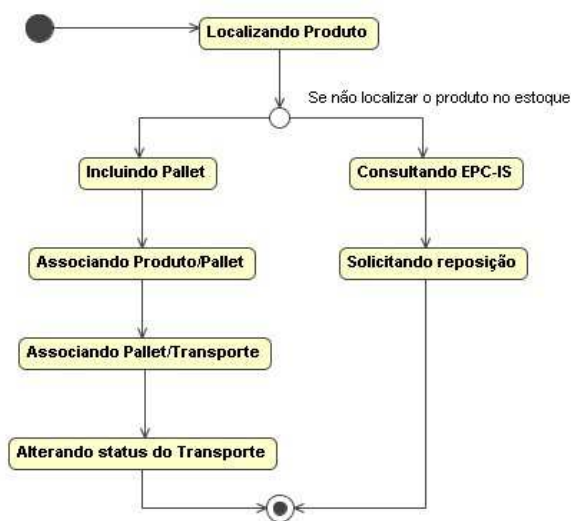
A terceira e última sub-atividade é o Escopo do Domínio, que tem como objetivo identificar a relevância dos Sistemas RFID nos domínios da cadeia de suprimentos, a relação entre os domínios para, finalmente, eliminar os domínios que não contém informações coerentes com essa proposta.

Uma vez que todos os domínios estabelecem relação com pelo menos outro domínio, deve-se aplicar métodos mais detalhados para a eliminação de domínios fora do escopo. O primeiro deles é analisar quantos sistemas distintos estão no domínio, método conhecido como análise horizontal do escopo. Nesta análise fica fácil perceber que o domínio dos Consumidores não comporta qualquer tipo de sistema semelhante aos desenvolvidos na Cadeia de Suprimentos e, portanto, está fora das próximas etapas da Análise do Domínio. O segundo método consiste em eliminar as partes dos sistemas que não correspondem ao

domínio. Neste estudo de caso, os módulos de localização das unidades de transporte no globo terrestre, contratação de pessoal, gerenciamento de escalas de trabalho, etc não serão partes dos sistemas a serem modelados.

### 8.1.2 Requisitos do Domínio

O objetivo desta fase é identificar as *features* do domínio e dos sistemas inseridos na cadeia de suprimentos. A identificação dessas *features* deve ser feita pelos *stakeholders* através de cenários que descrevem como o sistema é usado na prática. Uma forma de representação de cenários é através do diagrama de estados definido pela UML 2.0. A Figura 13 representa um cenário bastante comum na cadeia de suprimentos que é o embarque de mercadorias em unidades de transporte.



**Figura 13:** Diagrama de Estados – Embarque de mercadorias

Os cenários são descritos pelos *stakeholders* a partir de um estado inicial do sistema, os eventos externos, os eventos alternativos e o estado final do sistema ao término do cenário. O cenário que levou a obtenção do diagrama mostrado na Figura 13 foi o seguinte: para embarcar um produto, inicialmente, o sistema deve localizar este produto no domínio e verificar se existe a quantidade solicitada para embarque. Caso não exista a quantidade

exigida para o embarque, o sistema realiza uma consulta no EPC-IS a fim de encomendar a reposição do produto.

Caso exista produto suficiente em estoque, o administrador do sistema deve cadastrar as *pallets* que serão embarcadas na unidade de transporte, determinar o destino para cada uma delas e associar os produtos às *pallets* cadastradas. Existe ainda a associação das *pallets* às unidades de transporte a fim de facilitar a localização dos produtos posteriormente. Por fim, os leitores RFID posicionados nas saídas de embarque farão a leitura da mercadoria autorizando a alteração do status da unidade de transporte para “em trânsito”.

A partir desses cenários e de outros cenários propostos por cada *stakeholder*, chegamos a todas as *features* do sistema, conforme mostra a Tabela 8.

**Tabela 8:** Níveis das *features*.

Nível 1	Nível 2	Nível 3	Nível 4
F1. Produto	F1.1 Número EPC		
	F1.2 Localização	F1.2.1 Estoque	F1.2.1.1 Pallet
			F1.2.1.2 Prateleira
			F1.2.1.3 Container
		F1.2.2 Transporte	F1.2.2.1 Caminhão
			F1.2.2.2 Navio
			F1.2.2.3 Avião
F1.2.2.4 Trem			
F1.3 Transação			
F2. EPC-IS	F2.1 Inventário		
	F2.2 Registros	F2.2.1 Autorização	
		F2.2.2 Controle de Acesso	
F2.3 Fornecedor			
F3. NOS	F3.1 Consulta	F3.1.2 Fonte Confiável	F3.1.2.1 Servidores EPC-IS
			F3.1.2.2 Página Web
		F3.1.2.3 Interfaces Web Service	
		F3.1.2 Fonte não-confiável	

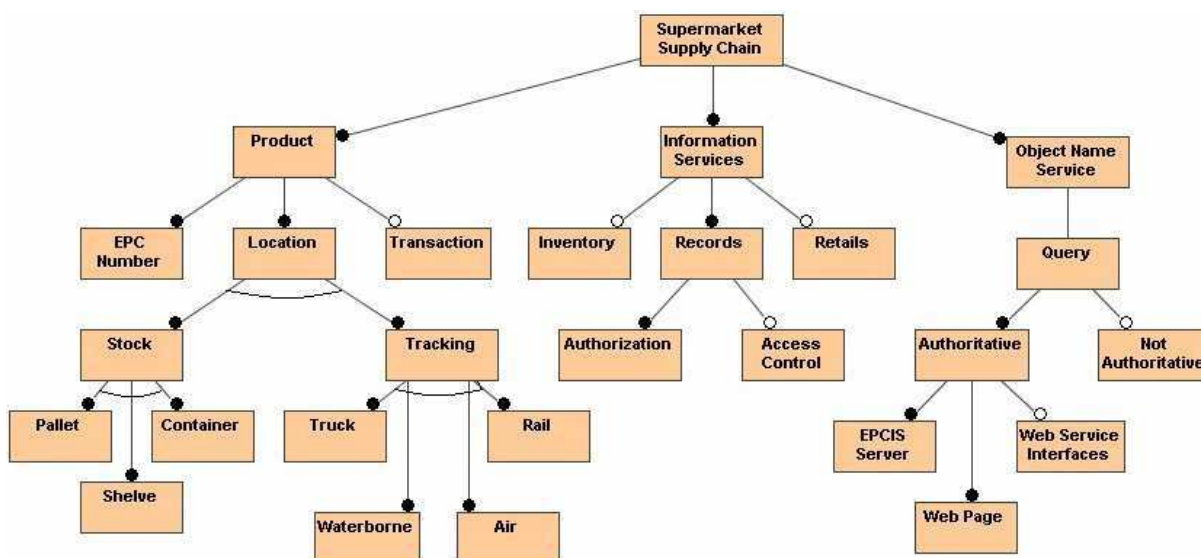
### 8.1.3 Modelagem dos Requisitos

Nessa atividade o domínio será modelado em função das *features* identificadas no levantamento de requisitos. Nesse contexto, a modelagem consiste em três sub-atividades: Requisitos Comuns, Requisitos variáveis e Modelagem. Para identificar os requisitos comuns e variáveis do domínio foi utilizada a matriz de requisitos do domínio baseada em prioridades. O objetivo é atribuir uma prioridade para cada *feature* de acordo com o *stakeholder* envolvido, como mostra a Tabela 9.

