Frederico Silva de Araujo
Marcelo Jorge Parente Burdelis
Thomas Takats Tenyi

Sistema Integrador On-line da Cadeia de Suprimentos Baseado no Modelo de Reposição Contínua de Estoque Gerenciado pelo Fornecedor (VMI)

São Paulo
2007

Frederico Silva de Araujo
Marcelo Jorge Parente Burdelis
Thomas Takats Tenyi

Sistema Integrador On-line da Cadeia de Suprimentos Baseado no Modelo de Reposição Contínua de Estoque Gerenciado pelo Fornecedor (VMI)

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do titulo de Engenheiro

Área de Concentração:
Engenharia de Computação e Sistemas Digitais

Orientadores:
Profa. Dra. Selma Shin Shimizu Melnikoff
Prof. Dr. Reginaldo Arakaki

São Paulo
2007

Frederico Silva de Araujo
Marcelo Jorge Parente Burdelis
Thomas Takats Tenyi

Sistema Integrador On-line da Cadeia de Suprimentos Baseado no Modelo de Reposição Contínua de Estoque Gerenciado pelo Fornecedor (VMI)

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro

São Paulo
2007

AGRADECIMENTOS

Agradecemos a Deus por nos ter concedido a oportunidade de estudar e poder contrinuir positivamente com o mundo em que vivemos.

Agradecemos às nossas famílias e entes queridos pelo apoio e carinho dedicados durante toda nossa vida.

Agradecemos aos nossos orientadores e professores da Escola Politécnica da Universidade de São Paulo pelos ensinamentos e suporte na execução deste trabalho.

Agradecemos à Rúbia e Luiza pela paciência e compreensão dispendidas durante o ano de intenso trabalho que precedeu nossa graduação.

RESUMO

ARAUJO, F.S., BURDELIS, M. J. P., TENYI, T. T. – Sistema Integrador On-line da Cadeia de Suprimentos Baseado no Modelo de Reposição Continua de Estoque Gerenciado pelo Fornecedor. Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2007.

Este trabalho trata do projeto de um sistema de integração logística baseado no modelo de reposição contínua de estoque gerenciado pelo fornecedor (*Vendor Managed Inventory*). Após revisão da literatura, os impactos de tal sistema sobre a cadeia logística são apresentados, e a seqüência de passos para sua implementação eficaz é prescrita. Quanto ao modelo de negócio em si, é proposta uma plataforma distribuída que opere de maneira intercambiável, tanto em ambientes privados de troca de mensagens eletrônicas quanto em rede Internet, religando clientes e fornecedores a eles conectados. Atenção especial é dedicada ao projeto da arquitetura e concepção dos elementos de *software* com enfoque em baixo nível de acoplamento ente componentes, objetivando-se com isso a construção de um sistema escalável, adaptável, de fácil operação e manutenção. O uso de padrões de desenvolvimento de software é estudado e os benefícios ao sistema desenvolvido reportados. Empregando-se o estado da arte em termos de desenvolvimento de software ágil, o sistema foi implementado lançando-se mãos de tecnologia *web* e os benefícios inerentes dessa escolha elucidados.

Palavras-chave: Logística de cadeia de suprimentos. *Vendor Managed Inventory* (VMI). Automação Industrial. Engenharia de *Software*.

ABSTRACT

ARAUJO, F.S., BURDELIS, M.J.P., TENYI, T.T. – Supply Chain Integrator System based on Vendor Managed Inventory using Continuous Replacement Model. Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2007.

The aim of this project is to develop an integrated logistic system using continuous replacement of supplies managed by the retailer (Vendor Managed Inventory). After literature review, the system's impacts on the supply chain are presented, and the sequence of steps for its efficient implementation is prescribed. For the business model, a distributive platform that operates interchangeable is proposed, for both Electronic Data Interchange networks and for the Internet, linking clients and retailers connected to them. Special attention is given to the architecture and conception of software elements with focus on low levels of coupling between components, aiming to construct a scalable system that is adjustable, easy to operate and to maintain. The use of software design patterns is studied and its benefits for the development process are reported. Using the state of the art in terms of agile software development, the system is implemented by using web technology and its inherent benefits are explained.

Key Words: Supply Chain Logistics, Vendor Managed Inventory (VMI), Industrial Automation, Software Engineering.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUP	Agile Unified Process
B2B	Business-to-Business
CIDX	Chemical Industry Data Exchange
CRM	Customer Relationship Management
CSS	Cascading Style Sheets
EAP	Estrutura Analitica do Projeto
ECR	Efficient Consumer Response
EDI	Electronic Data Interchange
ERP	Enterprise Resource Planning
MER	Modelo Entidade Relacionamento
POC	Proof of Concept
SCIP	Supply Chain Integration Platform
SGBD	Sistema Gerenciador de Banco de Dados
VAN	Value Added Network
VMI	Vendor Managed Inventory
VMOI	Vendor Managed Onwed Inventory
W3C	World Wide Web Consortium

SUMÁRIO

1 Introdução

Este capítulo introduz o contexto que motivou a proposta do projeto e seus objetivos principais. Em seguida, justifica-se o desenvolvimento do proposto e apresenta-se a estrutura do trabalho.

1.1 Contextualização

Após passar boa parte da década adquirindo custosos sistemas ERP (*Enterprise Resource Planning*), CRM (*Customer Relationship Management*) e de *e-Commerce*, as empresas estão se voltando para a integração destas ilhas de informação, a fim de transcender o confinamento corporativo para gerar novas oportunidades, eliminar custos e estabelecer vantagens competitivas. Dentre as principais iniciativas para adicionar valor a uma corporação destacam-se a otimização de processos internos, a redução de custos de transação e a eficiência da cadeia de suprimentos.

Para atingir o nível de introspecção de seus negócios e cadeias de suprimento, as empresas precisam definir processos que transponham as barreiras de sistemas e parceiros, sendo independentes de uma aplicação específica, com suporte a padrões industriais abertos e portabilidade. Tentativas de implementar projetos de integração B2B (*Business-to-Business*) mostram-se um desperdício de tempo e recursos, se os processos internos não estão bem definidos e estruturados.

A redução de custos de transação é atingida através da eliminação da interferência humana (susceptível a erros), alavancagem das oportunidades providas pela Internet para conduzir transações, padronização na troca de documentos e automação da comunicação entre parceiros comerciais.

Cadeias de suprimento compreendem empresas e atividades de negócio necessárias para desenvolver, produzir, entregar e utilizar um produto ou serviço. Toda empresa se encaixa, de alguma forma, em uma cadeia de suprimentos, dependendo dela para suprir suas necessidades básicas, sobreviver e triunfar.

O atual passo de mudança e as incertezas sobre alterações no mercado têm contribuído para que empresas dêem maior atenção e importância às cadeias de suprimento nas quais participam e entender seu papel dentro delas. As empresas que souberem construir e participar de uma cadeia de suprimento forte terão uma significativa vantagem competitiva no mercado (HUGOS, 2003).

Neste contexto, destaca-se uma família de negócios de relacionamento cliente-fornecedor denominada *Vendor Managed Inventory* (VMI, ou Inventário Gerenciado pelo Fornecedor, em Português), na qual o fornecedor de um produto se responsabiliza pelo gerenciamento dos níveis de estoque de seus clientes. Desta forma, o fornecedor monitora e toma ações pró-ativas no envio de insumos para reposição de estoque do cliente e seu faturamento, de acordo com contratos de fornecimento pré-estabelecidos. O emprego deste modelo é comum para fornecedores de insumos intermediários e é utilizado, dentre outros, por gigantes do setor químico (Basf, Dow, Rhodia) e grandes empresas de atacado e varejo como Wal-Mart.

Uma das chaves para o funcionamento do VMI é a partilha de riscos. Algumas vezes, o contrato implica que, se o estoque não for utilizado, o vendedor se comprometerá a recomprá-lo. Na maioria dos casos, o produto fica em possessão do cliente, só sendo pago quando for efetivamente consumido.

1.2 Proposta do Trabalho

O objetivo deste trabalho consiste em conceber e desenvolver um sistema *on-line*, capaz de integrar a cadeia de suprimentos de clientes e fornecedores através da monitoração e controle remoto dos estoques de produtos, aplicando o conceito de reposição contínua de estoque contido no VMI.