**Tabela 9:** Matriz de requisitos do domínio baseada em prioridade

Feature/Domínio	Fornecedor	Indústria	Centro de Distribuição	Supermercados
Product	Pr2	Pr1	Pr1	Pr1
EPC-IS	Pr1	Pr1	Pr1	Pr1
ONS	Pr2	Pr1	Pr1	Pr1
Shelve	X	Pr3	Pr3	Pr1

Após o preenchimento da matriz, o analista do domínio deve definir a prioridade ideal para os requisitos comuns. Nesse caso, a prioridade desejada foi *Pr2* visando a permanência de alguns requisitos variáveis como mostra a Figura 14.



**Figura 14:** Diagrama de *features* para a Cadeia de Suprimentos de uma rede de supermercado



### 8.1.4 Documentação dos requisitos

Na Documentação dos requisitos as *features* são documentadas usando texto em linguagem natural. A Tabela 10 mostra a melhor forma para documentar uma *feature* da Cadeia de Suprimentos.

**Tabela 10:** Documentação da *feature* Produto

<b>Nome da Feature: Produto</b>
<b>Descrição Semântica</b>
O produto são todas as mercadorias, objetos e serviços que podem ser identificados na Cadeia de Suprimentos. Esses produtos possuem uma tag RFID que armazena um identificador único, conhecido como Código Eletrônico do Produto.
<b>Razão</b>
A razão de sua existência no modelo de <i>features</i> é porque o produto ser o maior proposta da existência de qualquer Cadeia de Suprimentos. O produto não é uma <i>feature</i> específica dos Sistemas RFID.
<b>Stakeholders e programas clientes</b>
Os <i>stakeholders</i> que relacionados à essa <i>feature</i> são: Gerente do Domínio e o Especialista na tecnologia RFID.
<b>Aplicações exemplos</b>
As aplicações podem utilizar a <i>feature</i> produto para descobrir relações de negócio, a localização de um objeto, bem como a origem do produto.
<b>Restrições</b>
A <i>feature</i> não apresenta restrições e pode ser usada em qualquer domínio

<b>Ponto de Variação</b>
A <i>features</i> é aberta para novas <i>features</i> . Por exemplo, o número EPC que está associado a ela.
<b>Prioridades</b>
A <i>features</i> apresenta um alto grau de importância para a Cadeia de Suprimentos.

## 8.2 Projeto do Domínio

O Projeto do Domínio é a segunda fase neste Processo de Engenharia de Domínio para o desenvolvimento de aplicações em Cadeias de Suprimentos que utilizam RFID. O objetivo principal dessa fase é definir visões da arquitetura referência do domínio a partir dos artefatos gerados na fase de Análise do domínio. A seguir, é apresentada a aplicação dos passos do projeto do domínio na cadeia de suprimentos de uma rede de supermercado.

### 8.2.1 Mapeamento

Os padrões de projeto (*Design Patterns*) apresentados na seção 5.1 foram escolhidos criteriosamente a partir dos vinte e três métodos apresentados por [GAMMA 1995]. Nessa seção serão apresentadas as formas de utilização dos padrões de projeto definidos para cada tipo de *feature*.

Para as *features* alternativas que podem ser diretamente mapeadas em uma simples classe, a abordagem sugere a utilização do padrão de projeto Singleton devido sua orientação de instanciar um objeto específico dependendo da *feature* que é usada na aplicação, através de herança simples.

Quando a *feature* não pode ser implementada por uma única classe recomenda-se o uso do padrão Factory Method ou Abstract Factory, os quais, permitem através de herança da matriz, desenvolver mais que uma classe para a mesma *feature*. Apesar da semelhança desses dois padrões o Abstract Factory separa classe para instanciar o objeto, por outro lado, a instanciação do objeto no Factory Method é incorporada em outras classes da arquitetura.

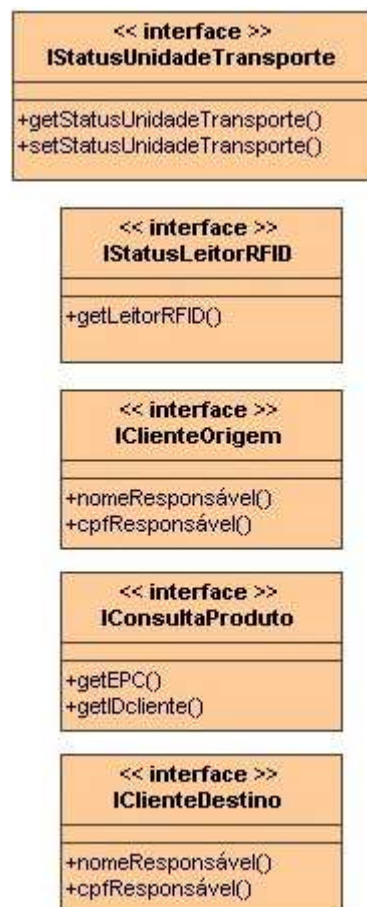
Outra diferença entre os padrões é que o Factory Method impede que herança seja usada em uma classe que encapsula-o, enquanto que o Abstract Factory não impõe essa restrição. A saída para alguns conflitos causados pelo Factory Method e o Abstract Factory é o padrão Singleton. Ao utilizar esse método que todas as classes que são relacionadas à *feature* são unidas através de uma única classe.

Como a *feature* opcional não necessariamente irá existir no domínio, o padrão Decorator pode ser usado quando diferentes *features* tem funcionalidades que são complementares, de modo que uma *feature* deva executar após a outra. Decorator é um padrão estrutural, mas pode ser usado em casos onde a interação entre várias *features* é complexa, principalmente, em casos onde a estrutura de funcionalidades adicionais pode ser definida durante o tempo de projeto. Para as *features* opcionais onde o mapeamento de uma *feature* em uma classe é direto, sugere-se o uso do padrão Prototype. A idéia é evitar que na especificação de uma classe, objetos executarem tarefas particulares sem relação com a classe, o que dificulta mudanças futuras.

Para as *or-features* os padrões devem assegurar que no mínimo uma *feature* esteja presente. Dessa forma sugere-se a utilização do padrão Builder para gerenciar e determinar quais *features* estarão presentes. O padrão Bridge será usado quando uma simples classe é necessária e a assinatura dos métodos da implementação for idêntica à especificação.

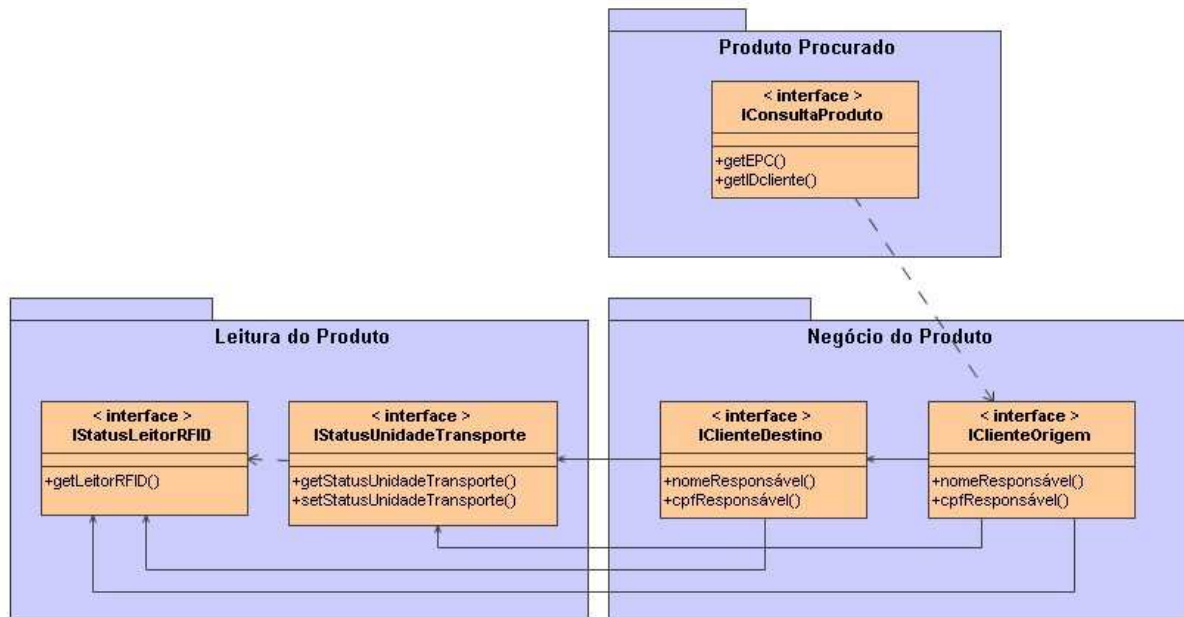
### 8.2.2 Projeto dos Componentes

O passo Projeto dos Componentes define duas atividades: a identificação e a especificação das interfaces. A interface é “uma coleção de operações que são usadas para especificar um serviço de uma classe ou componente” [SZYPERSKI 2002]. Dessa forma, as interfaces definidas para o componente de localização dos produtos na Cadeia de Suprimentos são mostradas na Figura 15:



**Figura 15:** Interfaces do Componente Localização

Após a especificação das interfaces, as informações adicionais de especificação que o desenvolvedor do componente necessita ter consciência são as dependências de um componente em relação às outras interfaces. A especificação das interfaces providas e requeridas pode ser visualizada na Figura 16:



**Figura 16:** Pacote de Especificação das Interfaces

Percebe-se que a interface responsável pela consulta do produto mantém uma dependência com o cliente de origem do produto. Dessa forma, é possível rastrear o produto a partir de sua origem e chegar ao destino. Caso o produto esteja sendo transportado, a leitura do produto ocorrerá dentro da unidade de transporte ou diretamente na prateleira através do leitor RFID.

### 8.2.3 Visões da Arquitetura

Entre as quatro visões definidas no Projeto do Domínio, optou-se pela visão deployment para representar o estudo de caso. É válido ressaltar a importância em descrever outras visões da arquitetura do sistema para que o arquiteto do sistema veja o mesmo sistema por diferentes aspectos. A Visão Deployment pretendida neste trabalho visa descrever a estrutura de hardwares na qual o sistema está sendo executado. A Figura 17 mostra a visão deployment da cadeia de suprimentos de uma rede de supermercado com base na arquitetura da Rede EPCglobal.

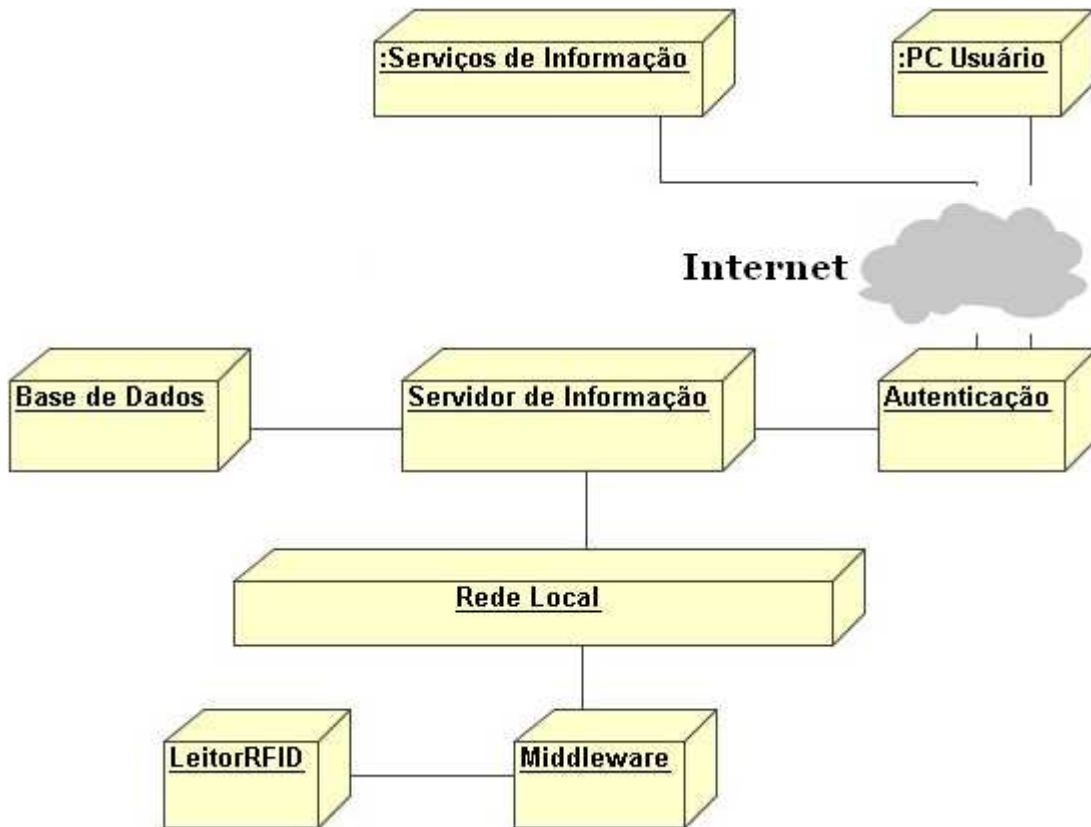


Figura 17: Visão Deployment da Cadeia de Suprimentos

### 8.2.4 Documentação da Arquitetura

Após a obtenção das visões e das interfaces dos componentes, o último passo é documentação da arquitetura como mostrada na Tabela 11.