O sistema deve integrar e automatizar de forma transparente o processo de gestão da cadeia de suprimentos, recebendo dados de estoques e informações de planejamento de produção de clientes, através de mensagens eletrônicas, e se comunicar com sistemas ERP (*Enterprise Resource Planning*) do fornecedor, com o intuito de programar entregas de produtos e realizar previsões sobre demandas

futuras. Trata-se de uma forma de reduzir custos de transação e melhorar o desempenho da cadeia produtiva na qual o fornecedor é responsável pelo gerenciamento dos níveis de estoque de seus clientes.

A execução deste projeto contempla a pesquisa e levantamento de requisitos e restrições para conceber um sistema que aplica o estado da arte em metodologia e técnicas de modelagem de *software*, vindo ao encontro de modernos conceitos e tecnologias de integração B2B.

O sistema foi denominado de **SCIP – *Supply Chain Integration Platform***.

1.3 Justificativa

O serviço de VMI é realizado, em muitas empresas, sem auxílio de ferramentas computacionais: equipes responsáveis pelo serviço realizam os cálculos necessários utilizando apenas calculadoras ou *software* de automação de escritório (por exemplo, Excel). Soma-se a esse problema outro agravante: dados cruciais para o funcionamento do negócio (níveis de estoque e planejamento de produção) normalmente são enviados para o fornecedor através de Fax ou e-mail e se encontram sujeitos a erros manuais.

Desta forma, o desenvolvimento de um aplicativo capaz de integrar os sistemas ERP de clientes e fornecedores promove a troca segura e rápida de dados necessários para o andamento do negócio, permitindo maior controle, eficácia e corte de custos no processo de monitoração e controle dos estoques. A manutenção de uma sólida integração B2B é, atualmente, um fator crítico no sucesso da comunicação e alinhamento estratégico entre parceiros e fornecedores.

Entre as vantagens decorrentes da integração propiciada pelo sistema proposto estão:

- Diminuição do tempo gasto nos cálculos de aprovisionamento, criação de programações de entrega e envio de notificações para o cliente;

- Maior segurança da informação e confiabilidade no processo, pois informações necessárias para o aprovisionamento são processadas com o mínimo de interferência humana;
- Promoção de uma arquitetura transparente onde clientes e fornecedores são ligados através da plataforma SCIP (conceito de *broker* de serviços);
- Diminuição de despesas de comunicação e gastos com re-trabalho;
- Otimização do nível de estoque; diminui-se a quantidade de material estocado possibilitando uma estrutura de fornecimento *on-demand* ou *just-in-time* automatizada e mais eficiente;
- Para os clientes: Maior confiabilidade no prazo de entregas e diminuição dos riscos de produção, transferindo-os em parte ao fornecedor;
- Para o fornecedor: Vantagem competitiva, estabelecendo-se um vínculo de fidelidade com o cliente e diminuição de falhas e erros no aprovisionamento.

Uma das principais vantagens do sistema é propiciar transparência e menos custos com estoques, abstraindo-os ao nível da informação, permitindo que empresas foquem em suas principais competências, agregando assim valor à cadeia de suprimentos como um todo (HUGOS, 2003).

A tarefa de desenvolver tal sistema é desafiadora, uma vez que o mesmo atua como intermediador de diversos sistemas diferentes, de forma adaptável, modular e escalável, a fim de atingir o maior nível de transparência entre cliente e fornecedor.

> Conectividade B2B não é mais um luxo; é uma necessidade para que empresas se mantenham competitivas. Integração B2B permite à empresa focar em suas competências principais e delegar outros serviços a parceiros de modo a ganhar eficiência e reduzir custos. (TROTTA, 2003)

1.4 Metodologia de Trabalho

O presente trabalho inspirou-se nos artefatos descritos dentro do PMBOK[1] para gerenciar e conduzir as atividades inscritas dentro do escopo do projeto do sistema SCIP.

1.4.1 Estrutura analítica e cronograma do projeto

Desta forma, partiu-se de um estudo inicial do tema para se chegar à definição do escopo e da estrutura analítica do projeto, ou EAP (vide APÊNDICE A.1), bases para o planejamento executado. Tal análise permitiu a decomposição do trabalho em partes manejáveis, identificando os elementos terminais para o planejamento, ou seja, os itens reais a serem feitos no projeto.

Conhecendo as tarefas a serem cumpridas e identificando os entregáveis, fez-se necessário elaborar um cronograma para o projeto para fins de controle e acompanhamento das atividades (vide APÊNDICE A.2). O mesmo passou por contínuas revisões ao longo das fases do projeto, recebendo os refinamentos necessários para se adequar à realidade de cada momento.

As principais fases identificadas neste projeto foram, conforme segue:

- Estudo do tema e definição do escopo;
- Planejamento;
- Análise de requisitos do projeto (compreende pesquisa, especificação e definição do plano de testes);
- Desenvolvimento (compreende preparação, planejamento da execução, codificação);
- Testes (compreende testes unitários e integrados);
- Documentação (compreende preparação dos documentos finais e confecção da monografia).

[1] O Project Management Body of Knowledge, também conhecido como PMBOK, é um conjunto de práticas em gerência de projetos levantado pelo Project Management Institute (PMI).

1.4.2 Análise preliminar de riscos

Uma análise preliminar de riscos (APR) foi realizada com a finalidade de se determinar os possíveis riscos que poderiam ocorrer na fase operacional do projeto. A APR, portanto, foi utilizada como uma ferramenta de análise inicial e planejamento do projeto, resultando na matriz de riscos do projeto (vide APÊNDICE A.5).

1.4.3 Plano de comunicação

1.4.3.1 *Responsabilidades*

Neste projeto, distinguem-se dois grupos de responsabilidades principais: orientação e desenvolvimento.

No que toca à orientação, os participantes deste grupo são responsáveis pelo acompanhamento e auxílio teórico do desenvolvimento. São eles que devem revisar e aprovar os documentos e entregáveis do projeto.

No que toca ao desenvolvimento, os participantes deste grupo são responsáveis pelo planejamento, concepção e desenvolvimento do projeto, recorrendo ao auxílio dos orientadores sempre que necessário. Além disso, devem reportar o status do projeto periodicamente aos orientadores através de relatórios de andamento do projeto.

A matriz de responsabilidades do projeto foi definida com base no diagrama RACI, conforme nomenclatura abaixo:

- ***R**esponsible* (pessoa responsável pela atividade);
- ***A**ssists* (pessoas que assistem o responsável da tarefa);
- ***C**onsulted* (pessoas que opinam e orientam);
- ***I**nformed* (pessoas que apenas são informadas das atividades realizadas).

Vide APÊNDICE A.4 para a matriz de recursos do projeto.

1.4.3.2 *Reuniões*

As reuniões foram programadas e então estabelecidas em um cronograma de reuniões entre orientados e orientadores. A freqüência de ocorrência, semanal, tem como resultado de cada uma, definições do projeto, metas e objetivos a serem atendidos, devidamente registrados em ata.

Os responsáveis pelas atas de reunião são os próprios desenvolvedores, que devem seguir um modelo padronizado (como toda a documentação deste projeto) e reportá-los aos demais *stakholders* do projeto.

A comunicação pode ser feita de forma rápida e eficiente, por e-mail ou telefone, podendo-se marcar encontros não programados no calendário de reuniões no momento que se julgar necessário.

1.4.4 Recursos Empregados

1.4.4.1 *Controle de versões*

Com o intuito de facilitar a comunicação entre os diferentes *stakholders*, toda a informação do projeto foi centralizada num repositório *online* com suporte a controle de versões (SVN). Desta forma, cada membro da equipe de desenvolvimento e orientadores possui acesso fácil e rápido a documentos, código fonte, cronograma, calendários de reuniões etc.

Para o controle de versões, dispomos de um servidor SVN e de clientes *Tortoise Subversion* que podem persistir e recuperar arquivos no servidor, mantendo sempre uma cópia controlada em suas máquinas locais. A cada alteração no repositório, um *log* é gerado e armazenado no servidor informando com os detalhes das modificações realizadas e uma mensagem de correio eletrônico com uma cópia do mesmo é enviada para os destinatários cadastrados no sistema.

Desta forma, tem-se de maneira centralizada, o controle da informação do projeto, evitando-se perdas e replicações que fatalmente ocorrem caso não exista um controle rígido por parte dos gestores do projeto.

1.4.4.2 *Trac*

O *Trac* é uma ferramenta muito útil para efeitos de rastreabilidade e acompanhamento da evolução do projeto. Desta forma, ele foi incluído no projeto com intuito de aumentar a visibilidade dos progressos conquistados e disponibilizar um mecanismo automatizado de adição de pendências e responsabilidades pelas mesmas.