Tabela 11: Documentação da Arquitetura do Estudo de Caso

Documento da Arquitetura
<b>1. Guidelines de Utilização</b>
Este documento contém seis seções (Guidelines de Utilização, Informações do Projeto, Documentação das Views, Relação entre Análise e Projeto e Glossário), sendo que a melhor forma de utilização destas seções é identificar sua finalidade e preenchê-las sequencialmente.
<b>2. Informações do Projeto</b>

O projeto está em sua primeira versão e foi desenvolvido por Leonardo Barreto Campos e Sérgio Donizetti Zorzo. O objetivo do projeto é desenvolver aplicações com reuso para Cadeias de Suprimentos que utilizam RFID. Os documentos que podem auxiliar a melhor compreensão dos Sistemas RFID podem ser encontrados no site da EPCglobal (<http://www.epcglobalinc.org/home>).

### **3. Informações do Domínio**

O domínio a ser projetado são as Cadeias de Suprimento baseadas na RFID. Os participantes deste domínio são equipados com:

- Unidade de transporte capaz de informar ao sistema sua posição atual através da tecnologia *Global Positioning System – GPS* e receber em seu container as pallets contendo os produtos que serão transportados;
- Leitor RFID nas saídas de embarque de mercadorias capaz de notificar ao sistema o decréscimo ou acréscimo de produtos no estoque e leitores nas baias de estocagem para informar ao sistema as tags que estão em sua área de atuação;
- Sensor nas prateleiras capaz de perceber acréscimos e decréscimos de produtos além de notificar ao estoque a necessidade de reposição de produtos nessas prateleiras;
- Carrinho de compras equipado com um sensor capaz de identificar os produtos inseridos nesse carrinho e um display para interação com o cliente (logon, logout, rota de produtos, etc);

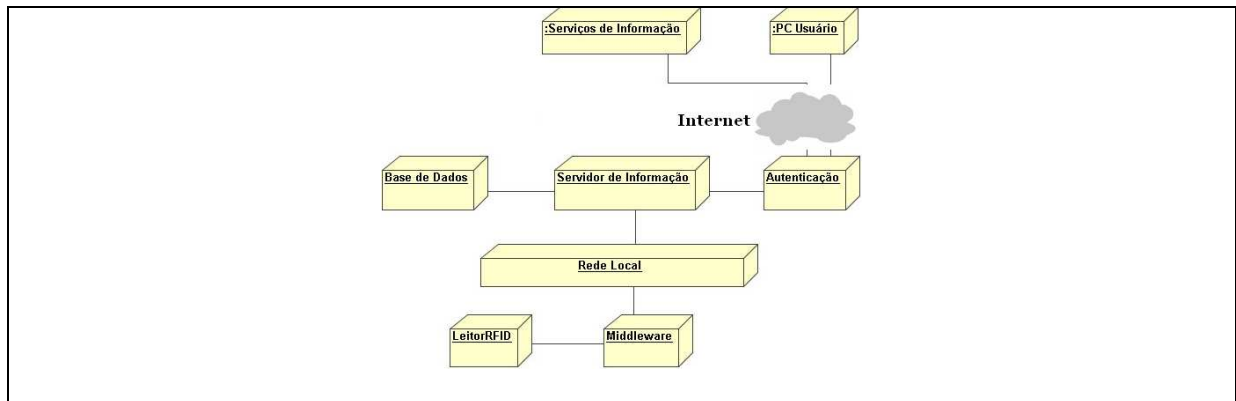
### **4. Documentação das View:**

#### **4.1 Nome da View**

Visão *Deployment*. Sua finalidade é descrever a estrutura de hardwares na qual o sistema está sendo executado.

#### **4.2 Representação Gráfica**

A representação gráfica da *view* é mostrada a seguir:.



### 4.3 Descrição dos Elementos

A visão *Deployment* da arquitetura contém os elementos de Sistema de Informação dentro e fora do domínio para a troca de informações relacionadas aos produtos, Autenticação para autorizar o acesso às bases internas, Base de Dados para o armazenamento dos dados referentes aos produtos, Rede EPCglobal, Middleware para o tratamento dos número EPCs lidos, Leitores RFID e o PC do Usuário para consulta de um número EPC.

### 4.4 Views Relacionadas

A visão *Deployment* relaciona com a visão de módulos, especialmente no elemento “Base de Dados”.

### 4.5 Outras Informações

A visão segue o padrão definido pela UML 2.0, entretanto, pode ser melhorada com mais funcionalidades trazidas pela UML.

## 5. Relações de Análise e Projeto

Os requisitos descritos na fase de análise foram contemplados nos módulos da visão deployment.

## 6. Glossário

A visão não apresentou acrônimos.



### 8.3 Implementação do Domínio

Na implementação do Domínio os objetivos são fornecer o projeto detalhado, implementar e a documentar os *assets* reusáveis, baseado na arquitetura descrita na fase do Projeto do Domínio. Vejamos um trecho da implementação de um componente e sua documentação nas seções a seguir:

#### 8.3.1 Implementação dos Componentes

Para executar a fase de implementação dos componentes o engenheiro do domínio deve ter os artefatos gerados na fase de Análise do Domínio (requisitos, modelo de *features* e cenários) e na fase de Projeto do Domínio (arquitetura do domínio e especificação dos componentes). A partir daí são executadas quatro tarefas: (i) descrição do componente, (ii) especificação das interfaces, (iii) implementação dos serviços e (iv) instalação do componente.

A Figura 18 mostra a implementação do componente Serviço de Localização em UML 2.0, usado para localizar produtos na cadeia de suprimentos.

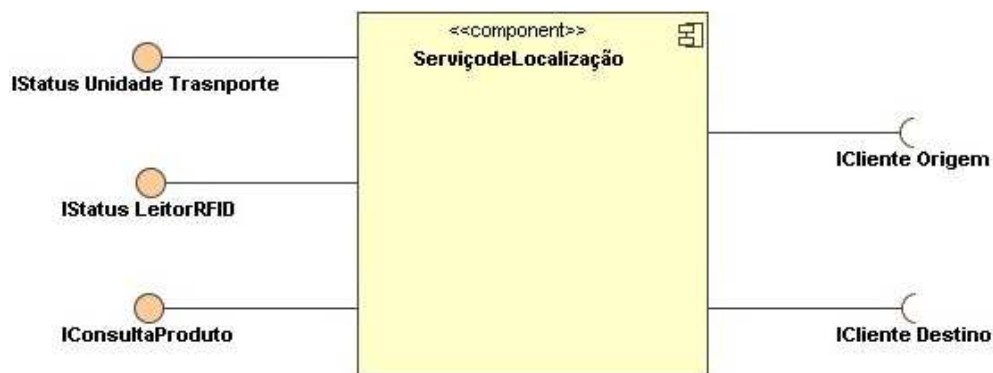


Figura 18: Componente Serviço de Localização

### 8.3.2 Documentação dos Componentes

De acordo com o *template* para documentação de componentes, tem-se a aplicação de parte dos passos no componente de Serviço de Localização, veja a Tabela 12 a seguir.