Nele encontramos um *roadmap*, uma linha do tempo, registro de pendências e um *wiki* contendo informações bibliográficas do projeto. Tudo isso pode ser acessado através de um endereço eletrônico e por meio de uma interface amigável.

1.5 Organização do Documento

Neste contexto, o capítulo 2 apresenta os principais conceitos logísticos e os requisitos necessários para compreensão dos processos dentro de uma cadeia de suprimentos e do modelo de negócio VMI.

O capítulo 3 aborda os aspectos que permearam a concepção do modelo de negócio do sistema SCIP – *Supply Chain Integration Platform*. Dá-se particular atenção à modelagem dos processos de negócio, inscrito dentro do contexto de reposição contínua de estoque gerenciado pelo fornecedor.

O capítulo 4 discorre sobre os principais conceitos de engenharia de *software* aplicados no projeto e desenvolvimento do sistema proposto (leia-se, por desenvolvimento, da fase de análise de requisitos aos testes integrados), ressaltando aspectos de arquitetura de *software*, padrões e metodologias de desenvolvimento.

O capítulo 5 concerne às fases de especificação e desenvolvimento do projeto, contemplando aspectos de especificação de requisitos e modelagem de *software*. A teoria nele exposta, somada aos processos de negócio apresentados no capítulo 4, serviu de base para a construção dos artefatos que orientaram a implementação do sistema SCIP.

O capítulo 6 apresenta as conclusões obtidas com o trabalho e realiza uma reflexão sobre futuras aplicações e extensões pertinentes ao sistema desenvolvido.

2 Logística de Cadeia de Suprimentos

Neste capítulo, são apresentados os fundamentos teóricos que motivam a integração da cadeia de suprimentos e as técnicas logísticas que podem ser empregadas para agregar vantagens competitivas às organizações. Neste cenário, o conjunto de ferramentas e estratégias que buscam aumentar o nível de serviço, agregando valor percebido ao produto, é denominado *Efficient Consumer Response* (ECR).

Uma atenção particular é dada aos aspectos de gerenciamento de estoques e à técnica de reposição contínua chamada *Vendor Managed Inventory* (VMI), onde o fornecedor assume a responsabilidade de gerenciar o estoque de seus clientes.

2.1 Fundamentos Logísticos

Segundo Porter (1988), diversas empresas têm percebido que a atividade logística é uma forte instauradora de barreiras competitivas, reduzindo custos, prazos e atuando na melhoria de diversos serviços (como atendimento, por exemplo). Desta forma, empresas têm buscado inovar seus produtos e serviços agregando diferenciais aos mesmos através de atividades de armazenamento, distribuição, planejamento, integração, gestão de estoques e outras técnicas baseadas no conceito logístico.

Diferentemente do que se acreditava no início da década de 80, hoje se verifica que as empresas já não concorrem de forma individual. Nota-se que, atualmente, são as cadeias de suprimento que competem entre si, devido à pluralidade de empresas que as compõem, buscando fortalecimento através de políticas de aliança e parcerias. Empresas que trabalham de forma independente de seus clientes e fornecedores tendem a ser mais ineficientes e terem mais custos do que empresas que trabalham de forma integrada.

Desta forma, um número cada vez maior de empresas tem compreendido a necessidade de colaboração com outros participantes de suas cadeias de suprimentos. Esta integração tem permitido a otimização de custos e uma maior

agilidade em todos os processos dependentes, apresentando-se não mais como uma vantagem competitiva, mas como um meio de sobrevivência no mercado.

Segundo Lambert, Stock e Vantine (1998) apud Tavares (2003), o Gerenciamento integrado da Cadeia de Suprimentos (*Suppy Chain Management* - SCM) representa uma importante área de pesquisa na busca pelo diferencial competitivo das organizações e foi definido como sendo:

> O Supply Chain Management é a integração dos processos de negócio desde o usuário final até os fornecedores originais que proporcionaram os produtos, serviços e informações que agregam valor para o cliente. (LAMBERT, STOCK e VANTINE, 1998 apud TAVARES, 2003, p.15)

O gerenciamento da cadeia de suprimentos é um determinante fundamental da vantagem competitiva. Pequenas ações, tais como a partilha de previsões de consumo entre clientes e fornecedores permitem ganhos consideráveis. Entre eles pode-se citar o melhor planejamento da produção por parte do fornecedor e, conseqüentemente, uma otimização de prazos de entrega, beneficiando também o cliente.

Fornecedores deixaram de ser vistos apenas como provedores de matéria-prima e hoje passaram a fazer parte do planejamento das empresas dos clientes, definindo custos, processos e atuando até no curso de melhoria e desenvolvimento de produtos. Diversas técnicas e estratégias são utilizadas para criar benefícios, entre elas podemos citar:

- *Just in time*, suprimento de insumos sob demanda, mantendo os níveis de estoque tão baixo quanto possível;
- Integração através da troca eletrônica de dados (*Electronic Data Interchange* – EDI), largamente utilizado no mercado para a realização de comércio eletrônico;
- Resposta Eficiente ao Consumidor (*Efficient Consumer Response* – ECR).

A Figura 1 demonstra um exemplo de cadeia integrada de suprimentos. A utilização de estratégias logísticas e de um fluxo mais apurado de informações permite a

redução dos níveis de estoque. Desta forma, organizações que outrora estocavam grandes volumes de matéria-prima e produto acabado passaram a reduzir os mesmos, por representarem custos desnecessários em suas cadeias de suprimentos, diminuindo assim seus custos e aumentando o fluxo de mercadorias.

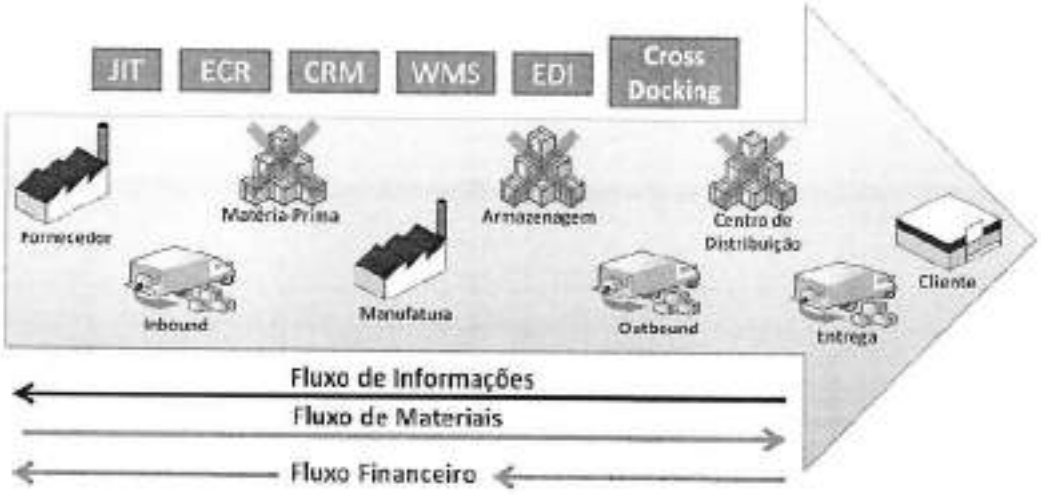


Figura 1: Visão Integrada da Cadeia de Suprimentos[1].

2.2 *Efficient Consumer Response* (ECR)

Dentro do modelo de foco na eficiência da cadeia de suprimentos como um todo, ao invés da eficiência individual das partes, surgiu um movimento chamado ECR – *Efficient Consumer Response*, que pode ser traduzido como sendo Resposta Eficiente ao Consumidor (Tavares, 2003).

Este movimento foi iniciado no setor supermercadista dos Estados Unidos no final da década de 80. O mesmo foi criado por redes de supermercados que buscavam oferecer uma resposta à perda de espaço causada pela rede Wal-Mart.

O ECR propõe economia de custos através da aplicação de três fatores:

- Redução de custos através da troca de informações por toda a cadeia produtiva;

[1] Adaptado de BALLOU R. H. Logística Empresarial. São Paulo: Ed. Atlas, 1993

- Uso de novas políticas de gerenciamento de estoques e de riscos;
- Ganhos financeiros derivados do aumento da produtividade e diminuição de estoques.