**Tabela 12:** Documentação do Componente Serviço de Localização

<b>Documento do Componente</b>
<b>1. Informações Básicas:</b>
<b>1.1 Nome</b>
Serviço de Localização
<b>1.2 Tipo</b>
Componente de Serviço
<b>1.3 Finalidade</b>
Este componente fornece a localização de um produto na cadeia de suprimentos. Ele recebe um número ECP e retorna a localização atual do produto e a relação de negócio que envolve o produto (origem e destino).
<b>1.4 História</b>
Versão: 1.0
Data: Maio de 2007
Desenvolvedor: Leonardo Barreto Campos, UFSCar
<b>2. Informações das Interfaces</b>
<b>2.1 Interfaces Requeridas</b>
Nome: IClienteOrigem
Tipo: Interface básica de troca de mensagem entre dois serviços.
Descrição: Esta interface fornece comunicação com a base de dados dos fornecedores.

Comportamento: Interface estática

Finalidade: Localizar os dados da Origem do produto.

## **2.2 Interfaces Providas**

Nome: ConsultaProduto

Tipo: Interface básica de troca de mensagem entre o Serviço de Informação EPC.

Descrição: Esta interface prover a localização do produto na Cadeia de Suprimentos.

Comportamento: Interface estática

Finalidade: Revelar a localização relativa de um produto. A localização relativa é sempre dada em relação a algum referencial, por exemplo, o produto encontra-se em um pallet que está dentro de uma unidade de transporte que está na Rodovia BR-116 Km 1160.

## **8.4 Considerações Finais**

O estudo de caso mostrou a aplicação das fases de Análise, Projeto e Implementação descritas nos capítulos 4, 5, 6 e 7. O domínio escolhido, a Cadeia de Suprimentos de uma Rede de Supermercados, contemplou boa parte das possíveis aplicações dos Sistemas RFID nas Cadeias de Suprimentos. Dessa forma, foi possível mostrar a aplicação de todas as fases do processo de Engenharia de Domínio sem excluir fases. O próximo capítulo mostrará as considerações finais deste trabalho e identificará as linhas de pesquisa deixadas em aberto na descrição do processo.

## 9. Conclusão

Os processos de reuso de software descritos na literatura seguem em duas direções: Engenharia de Domínio e as Linhas de Produto de Software. Como pôde ser observado no Capítulo 3, os processos de reuso, ao longo dos anos, buscaram solucionar falhas deixadas por processos anteriores. Dessa forma, o presente trabalho procurou atacar brechas dos diversos processos de reuso de software direcionando para um domínio específico, as aplicações em Cadeias de Suprimentos baseadas na RFID.

O trabalho deixa como principais contribuições (i) a descrição de três fases de um processo de Engenharia de Domínio focadas nos Sistemas de Identificação por Radiofrequência, (ii) *templates* para a documentação das *features* identificadas na Análise do Domínio, documentação da Arquitetura referência obtida no Projeto do Domínio e documentação dos componentes desenvolvidos na Implementação do Domínio e (iii) aplicação das fases definidas no Processo em um estudo de caso para a melhor compreensão da abordagem.

### 9.1 Trabalhos Relacionados

Os trabalhos relacionados identificados na literatura são apresentados no Capítulo 3. São onze processos de reuso de software que contribuíram na especificação das três fases do processo descritos neste trabalho. A principal contribuição ao revisar a literatura dos trabalhos relacionados foi a possibilidade de explorar falhas deixadas pelos processos de reuso e reduzir as brechas, bem como modificar pontos positivos de cada processo validados pela academia após anos de utilização.

## 9.2 Trabalhos Futuros

Ao descrever um processo completo de Engenharia de Domínio o trabalho abre algumas linhas de pesquisa a serem exploradas em trabalhos futuros. Entre ele destacam-se: (i) a Implementação de um framework e aplicação do trabalho em um cenário real para o aprimoramento dos passos e inclusão de atividades relacionadas à gerência do negócio, (ii) Descrição de um processo de reuso de software para qualquer aplicação baseada na tecnologia RFID, não somente aplicações da cadeia de suprimentos, (iii) Na Análise do Domínio, a identificação de variações no sistema pode ser mais bem explorada, uma vez que a permanência das variações torna o sistema mais flexível e mais fácil de ser customizado, (iv) Nas fases de Projeto e Implementação do Domínio, a abordagem deixa fases sem aplicação direta nos Sistemas RFID.

## REFERÊNCIAS BIBLIOGRÁFIAS

[ALEXANDER et al. 1977] ALEXANDER C.; ISHIKAWA S.; SILVERSTEIN M.; JACOBSON M.; KING I. F.; AANGEL S. **A Pattern Language: Towns, Buildings, Construction**, Oxford University Press, 1977, p. 1216.

[ALMEIDA et al. 2004] ALMEIDA E. S.; ALVARO A.; LUCRÉDIO D.; GARCIA V. C.; MEIRA S. R. L. **RiSE Project: Towards a Robust Framework for Software Reuse**. In: IEEE International Conference on Information Reuse and Integration, 2004, Las Vegas. **Proceedings...** Las Vegas: 2004. p. 48-53.

[ALMEIDA et al. 2005] ALMEIDA E. S.; ALVARO A.; LUCRÉDIO D.; GARCIA V. C.; MEIRA S. R. L. **A Survey on Software Reuse Processes**. In: IEEE References 160 Internacional Conference on Information Reuse and Integration, 2005, Las Vegas. **Proceedings...** Las Vegas: 2005. p.66-71.

[ALMEIDA 2007] ALMEIDA, E. S. **RiDE: The RiSE Process for Domain Engineering**. 2007. Tese (Doutorado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2007.

[AMERICA et al. 2000] AMERICA P.; OBBINK H.; OMMERING R. V.; LINDEN F. V. D. **CoPAM: A Component-Oriented Platform Architecting Method Family for Product Family Engineering**. In: First Software Product Line Conference, 2000, Denver. **Proceedings...** Denver: 2000. p.15.

[ARMENIO 2007] ARMENIO, F et al., **The EPCglobal Architecture Framework**, EPCglobal Final Version, pp. 7, Setembro 2007.

[ATKINSON; BAYER; MUTHING 2000] ATKINSON C.; BAYER J.; MUTHING D. **Component-Based Product Line Development: The KobrA Approach**. In: First Software Product Line Conference, 2000, Denver. **Proceedings...** Denver: 2000. p. 19.