2.3 Troca Eletrônica de Dados

A troca eletrônica de dados ou EDI é uma prática altamente difundida no exterior que atualmente começa a se popularizar no Brasil, principalmente no mercado automotivo. O EDI consiste na troca automatizada de dados entre empresas através do uso de redes e de padrões predefinidos de mensagens. (TAVARES, 2003)

Atualmente, diversas informações são trocadas entre empresas utilizando EDI. Entre estas, pode-se citar como exemplo:

- Pedido de compra de produto ou programação de entrega – consiste em uma mensagem EDI enviada de um cliente para um fornecedor, efetuando um pedido de compra ou um conjunto de pedidos constituindo uma programação de entrega;
- Confirmação de recebimento de pedido de compra ou de programação de entrega – consiste em uma mensagem EDI enviada do fornecedor para o cliente, confirmando o recebimento de uma programação de entrega ou de um pedido de compra;
- Aviso de embarque – consiste em uma mensagem EDI enviada do fornecedor para o cliente, indicando a saída de mercadoria para entrega. Contém dados importantes para a entrada de mercadoria no cliente (data prevista de entrega, número de pedido, número de nota fiscal, impostos, entre outros);
- Previsões de consumo – consiste em uma mensagem EDI enviada pelo consumidor ao fornecedor, indicando suas previsões de consumo.

O uso do EDI apresenta diversas vantagens. Dados que antes eram trocados via telefone, fax ou email passam a ser trocados de forma segura e sem nenhum tratamento manual. Isso implica em maior agilidade para todos os parceiros que se comunicam de forma eletrônica, diminuindo de forma drástica o trabalho de entrada

manual de dados em sistemas e, conseqüentemente, reduzindo retrabalhos devido a erros de digitação.

Normalmente, as mensagens EDI são trocadas através do uso de *Value Added Networks* (VANs). Estas redes se diferenciam da Internet por serem mais seguras e apresentarem um alto nível de qualidade de serviço, com garantia de envio e recebimento de informações pelas empresas. Entretanto, com a difusão da Internet, algumas empresas iniciaram a troca de informações EDI através do uso de conexões seguras e FTP.

Com o objetivo de regulamentar e normalizar as transferências de documentos EDI, diversos padrões foram criados. Entre estes podem-se citar:

- UN/EDIFACT (Desenvolvido pelas Nações Unidas, é o padrão predominante no mundo);
- ANSI ASC X12 (Desenvolvido nos Estados Unidos e predominante nesta região);
- ODETTE (Padrão automotivo inglês);
- ChemXML (Padrão EDI desenvolvido pelo consórcio das empresas químicas CIDX, é baseado em XML);
- RND (Padrão automotivo da indústria brasileira).

2.4 Reposição Contínua de Estoques

Com o objetivo de reduzir níveis de estoque ao ponto mínimo, sem que ocorram faltas de produtos, diversos clientes e fornecedores têm adotado a técnica da reposição contínua. Reposição contínua é todo processo automático de reposição de mercadoria entre um cliente e fornecedor, utilizando-se a troca eletrônica de informações.

No processo de reposição contínua, algoritmos executados por computadores são responsáveis por calcular quantidades a serem repostas nos estoques. Estes algoritmos se baseiam em consumos históricos e em níveis de estoque existentes.

Todo cálculo é executado de forma automática, sem interferência humana, garantindo um processo seguro e livre de erros de cálculo e digitação.

A reposição continua apresenta os seguintes benefícios para a cadeia de suprimento:

- Redução de erros e retrabalhos na criação de pedidos. Redução de custos logísticos devido à racionalização dos transportes;
- Melhoria no planejamento e diminuição de pedidos de última hora;
- Ganho de tempo, pois todo cálculo de aprovisionamento é realizado de forma automática.

Para que a reposição continua tenha sucesso, os seguintes pontos chave devem ser tratados:

- As empresas participantes devem ter uma relação comercial estável que permita fluxo de produtos sem renegociação de preços a cada pedido;
- Fornecimento garantido pelo fornecedor, com acurácia na entrega de pedidos nas datas combinadas;
- Conhecimento mútuo e respeito das regras de negócio (*lead-time*, quantidade mínima a ser entregue, calendários de entrega e recebimento de produtos) por ambas as partes;
- Agilidade para a troca de informações entre as empresas, se possível utilizando EDI;
- Uso de algoritmo eficiente para realização do cálculo de aprovisionamento.

O modelo do VMI é uma técnica de reposição continua. Entretanto, neste modelo o fornecedor tem a total responsabilidade de realizar a gestão de estoque de seu cliente. Cabe ao fornecedor processar o algoritmo de aprovisionamento para definir datas e quantidades de entrega. Este é executado utilizando-se como parâmetros regras de negócios pré-definidas por ambas as partes e também dados fornecidos pelo cliente (previsões de consumo, níveis de estoque). (GAPSKI, 2003)

2.5 *Vendor Managed Inventory* (VMI)

2.5.1 Definição

O *Vendor Managed Inventory* (VMI) é uma técnica amplamente utilizada em programas de reposição contínua. Nela, o fornecedor de um produto se responsabiliza pelo gerenciamento dos níveis de estoque de seus clientes. É um negócio comum para fornecedores de insumos intermediários e é utilizado por atores importantes do setor químico e grandes empresas de atacado e varejo. (BOWERSOX, 2002)

Uma das chaves para o funcionamento do VMI é a partilha de riscos. Algumas vezes, o contrato implica que, se o estoque não for utilizado, o vendedor se comprometerá a recomprá-lo. Existe uma variação do VMI chamada de *Vendor Managed Owned Inventory* (VMOI). A diferença entre o VMOI e o VMI é que o VMOI agrega o conceito de estoque consignado. Neste caso o estoque de produto no cliente pertence ao fornecedor, o cliente paga apenas pelo produto consumido. A cobrança pode ser feita de forma quinzenal, mensal, bimestral, etc. dependendo da relação existente entre o fornecedor e cliente. Em geral, no VMOI, é de grande interesse para o fornecedor que o estoque no cliente seja o mais baixo possível.

O VMI ou o VMOI é realizado a partir do monitoramento dos níveis de estoque do cliente. Existem no mercado basicamente dois tipos de VMI/VMOI:

- Baseado apenas no nível de estoque atual do cliente: Neste caso, o fornecedor envia uma quantidade pré-definida de produto para o cliente quando seu nível de estoque atinge um determinado valor de aprovisionamento;
- Baseado no nível de estoque atual e em previsões futuras de consumo: O fornecedor realiza cálculos de aprovisionamento basea-se em previsões de consumo do cliente. Pontos de aprovisionamento (datas e quantidades) são definidos a partir destes dados.

O uso do VMI/VMOI busca atingir os seguintes benefícios para a cadeia de suprimentos:

- Redução de pedidos de última hora (*rush orders*);
- Maior regularidade de produção devido ao melhor planejamento;
- Garantia de recebimento de produto por parte do cliente;
- Exclusividade de fornecimento por parte do fornecedor;
- Melhoria de serviço para o cliente;
- Redução de níveis de estoque;
- Alinhamento da produção dos fabricantes com demandas dos clientes.

2.5.2 Passos para o VMI

A seguir são listados os principais passos para o estabelecimento de uma relação VMI entre dois parceiros. Os passos foram enumerados a partir da experiência de mercado dos integrantes deste trabalho de conclusão de curso e também na metodologia apresentada por Taras (2003) apud Tavares (2003) em um trabalho que consolida modelos de diversos outros autores.

Passo 1: Confiabilidade, credibilidade e comprometimento entre parceiros

Para iniciar qualquer projeto de reposição contínua é necessário que existam confiabilidade, credibilidade e comprometimento entre os parceiros da cadeia de suprimentos. Ambos os parceiros ganham com a implementação de uma relação VMI/VMOI. Para que o projeto seja um sucesso é necessário comprometimento mútuo com os resultados. O fornecedor deve ser capaz de suprir a demanda do cliente, e o cliente deve ser transparente e fornecer seus dados de produção ao fornecedor.

Passo 2: Comprometimento de gerências e diretorias

É necessária compreensão do funcionamento do programa e das suas diversas vantagens pelo alto escalão das empresas parceiras. A partir desta compreensão, os membros do alto escalão devem se comprometer com o bom andamento do

programa e identificá-lo como objetivo estratégico de ambas as empresas. O programa deve então ser divulgado a todos os colaboradores. Recursos devem ser alocados ao projeto para que o mesmo possa ser implantado com sucesso.

Passo 3: Aceitação dos colaboradores

A divulgação pelo alto escalão deve ser feita a todos os colaboradores como algo positivo e alinhado às estratégias da empresa. Todos os colaboradores devem compreender o conceito, principalmente os que estarão diretamente envolvidos com o resultado final, como é o caso dos vendedores do fornecedor (*costumer service*) e os compradores ou responsáveis pela gestão de estoque no cliente.

Os colaboradores devem enxergar o processo como uma forma de aprimoramento de seus trabalhos, tornando os mais produtivos. Sem a participação e a aceitação dos colaboradores, o processo de VMI/VMOI não obterá o sucesso esperado.

Passo 4: Definição de informações a serem trocadas e forma de troca

Neste passo, os parceiros devem definir as informações que devem ser trocadas durante o funcionamento do VMI/VMOI, a freqüência de envio dos dados e o meio de troca utilizado. Normalmente devem ser trocados os dados apresentados na Tabela 1.

Tabela 1: Mensagens e seus emissores.

Direção da Mensagem	Tipo de Mensagem	Freqüência Recomendada de Envio
Do cliente para o fornecedor	Nível de estoque do cliente	Diária
Do cliente para o fornecedor	Planejamento de produção do cliente *	Diária
Do cliente para o fornecedor	Confirmação de recebimento de produtos sempre que uma remessa é descarregada no cliente*	Semanal
Do fornecedor para o cliente	Confirmação de criação de remessas no sistema ERP do fornecedor	Sempre que uma remessa é criada no sistema ERP do fornecedor
Do fornecedor para o cliente	Avisos de embarque de mercadoria para ser entregue ao cliente	Semanal, enviada assim que produto deixa o centro de distribuição

Estes dados podem ser trocados das seguintes formas:

- Via Telefone ou Fax;
- Via Email;
- Via Portais Internet que disponibilizam dados para visualização ou *download*;
- Via EDI (recomendado).

Passo 5: Definição de dados

Nesta passo devem ser definidos alguns itens, conforme segue:

Produtos que farão parte da relação VMI

- Nome do produto no cliente;
- Nome do produto no fornecedor;
- Código do produto no cliente;
- Código do produto no fornecedor;
- Descrição do produto;

Regras de negócio para cada produto da relação seguinte (as regras de negócio dependem do algoritmo VMI/VMOI utilizado)

- Estoque mínimo no cliente: quantidade mínima de produto que o cliente pode ter;
- Estoque máximo: quantidade máxima de produto que o cliente pode armazenar;
- Mínima quantidade a ser entregue: quantidade mínima que o fornecedor realiza por entrega. Em tráfego terrestre pode corresponder, por exemplo, ao valor de um caminhão carregado completamente;
- Quantidade incremental: são os múltiplos incrementais a serem utilizados, caso seja necessário enviar uma quantidade maior que a mínima para o cliente;
- Tempo de trânsito: tempo médio desde a partida do produto do centro de distribuição do fornecedor até o recebimento do mesmo no cliente;

- Tempo de processamento: demora média para que a expedição de um pedido seja realizada a partir do momento que o pedido é criado no fornecedor;
- *Lead-Time*: tempo de trânsito somado ao tempo de processamento;
- Calendário de consumo: calendário de consumo do cliente. Normalmente apresenta o número de dias da semana em que o cliente consome o produto;
- Calendário de entrega: calendário que indica os dias nos quais o fornecedor pode realizar envio do produto;
- Calendário de recebimento: calendário que indica em quais dias o cliente esta aberto para recebimento de produto.

Passo 6: Testes e homologação

Nesta etapa devem ser realizados testes com o intuito de:

- Testar os meios de comunicação entre as empresas parceiras. Caso esteja sendo utilizado o EDI é necessário testar o funcionamento e integração de todas as mensagens a serem trocadas;
- Validar freqüências de envio de informação;
- Validar regras de negócio estabelecidas;
- Validar algoritmo de aprovisionamento utilizado;
- Validar funcionamento de todos os aplicativos utilizados no processo.

Os testes devem englobar todos os usuários envolvidos no processo. Para realizar os testes, normalmente as empresas utilizando dados reais e simulam o funcionamento do VMI durante pelo menos dois meses para homologar o processo.

Passo 7: Conhecimento e aceitação

Ambas as partes devem estar cientes dos seguintes fatores:

- Responsabilidades de cada parceiro no processo;
- Metodologias utilizadas para estimar as quantidades de reposição e previsões de consumo;
- Procedimentos a serem realizados por ambas as partes;

- Planos de contingência.

O consentimento mútuo de todos os fatores citados anteriormente é de suma importância para que não ocorram falhas devido à falta de procedimentos, desvios entre previsões de produção e demanda real e falta de dados devido à irresponsabilidade dos parceiros.

Passo 8: Definição de indicadores de desempenho

Ambas as partes devem definir indicadores de desempenho para acompanhar o sucesso do modelo. Os indicadores servirão como guias para acompanhar o funcionamento do processo, apontando resultados e possíveis pontos de melhoria. É importante que ambas as partes possuam, no mínimo, um ano de histórico com informações de estoque e vendas para se possa compará-los com dados obtidos após a implantação do VMI.

Passo 9: Início da operação

Terminados todos os passos anteriores é estabelecido então o início da operação. Nos três meses iniciais, todos os envolvidos devem dar atenção especial ao processo VMI. Recomenda-se que o responsável por vendas no fornecedor revise os cálculos de aprovisionamento realizados durante este período.

2.5.3 Algoritmo do VMI

O algoritmo do VMI é um conjunto de fórmulas e parâmetros que tem como objetivo definir quantidades e momentos para a reposição de níveis de estoque.

Existem basicamente duas classes de algoritmos de aprovisionamento:

- Algoritmos que não levam em conta previsões futuras de consumo. Estes algoritmos são mais comuns e menos eficientes.

- Algoritmos que levam em conta previsões futuras de consumo. São mais eficientes, entretanto são menos comuns no mercado devido à maior complexidade de execução.

Ambos os algoritmos listados anteriormente necessitam dos seguintes parâmetros para serem executados:

- Tempo médio de reposição (*Lead-time*);
- Nível de estoque atual do cliente;
- Estoque de segurança.

O tempo médio de reposição é definido como a soma dos seguintes tempos:

- Tempo médio de trânsito: tempo médio desde a partida do produto do centro de distribuição do fornecedor até o recebimento do mesmo no cliente;
- Tempo médio de processamento: demora média para que a expedição de um pedido seja realizada a partir do momento que o pedido é criado no fornecedor.

Segundo Tavares (2003), o estoque de segurança (ES), pode ser calculado utilizando a seguinte equação:

$$ES = k\,(\,\sigma^{2}VM * TR + \sigma^{2}TR * VM^{2}\,)^{1/2}$$

Legenda:
σVM = desvio-padrão da série de saídas do centro de distribuição ou de venda a consumidor final
σTR = desvio-padrão do tempo de reposição
TR = tempo médio de reposição
VM = venda média diária nas últimas "n" semanas.
k = fator de segurança

O fator de segurança **k** está intimamente ligado aos riscos de se manter estoques e ao nível de confiança que se quer garantir. Este fator deve ser aplicado no cálculo

do estoque de segurança, pois raramente a demanda será a mesma e pode variar das estimativas iniciais, como apresenta Wanke (1999) apud Tavares (2003).

2.5.3.1 Algoritmos que não levam em conta previsões futuras de consumo

Este tipo de algoritmo se baseia em dias médios de venda (ddv) para definir um ponto de reposição chamado de ponto de pedido (PP). Sempre que o estoque do cliente chegar a PP, um pedido deve ser criado com os seguintes parâmetros:

- Data de entrega: D + tempo médio de reposição, onde D é o dia atual.
- Quantidade a ser entregue: Lote de reposição (LR).

A variável ddv indica a quantidade média de produto consumida pelo cliente por dia. O nome desta variável teve origem no mercado varejista, onde a quantidade vendida de produto equivale à quantidade consumida pelo estabelecimento. A ddv pode ser calculada utilizando a seguinte fórmula:

$$\mathbf{1ddv = (QV) / (DE)}$$

Legenda:
QV = Quantidade vendida
DE = Quantidade de dias de estoque

Caso o cliente possua variação de consumação devido a realizações de promoções de vendas, deve ser utilizada a seguinte fórmula para o cálculo da ddv:

$$\mathbf{1ddv = (QV\text{-}QP) / (DE - DP)}$$

Legenda:
QV = Quantidade vendida
QP = Quantidade vendida em promoção
DE = Quantidade de dias de estoque
DP = Quantidade de dias de promoção

Para identificar o ponto de pedido (PP), ou seja, o ponto atingido pelo estoque que irá disparar o processo de reposição, a seguinte fórmula é utilizada:

$$PP = 1ddv * TR + ES$$

Legenda:
1ddv = dia médio de venda
TR = tempo médio de reposição
ES = estoque de segurança

Sempre que o nível de estoque de um produto atingir o ponto de pedido, um pedido deve ser criado para realizar aprovisionamento de estoque. Conforme mencionado anteriormente, a data de entrega do pedido é D + tempo médio de reposição, e a quantidade a ser entregue é um lote de reposição.