[AUTO-ID CENTER 2001] **Technology Guide**. Massachusetts Institute of Technology, Cambridge, 2001. Disponível em: <[http://archive.epcglobalinc.org/new\\_media/brochures/Technology\\_Guide.pdf](http://archive.epcglobalinc.org/new_media/brochures/Technology_Guide.pdf)>. Acesso em: 07 de fev. 2006.

[AUTO-ID LABS 2006] **Welcome to Auto-ID Labs Global Site!**. Disponível em: <<http://www.autoidlabs.org/aboutthelabs.html>>. Acesso em: 07 de fev. 2006.

[BACHELDOR 2008] BACHELDOR, B., **U.N.'s Universal Postal Union Gears Up for Large RFID Pilot**, RFID Journal. Disponível em <<http://www.rfidjournal.com/article/print/4504>>. Acesso em: 09 de set. 2010.

[BASILI; BRIAND; MELO 1996] BASILI V. R.; BRIAND L. C.; MELO W. L. **How Reuse Influences Productivity in Object-Oriented Systems**. Communications of the ACM, Vol.39, N. 10, p. 104-116, Outubro, 1996.

[BASS; CLEMENTS; KAZMAN 2003] BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**, Boston: Addison Wesley, 2003. p. 21-24.

[BAUER 1993] BAUER, D. **A Reusable Parts Center**, IBM Systems Journal, Vol. 32, No. 04, pp. 620-624, Setembro, 1993.

[BAYER et al. 1999] BAYER J.; FLEGE O.; KNAUBER P.; LAQUA R.; MUTHING D.; SCHMID K.; WIDEN T.; DEBAUD J. **PuLSE: A Methodology to Develop Software Product Lines**, In: Symposium on Software Reusability, 1999, Los Angeles. **Proceedings...** Los Angeles: 1999, p. 122-131.

[BOEHM 1988] BOEHM, B. W. **A Spiral Model of Software Development and Enhancement**, IEEE Computer, Vol. 21, No. 05, pp. 61-72, Maio, 1988.

[BOOCH; RUMBAUGH; JACOBSON 2005] BOOCH, G.; RUMBAUGH J.; JACOBSON I. **The Unified Modeling Language User Guide**, 2. ed. Boston: Addison-Wesley, 2005.

[BOSCH 2000] BOSCH J. **Design and Use of Software Architecture**, Boston: Addison-Wesley, 2000. p. 354.

[BROCK 2001b] BROCK, L. D. **The Eletronic Product Code – A Naming Schema for Pysical Objects**. Disponível em: <<http://www2.hkana.org/files/epcGlobal/paper/MIT-AUTOID-WH-002.pdf>>. Acesso em: 07 de fev. 2006.

[BROCK 2001c] BROCK, L. D. **The Compact Eletronic Product Code – a 64-bit Representation of the Eletronic Product Code**. Disponível em: <<http://www2.hkana.org/files/epcGlobal/paper/MIT-AUTOID-WH-008.pdf>>. Acesso em: 07 de fev. 2006.



[BROCK; CUMMINS 2003] BROCK, D.; CUMMINS, C. **EPC Tag Data Specification**. Disponível em <<http://www.autoidlabs.org/whitepapers/mit-autoid-wh025.pdf>>. Acesso em: 07 de fev. 2006.

[BUSCHMANN et al. 1996] BUSCHMANN F.; MEUNIER R.; ROHNERT H.; SOMMERLAD P.; STAL M. **Pattern-Oriented Software Architecture. A System of Patterns**. Wiley, Chichester, 1996.

[CAMPOS; ZORZO 2007] CAMPOS, L. B.; ZORZO, S. D. **A Domain Analysis Approach for Engineering RFID Systems in Supply Chain Management**, In: IEEE International Conference on System of Systems Engineering, 2007, San Antonio. **Proceedings...** San Antonio: 2007. p.165-171.

[CAMPOS et al. 2008] CAMPOS, L. B.; ALMEIDA, E. S.; ZORZO, S. D.; MEIRA, S. R. L. **A Domain Engineering Process for RFID Systems Development in Supply Chain**. In: I-Tech Education. (Org.). Supply Chain The Way to Flat Organization. 1ª ed. Vienna, Austria: In-Teh, 2008, p. 103-126.

[CAMPOS; ZORZO 2008] CAMPOS, L. B.; ZORZO, S. D. **Domain Analysis: A Case Study of Engineering RFID Systems in Supply Chain**. In: World Congress in Computer Science Computer Engineering and Applied Computing, 2008, Las Vegas. World Congress in Computer Science Computer Engineering and Applied Computing, 2008.

[CANGIALOSI; MONALY; YANG 2007] CANGIALOSI, A.; MONALY J. E.; YANG S. C., **Leveraging RFID in hospitals: Patient life cycle and mobility perspectives**, IEEE Communication Magazine, Vol 45, No. 9, Setembro 2007.

[CLEMENTS; NORTHROP 2001] CLEMENTS P.; NORTHROP L. **Software Product Lines: Practices and Patterns**. Addison-Wesley, p. 608, 2001.

[CHEESMAN; DANIELS 2000] CHEESMAN J.; DANIELS J. **UML Component A Simple Process for Specifying Component-Based Software**. Boston: Addison-Wesley, 2000. p. 208.

[CSX WORLD TERMINALS 2001] **CSX World Terminals**. Disponível em: <<http://www.csxworldterminals.com/Resources/Glossary.asp?s=s>>. Acesso em: 07 de fev. 2006.

[CZARNECKI; EISENECKER 2000] CZARNECKI K; EISENECKER U. W. **Generative Programming: Methods, Tools, and Applications**, Boston: Addison-Wesley, 2000. p. 832.

[D'SOUZA; WILLS 1998] D'SOUZA D. F.; WILLS A. C. **Objects, Components and Frameworks with UML: The Catalysis Approach**. Boston: Addison-Wesley, 1998.

[EANUCC 2006] **General Specifications v7.0**. Disponível em: <[http://www.gs1uk.org/EANUCC/WORD\\_Files/word.html](http://www.gs1uk.org/EANUCC/WORD_Files/word.html)>. Acesso em: 07 de fev. 2006.

[ENDRES 1993] ENDRES, A. **Lessons Learned in an Industrial Software Lab**, IEEE Software, Vol. 10, No. 05, pp. 58-61, Setembro, 1993.

[ENGELS 2003a] ENGELS, D. **The use of the Eletronic Product Code**. Disponível em: <<http://www.autoidlabs.org/whitepapers/mit-autoid-tr009.pdf>>. Acesso em: 07 de fev. de 2006.

[ENGELS 2003b] ENGELS, D. **EPC-256-bit Eletronic Product Code Representation**. Disponível em: <<http://www.autoidlabs.org/whitepapers/mit-autoid-tr010.pdf>>. Acesso em: 07 de fev. 2006.