O valor do lote de reposição pode variar e deve ser calculado sempre que o ponto de pedido for atingido. Para calcular o lote de reposição, pode-se utilizar a fórmula a seguir:

$$LR = E_{max} - (E_{atual} + E_{tr}) + 1ddv * TR$$

Legenda:
Emax = Estoque máximo admitido para a mercadoria
Eatual = Estoque atual no depósito
Etr = Estoque em trânsito ou em pedido já confirmado
TR = tempo médio de reposição
1ddv = dia médio de venda

A quantidade obtida como necessária deverá ser arredondada para se ajustar à capacidade das embalagens do produto, de *pallets*, de *containers* e de outras embalagens de despacho (Tavares, 2003).

O algoritmo especificado acima é base para todas as técnicas VMI que não levam em conta previsões futuras de consumo. Pequenas variações podem ser encontradas no mercado quanto às formas de cálculo dos parâmetros envolvidos;

Entretanto o conceito principal sempre será o mesmo para todos os utilizadores desta técnica de VMI: Criar um pedido de aprovisionamento sempre que o nível de estoque do cliente atingir o ponto de pedido.

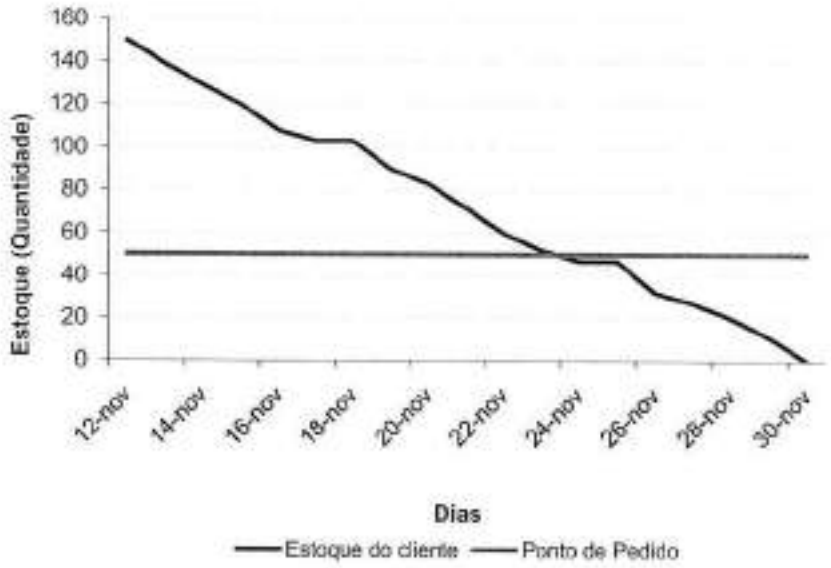


Figura 2: Curva de estoque de cliente e ponto de pedido.

Esta técnica possui uma grande fragilidade: uma vez que todos os cálculos se baseiam em dias médios de venda, caso o cliente sofra uma grande variação em seu consumo (devido à parada súbita de produção ou então devido a um aumento de consumo transitório) podndo acarretar riscos sérios como falta de produto no cliente ou então estouro de estoque devido a recebimento de produtos em excesso.

2.5.3.2 Algoritmos que levam em conta previsões futuras de consumo

Este tipo de algoritmo se baseia nas previsões futuras de consumo do cliente para calcular datas e quantidades de entrega. O uso destas informações no cálculo de aprovisionamento corrige a principal fragilidade presente nos algoritmos que não levam em conta previsões futuras: a falta de adaptabilidade a variações de consumo.

A operação base do algoritmo é o cálculo do estoque projetado. O estoque projetado é uma estimativa diária do estoque do cliente e pode ser calculado através da seguinte formula:

$$EP(d) = E_{atual} - \Sigma Consumos(d) + \Sigma E_{tr}(d)$$

Legenda:

EP(d) = Estoque projetado para o dia d

Eatual = Estoque atual no depósito do cliente

ΣConsumos(d) = Somatório de previsões de consumo desde o dia atual até o dia d

ΣEtr(d) = Somatório de todo estoque em trânsito a ser entregue até o dia d mais somatório de pedidos já confirmados para entrega até o dia d

O algoritmo consiste em criar um pedido sempre que o estoque projetado atingir o estoque de segurança. A data de entrega é o dia **d**, dia no qual o estoque projetado atingiu o estoque de segurança. O calculo do lote de reposição depende da técnica utilizada pelo algoritmo. Em uma versão simplificada, a fórmula utilizada pode ser a mostrada anteriormente na seção 2.5.3.1.

Exemplo de funcionamento do algoritmo:

Parâmetros

- Data atual = 17/11/07;
- Estoque atual no cliente: 3500 unidades;
- Previsões de consumo em unidades para os próximos sete dias.

O algoritmo é iniciado calculando o estoque projetado. O estoque projetado é calculado até a última previsão de consumo fornecida pelo cliente (neste caso 17 + 7 = 24), conforme a Tabela 2.

Tabela 2: Cálculo de estoque projetado.

Data	17/11/2007	18/11/2007	19/11/2007	20/11/2007	21/11/2007	22/11/2007	23/11/2007
Previsões de consumo	1687	200	365	987	10	762	1002
Entregas Confirmadas	0	0	0	0	0	0	0
Estoque Projetado	1813	1613	1248	261	**251**	**-511**	**-1513**
Estoque de Segurança	500	500	500	500	500	500	500

No dia em que o estoque projetado atinge o estoque de segurança, uma ordem deve ser planejada. A quantidade a ser entregue depende do algoritmo utilizado. No

exemplo considerado será criada uma ordem de 2500 unidades para o dia 20/11/2007, conforme a Tabela 3.

Tabela 3: Criação de pedido.

Data	17/11/2007	18/11/2007	19/11/2007	20/11/2007	21/11/2007	22/11/2007	23/11/2007
Previsões de consumo	1687	200	365	987	10	762	1002
Entregas Confirmadas	0	0	0	**2500**	0	0	0
Estoque Projetado	1813	1613	1248	2761	2751	1989	987
Estoque de Segurança	500	500	500	500	500	500	500

O estoque projetado é então recalculado, agora considerando a ordem planejada. Caso o estoque projetado atinja novamente o estoque de segurança em uma data futura, uma nova ordem deve ser planejada. O algoritmo repete até que o estoque projetado seja superior ao estoque de segurança para todo horizonte previsto.

O algoritmo enunciado nesta seção apresenta o funcionamento simplificado da técnica de reposição baseada em previsões futuras de consumo. Técnicas mais elaboradas devem considerar restrições de entrega para a criação de pedidos. Caso não seja possível criar uma entrega em um determinado dia (posto que o cliente tenha restrições de recebimento de produto), o algoritmo deve ser capaz de encontrar a melhor data para a entrega.

Este algoritmo é menos comum no mercado devido à sua complexidade. Nem todos os fornecedores são capazes de integrar via EDI às previsões de consumo de seus clientes, e o trabalho manual para entrar estes dados em sistema é grande demais para ser realizado diariamente.

2.5.4 Benefícios do VMI

Segundo avaliações realizadas por empresas que aplicaram o VMI, podem-se citar os benefícios ao cliente e ao fabricante. (WALLER, 1999 apud TAVARES, 2003)

2.5.4.1 *Benefícios ao Cliente*

- Redução de falta de mercadorias (*stockouts*) e dos níveis de estoque, pois reposições são mais constantes e baseadas no movimento real;

- Redução dos custos de planejamento e pedido, uma vez que estes são repassados ao fabricante;
- O nível global de serviço é melhorado tendo o produto certo no momento certo;
- O fabricante fica mais focado em melhorar seu nível de serviço, em função da responsabilidade adquirida.

2.5.4.2 *Benefícios ao Fabricante*

- A informação de previsão de consumo facilita o processo de planejamento de produção. O planejamento, antes realizado com uma visão das vendas feitas aos clientes, passa a ser realizado com a informação correta dos níveis de estoques e o movimento de consumo dos mesmos;
- Redução na quantidade de erros em pedidos de clientes, pois estes passam a ser realizados diretamente pelo fabricante. Conseqüentemente, existe uma redução de devoluções de mercadorias entregues fora dos conformes;
- Vantagem competitiva;
- Oportunidade de negócios e serviços adicionais;
- Vínculo de fidelidade através de um diferencial competitivo;

2.5.5 Riscos Potenciais

Potenciais problemas podem interferir no funcionamento do VMI. Alguns pontos devem receber atenção especial para garantir a execução correta do processo.

2.5.5.1 *Risco de comunicação na troca de dados*

O cliente é responsável por fornecer seus dados de produção ao fornecedor. Diversos problemas ocorrem em processos VMI devido às falhas na transmissão de dados. Em processos onde o EDI não está presente, é comum a existência de problemas de envio devido às falhas humanas. Em sistemas onde o EDI está presente, podem-se observar falhas de envio devido a erros de sistema. Ambas as partes devem garantir a troca de informações e possuir planos de contingência para garantir a integridade do processo.

2.5.5.2 *Risco no processo de EDI*

A grande quantidade de padrões existentes para a comunicação de dados entre computadores dificultam, mesmo no padrão EDI, a integração de dados entre clientes e fornecedores. Assim, devem-se realizar testes planejados e sistematizados no processo de troca de informações para validar os dados que são enviados. Deve-se garantir que o cliente está enviando todos os dados necessários e que cada campo é alimentado com informações corretas.

2.5.5.3 *Aceitação por parte dos funcionários*

Todos os envolvidos no processo devem compreender o funcionamento do VMI e respeitar seus procedimentos. A falta de compreensão e a falta de aceitação podem representar grandes riscos para a cadeia de suprimentos. Membros de alto escalão das empresas envolvidas devem garantir a aceitação do processo por parte de seus subordinados.

2.5.5.4 *Falta de acompanhamento de indicadores*

Parâmetros como estoque máximo, estoque mínimo, ponto de pedido, nível de serviço desejado, entre outros, podem sofrer variações e devem ser constantemente revistos e ajustados para acordar o VMI a novas realidades de mercado.

2.5.6 Tecnologia da Informação

Na implantação de um sistema VMI, a tecnologia da informação pode ser empregada para fornecer benefícios ao processo através da automação do mesmo. Dentre as vantagens oferecidas pelo uso da tecnologia da informação, podem ser listados os seguintes benefícios:

- Otimização de processos;
- Redução de tempos envolvidos;
- Maior confiança na execução do processo;
- Redução de custos de processo;

- Maior controle da atividade como um todo.

O nível de automação dos processos dependerá muito da capacidade conjunta do cliente e fornecedor em utilizar sistemas de informação. Tais benefícios podem ser alcançados utilizando um conjunto de recursos apresentados a seguir.

2.5.6.1 *Integração EDI para troca de dados entre os parceiros*

Permite otimizar processo de recebimento e envio de informações, reduzindo tempos envolvidos, aumentando a confiança na execução do processo, além de reduzir custos ligados a retrabalho e aumentar rastreabilidade de erros.

2.5.6.2 *Cálculo de aprovisionamento e monitoramento de estoque via software*

Otimiza o processo de aprovisionamento reduzindo tempos envolvidos, além de aumentar a confiança na execução do processo, reduzir custos ligados a retrabalho e aumentar rastreabilidade de erros.

2.5.6.3 *Criação de pedidos de forma automática*

Pedidos podem ser criados no sistema ERP do fornecedor de forma automática sem interferência humana, reduzindo erros de digitação e, conseqüentemente, aumentando confiança no processo e reduzindo tempos envolvidos.

2.5.7 Considerações sobre o VMI

O VMI não é apenas uma inovação na forma de se gerenciar estoques em depósitos de clientes. Trata-se de uma mudança de paradigma no modelo de gestão de estoques, mudando a forma de relacionamento entre parceiros e provendo maior integração nos elos da cadeia de suprimentos. Estas mudanças impactam diretamente os aspectos estratégicos das empresas envolvidas. Desta forma, todo processo de implantação de um VMI deve ser feito de forma cuidadosa, respeitando todos os passos de estabelecimento mencionadas na seção 2.5.2.

3 Processos de Negócio

Neste capítulo, são abordados os aspectos que permearam a concepção do modelo de negócio do sistema SCIP – *Supply Chain Integration Platform*. Dá-se particular atenção à modelagem dos processos de negócio, inscrito dentro do contexto de reposição contínua de estoque, gerenciado pelo fornecedor.

3.1 Atuação

O intuito do sistema SCIP é integrar sistemas ERP de clientes e fornecedores que desejem utilizar o modelo de reposição contínua de estoque VMI.

O SCIP atua como mediador dos sistemas ERP envolvidos através do fornecimento de serviços de integração e gestão. Uma vez operando, o sistema se responsabiliza por realizar a gestão de estoque de clientes através da criação de pedidos de entrega de produto nos sistemas ERP de seus fornecedores. Para realizar a gestão, o sistema SCIP se integra aos sistemas ERP envolvidos, trocando automaticamente as informações necessárias para o andamento do processo.

O sistema foi projetado de forma a ser altamente adaptável, uma vez que o mesmo deve intermediar diferentes tipos de sistemas ERP. A integração do SCIP com outros sistemas pode ser realizada utilizando EDI ou Internet. Caso um sistema ERP envolvido no processo não seja capaz de se comunicar utilizando estes meios, o sistema SCIP provê interfaces *Web* para inserção e extração manual de dados.

3.2 Comunicação via mensagens

Toda comunicação entre o sistema SCIP e sistemas externos, seja ela feita via EDI ou Internet, é realizada através de mensagens eletrônicas. Por se situar entre os sistemas ERP do fornecedor e do cliente (operando como um *broker*), todas as mensagens trocadas entre os mesmos são lidas e processadas pelo sistema SCIP. Apresenta-se a seguir a descrição das mensagens recebidas e enviadas pelo SCIP.

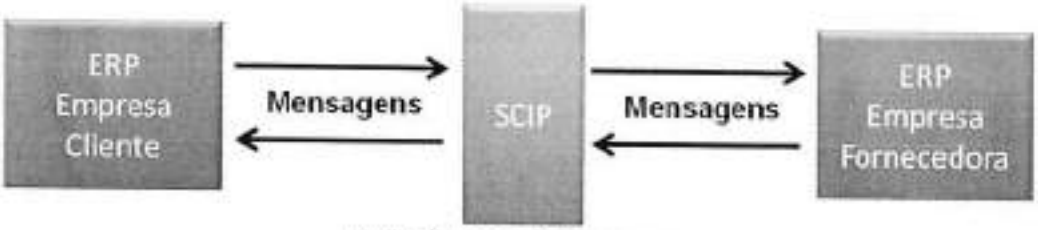


Figura 3: Troca de mensagens.

3.2.1 *Inventory Actual Usage*

Esta mensagem é enviada pelo sistema ERP do cliente, sendo destinada ao sistema ERP do fornecedor. Ela contém o nível de estoque atual de um produto no armazém do cliente. Todo cliente que participa de um processo VMI deve enviar uma mensagem por produto controlado. A freqüência de envio pode ser horária, diária ou sempre que ocorrer uma variação de estoque. O nível de estoque enviado na mensagem é utilizado pelo sistema SCIP para realizar a gestão de estoque.

3.2.2 *Demand Forecast*

Esta mensagem é enviada pelo sistema ERP do cliente, sendo destinada ao sistema ERP do fornecedor. Ela contém previsões de consumo diárias de um produto até um horizonte pré-definido. O horizonte da mensagem depende do tipo de produto e tipo de relação existente entre as empresas envolvidas, sendo normalmente definido por especialistas em *supply chain* da empresa cliente e fornecedora. A freqüência de envio depende da freqüência com que o cliente atualiza suas previsões de produção. Recomenda-se que o cliente envie uma mensagem *Demand Forecast* ao seu fornecedor sempre que ele realizar uma mudança em suas previsões. Em processos estáveis de VMI, esta mensagem é enviada diariamente. As previsões de consumo enviadas pelo cliente são utilizadas pelo sistema SCIP para realizar a gestão de estoque.