[EPCGLOBAL 2005] **Object Naming Service (ONS) Version 1.0**. Disponível em: <[http://www.epcglobalinc.org/standards\\_technology/EPCglobal\\_Object\\_Naming\\_Service\\_ONS\\_v112-2005.pdf](http://www.epcglobalinc.org/standards_technology/EPCglobal_Object_Naming_Service_ONS_v112-2005.pdf)>. Acesso em: 08 de fev. 2006.

[EZTRAN et al. 2002] EZTRAN, M.; MORISIO, M; TULLY, C. **Practical Software Reuse**, pp. 374, Springer

[FLEURY 2000] FLEURY, P. F. **Conceitos de logística integrada e supply chain management**, São Paulo, 2000, p. 27-55.

[FRAKES; ISODA 1994] FRAKES, W. B.; ISODA, S. **Success Factors of Systematic Software Reuse**, IEEE Software, Vol. 12, No. 01, pp. 15-19, Setembro, 1994.

[FRAKES; KANG 2005] FRAKES W. B.; KANG K. **Software Reuse Research: Status and Future**. IEEE Transactions on Software Engineering, Vol. 31, n. 7, p. 529, Julho, 2005.

[GALLEN 2009] GALLEN, C. **Total RFID Revenue to Exceed \$5.6 Billion in 2009, According to ABI Research**, ABIREsearch, Disponível em: <<http://www.abiresearch.com/press/1395>>. Acesso em 09 de set. 2010.

[GAMMA et al. 1995] GAMMA E.; HELM R.; JOHNSON R.; VLISSIDES J. **Design Patterns: Elements of Reusable Object-Oriented Software**, Boston: Addison-Wesley, 1995. p. 395.

[GOMAA 2005] GOMAA H. **Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures**, Boston: Addison-Wesley, 2005. p. 701.

[GRISS 1994] GRISS, M. L. **Software Reuse Experience at Hewlett-Packard**, 16th IEEE International Conference on Software Engineering, pp. 270, Italia, Maio, 1994, Sorrento

[GUANGJIN 2008] GUANGJIN, L. **RFID Application in 2008 Olympic Beijing**, Presentation Solution, Vol 29, No. 4, 2008.

[HARRISON 2004] HARRISON M. **EPC Information Service (EPC)**. Cambridge Auto-ID Lab, Institute for Manufacturing. Universidade de Cambridge. 2004.

[HODGES; HARRISON 2003] HODGES, S.; HARRISON, M. **Demystifying RFID: Principles & Practicalities**. Disponível em: <<http://www.autoidlabs.org/whitepapers/cam-autoid-wh024.pdf>>. Acesso em: 07 de fev. 2006.

[JACOBSON; GRISS; JONSSON 1997] JACOBSON I.; GRISS M. L.; JONSSON P. **Software Reuse: Architecture, Process and Organization for Business Success**. Boston: Addison-Wesley, 1997. 497 p.

[JOOS 1994] JOOS, R. **Software Reuse at Motorola**, IEEE Software, Vol. 11, No. 05, pp. 42-47, Setembro, 1994

[KABACHINSKI 2005] KABACHINSKI, J. **An Introduction to RFID**. Disponível em: <<http://www.aami.org/resources/hottopics/wireless/JeffKRFID.pdf>>. Acesso em: 08 de fev. 2006.

[KANG et al. 1990] KANG K. C.; COHEN S. G.; HESS J. A.; NOVAK W. E.; PETERSON A. S. **Feature-Oriented Domain Analysis (FODA) Feasibility Study**, Software Engineering Institute (SEI), Technical Report, Novembro, 1990, p.161.

[KANG et al. 1998] KANG K. C.; KIM S.; LEE J.; KIM K.; SHIN E.; HUD M. **FORM: A Feature-Oriented Reuse Method with domain-specific reference architecture**, Annals of Software Engineering Notes, Vol. 05, n. 00, p. 143-168. Janeiro,1998.

[KANG; LEE; DONOHOE 2002] KANG K. C.; LEE J.; DONOHOE P. **Feature-Oriented Product Line Engineering**. IEEE Software, Vol. 19, n 04, p.58-65, Julho/Agosto, 2002.

[KEN et al. 2005] KEN, T. et al. **The EPCglobal Architecture Framework: EPCglobal Final Version of 1 July 2005**. Disponível em:

<[http://www.epcglobalinc.org/standards\\_technology/Final-epcglobal-arch-20050701.pdf](http://www.epcglobalinc.org/standards_technology/Final-epcglobal-arch-20050701.pdf)>.

Acesso em: 07 de fev. 2006.

[KNOSPE et al. 2005] **RFID Security**. Disponível em: <[http://www.inf.fh-bonn-rhein-sieg.de/data/informatik\\_/fb\\_informatik/personen/pohl/Aufsaeetze/Pohl\\_Knospe\\_RFID\\_Security\\_050126.pdf](http://www.inf.fh-bonn-rhein-sieg.de/data/informatik_/fb_informatik/personen/pohl/Aufsaeetze/Pohl_Knospe_RFID_Security_050126.pdf)>. Acesso em: 07 de fev. 2006.

[KOTULA 1998] KOTULA J. **Using Patterns To Create Component Documentation**, IEEE Software, Vol. 15, n 02, p. 84-92, Março/Abril, 1998.

[KRUCHTEN et al. 2006] KRUCHTEN, P.; OBBINK, H. & STAFFORD, J. **The Past, Present, and Future of Software Architecture**, IEEE Software, Vol. 23, No. 02, pp. 22-30, Março/Abril, 2006.

[LAMBERT; STOCK; VANTINE 1998] LAMBERT, D. M.; STOCK, J. R.; VANTINE, J. G. **Administração estratégica da logística**, São Paulo: Vantine Consultoria, 1998, p. 827.

[LANDT 2005] LANDT, J. **The history of RFID**, Potentials, Vol 24, No.4, pp.8-11, Novembro, 2005.

[LARAN RFID 2004] **A Basic introduction to RFID technology and its use in the supply chain**. Disponível em: <<http://www.rfidjournal.com/whitepapers/download/95>>. Acesso em: 08 de fev. 2006.

[LEE 2008] LEE, J., **First RFID Lap Counters, Now Microchipped Olympic Tickets?**, SpeedEndurance. Disponível em: <<http://speedendurance.com/2008/05/31/first-rfid-lap-counters-now-microchipped-olympic-tickets/>>. Acesso em: 09 de set. 2010.

[LOGICALCMG 2004] LogicalCMG. **Waves: RFID Adoption in Returnable Packaging**. Disponível em: <[http://www.logicacmg.com/pdf/RFID\\_study.pdf](http://www.logicacmg.com/pdf/RFID_study.pdf)>. Acesso em: 08 de fev. 2006.

[MORISIO et al. 2002] MORISIO, M; EZRAN, M; TULLY, C. **Success and Failure Factors in Software Reuse**, IEEE Transactions on Software Engineering, Vol. 28, No. 04, pp. 340-357, Abril, 2002

[MOTOROLA 2007] MOTOROLA, **The next-generation warehouse MegatruX improves service and reduces costs with RFID**, RFID world, Disponível em <[http://www.motorola.com/staticfiles/Business/Solutions/Industry%20Solutions/RFID%20Solutions/Documents/Static%20Flies/CS\\_MegatruX\\_1007.pdf](http://www.motorola.com/staticfiles/Business/Solutions/Industry%20Solutions/RFID%20Solutions/Documents/Static%20Flies/CS_MegatruX_1007.pdf)>. Acesso em set. 2010.

[NEIGHBORS 1980] NEIGHBORS J. M. **Software Construction Using Components**, Ph.D. Thesis, University of Califórnia, Irvine, References 175 Department of Information and Computer Science, Abril, 1980, p. 217.

[NORTHROP 2002] NORTHROP, L. **A Framework for Software Product Line Practice. Version 3.0**. Software Engineering Institute, Carnegie Mellon University, Pittsburgh. 2002. Disponível em: <<http://www.sei.cmu.edu/plp/framework.html>>. Acesso em: 22 de jan. 2002.

[OMG 2005] **Unified Modeling Language: Superstructure**. Disponível em: <<http://www.omg.org/docs/formal/05-07-04.pdf>>. Acesso em: 08 de fev. 2006.

[OSTERWEIL 1987] OSTERWEIL, L. **Software Process are Software too, 9th International Conference on Software Engineering**, pp. 02-13, California, USA, Março/Abril, 1987, Monterey

[PAPAZOGLU; GEORGAKOPOULOS 2003] PAPAZOGLU, M. P. & GEORGAKOPOULOS, D. **Service-Oriented Computing**, Communications of the ACM, Vol. 46, No. 10, pp. 25-28, Outubro, 2003.

[PHILIPS 2000] Philips, Disponível em : <<http://www.extra.research.philips.com/natlab/sysarch/GaudiProject>>. Acesso em: 08 de fev. 2006.

[POHL; BOCKLE; LINDEN 2005] POHL K.; BOCKLE G.; van der LINDEN F. **Software Product Line Engineering: Foundations, Principles, and Techniques**. Springer, 2005. p. 467.

[PRESSMAN 2005] PRESSMAN, R. S. **Software Engineering: A Practitioner's Approach**, pp. 880, McGraw-Hill, 2005.

[PRIETO-DIAZ 1990] PRIETO-DIAZ R. **Domain Analysis: An Introduction**, ACM SIGSOFT Software Engineering Notes, Vol. 15, n. 02, p. 47-54, Abril, 1990.



[RINE 1997] RINE, D. C. **Success Factors for Software Reuse that are Applicable Across Domains and Businesses**, ACM Symposium on Applied Computing, pp. 182-186, California, USA, Março, 1997.

[RFID JOURNAL 2005] **What is RFID?** Disponível em: <<http://www.rfidjournal.com/faq/16/49>>. Acesso em: 08 de fev. 2006.

[ROSS 1998] ROSS, D. F. **Competing through Supply Chain Management – Creating Market-Winning Strategies through Supply Chain Partnership**, IIE Transactions. v. 30, n. 8, p 762. Agosto, 1998.

[ROTHENBERGER et al. 2003] ROTHENBERGER, M. A.; DOOLEY, K. J.; KULHANI, U. R.; NADA, N. **Strategies for Software Reuse: A Principal Component Analysis of Reuse Practices**, IEEE Transactions on Software Engineering, Vol. 29, No. 09, pp. 825-837, Setembro, 2003.

[SABBAGHI; VALDYANATHAN 2008] SABBAGHI, A.; VALDYANATHAN, G. **Effectiveness and Efficiency of RFID Technology in Supply Chain Management: Strategic values and Challenges**. Journal of Theoretical and Applied Electronic Commerce Research, Vol. 03, No. 02, Agosto 2008, pp. 71-81, ISSN 0718-1876

[SARMA; ENGELS 2003] SARMA, S; ENGELS D. **On the Future of RFID Tags and protocols**. Disponível em: <<http://www.autoidlabs.org/whitepapers/mit-autoid-tr018.pdf>>. Acesso em: 08 de fev. 2006.

[SCHMIDT 2006] SCHMIDT, D. C. **Model-Driven Engineering**, IEEE Computer, Vol. 39, No. 02, pp. 25-31, Fevereiro, 2006.

[SIMOS et al. 1996] SIMOS M.; CREPS D.; KLINGLER C.; LEVINE L.; ALLEMANG D. **Organization Domain Modeling (ODM) Guidebook**, Version 2.0 Technical Report, Junho, 1996, p.509.

[SOMMERVILLE 2006] SOMMERVILLE, I. **Software Engineering**, pp. 840, Addison Wesley

[STARS 1993] **Software Technology for Adaptable, Reliable Systems (STARS), The Reuse-Oriented Software Evolution (ROSE) Process Model**, Technical Report, Julho, 1993, p. 143.

[SUPPLY-CHAIN COUNCIL 1997] SUPPLY-CHAIN COUNCIL. **New group aims to improve supply chain integration**, Purchasing, pp. 8-32, Vol. 123, No. 6, 1997.

[SVANHNBERG; GRUP; BOSH 2001] SVAHNBERG M.; van GRUP J.; BOSH J. **On the Notion of Variabilities in Software Product Lines**. In: Working IEEE/IFIP Conference on Software Architecture, 2001, Amsterdam. **Proceedings...** Amsterdam: 2001, p. 45-54.

[TAULAVUORI; NIEMELA; KALLIO 2004] TAULAVUORI A.; NIEMELA E.; KALLIO P. **Component documentation—a key issue in software product lines**, Journal Information and Software Technology, v. 46, n. 08, p. 535–546, Junho, 2004.

[WEINSTEIN 2005] WEINSTEIN, R., **A technical overview and Its Application to the Enterprise**, IT Professional, Vol 7, No.3, pp.27-33, Junho 2005.

[WEISS; LAI 1999] WEISS D. M.; LAI C. T. R., **Software Product-Line Engineering: A Family-Based Software Development Process**. Boston: Addison-Wesley, 1999. 426 p.

[WICKS; VISICH; LI 2006] WICKS, A. M.; VISICH J. K.; LI S., **Radio Frequency Identification Applications in Hospital Environment**, heldref publications, Hosp. Top., Vol 84, No. 3, pp. 3–9, 2006.

[WINTER; ZEIDLER; STICH 2002] WINTER M.; ZEIDLER C.; STICH C. **The PECOS Software Process**. In: Workshop on Component-based Software Development, 7th International Conference on Software Reuse, 2002, Austin. **Proceedings...** Austin: 2002. p.07.