3.2.3 *Order Create*

Esta mensagem é enviada pelo sistema SCIP, sendo destinada ao sistema ERP do fornecedor. Ela é um pedido de criação de ordem de vendas e é enviada sempre que o sistema SCIP identifica a necessidade de aprovisionar o estoque de um cliente. A mensagem contém basicamente os seguintes itens:

- Identificação de produto encomendado;
- Data de entrega;
- Quantidade a ser entregue;
- Endereço de entrega.

3.2.4 *Order Response*

Esta mensagem é enviada pelo sistema ERP do fornecedor, sendo destinada ao sistema ERP do cliente. O envio desta mensagem está vinculado aos seguintes objetivos:

- Confirmar a criação de ordens de venda no sistema ERP do fornecedor;
- Comunicar ao cliente a criação de uma ordem de venda;
- Caso o fornecedor não seja capaz de cumprir as datas e quantidades pedidas na mensagem *Order Create*, a mensagem tem o objetivo de notificar ao sistema SCIP as novas datas e quantidades propostas pelo sistema ERP do fornecedor. Neste caso, o sistema SCIP deve avaliar as novas datas de entrega e realizar o necessário para manter o nível de estoque do cliente dentro dos níveis desejáveis.

3.2.5 *Ship Notice*

Esta mensagem é enviada pelo sistema ERP do fornecedor, sendo destinada ao sistema ERP do cliente. Esta mensagem tem como objetivo informar ao cliente que ocorreu um envio de mercadoria para entrega. Contém dados importantes para a entrada de mercadoria no cliente (data prevista de entrega, número de pedido, número de nota fiscal, impostos, entre outros).

A Figura 4 apresenta um exemplo de troca de mensagens entre as entidades envolvidas no processo VMI. Neste exemplo, o sistema SCIP identificou a necessidade de criar uma ordem de venda no sistema ERP do fornecedor a partir dos dados de estoque e previsão de consumo recebidos do cliente. Caso a mensagem *Order Response* enviada pelo fornecedor estivesse notificando uma mudança de data ou quantidade, o sistema SCIP poderia enviar mensagens *Order Create* adicionais para compensar as mudanças realizadas. Conseqüentemente, observar-se-ia um fluxo maior de mensagens *Order Create*, *Order Response* e *Ship Notice* envolvidas no processo.

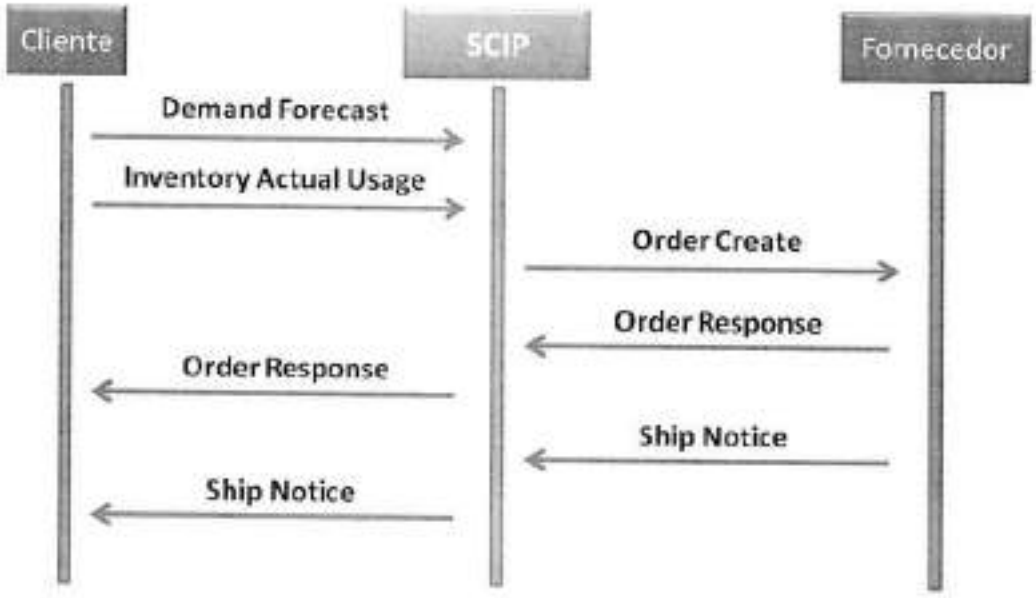


Figura 4: Troca de mensagens no tempo.

3.3 Modelo de negócio

Nas seções seguintes, é apresentado o modelo de negócio do sistema SCIP e suas variantes, abordadas no escopo deste projeto. Os modelos definem as partes e sistemas envolvidos e o fluxo de informação entre as entidades do modelo, contemplando as diferentes formas de aplicação do sistema SCIP.

3.3.1 Modelo de Negócio ERP/EDI

Este modelo de negócio envolve utilização do sistema SCIP conectado a mesma rede EDI do fornecedor e do cliente, onde ambos possuem seu próprio *software* ERP.

A Figura 5 mostra um exemplo de modelo de negócio que envolve apenas um fornecedor e um cliente. Entretanto, o sistema SCIP pode suportar diversos fornecedores e diversos clientes simultaneamente: Um fornecedor pode realizar serviços de VMI para diversos clientes distintos e o sistema SCIP é capaz de gerenciar diversos fornecedores.

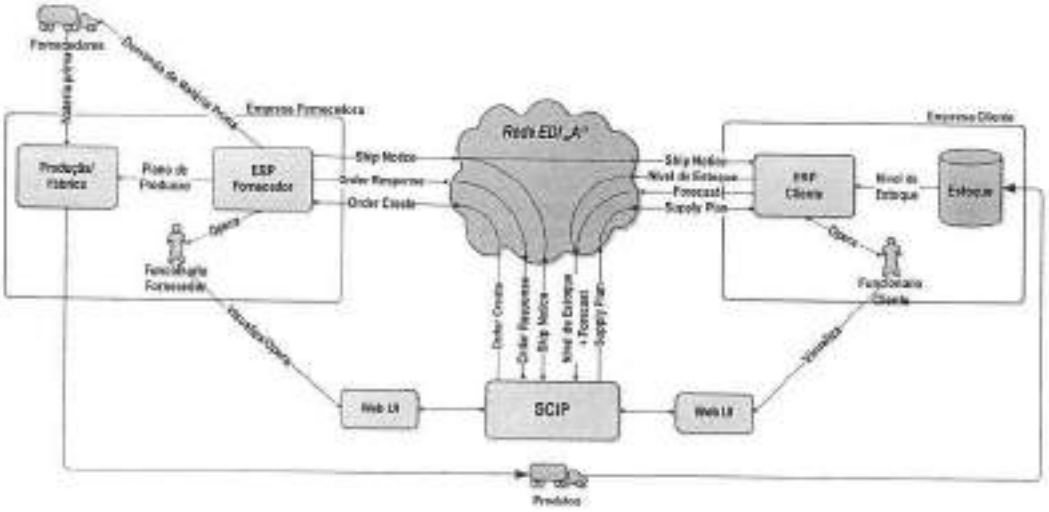


Figura 5: Modelo de Negócio - ERP + EDI.

Neste modelo, os dados de nível de estoque e previsão de consumo são enviados automaticamente pelo sistema ERP do cliente ao SCIP através da rede EDI. Estes dados são processados e o SCIP envia uma proposta de programação de entregas ao sistema ERP do fornecedor. Se as entregas propostas forem factíveis e o fornecedor dispuser de recursos para realizá-las, uma confirmação é enviada ao SCIP. Caso contrário, uma resposta é enviada propondo uma nova data de entrega e o sistema SCIP recalcula todas as propostas planejadas, de forma a adequar as programações de entrega.

Sempre que um produto é enviado do fornecedor para o cliente, uma mensagem de aviso de expedição (*Ship Notice*) é enviada pelo ERP do fornecedor para o SCIP e para o ERP do cliente.

Os funcionários autorizados da empresa fornecedora podem acessar o SCIP através de uma interface *Web* para configurar os parâmetros de seus produtos, aprovar proposições de programações de entrega (para serem enviadas ao sistema ERP do fornecedor), visualizar relatórios de histórico de entregas, entregas programadas e dados de clientes.

Os funcionários autorizados da empresa cliente podem acessar o SCIP através de uma interface *Web*, para verificar o histórico de entregas, níveis de estoque, entregas programadas e previsões de consumo atuais.

3.3.2 Variante 1: Cliente sem acesso a rede EDI

A primeira variante compreende o caso no qual o cliente não tem acesso à rede EDI do fornecedor, sendo a comunicação entre SCIP e cliente efetuada via Internet (Figura 6).

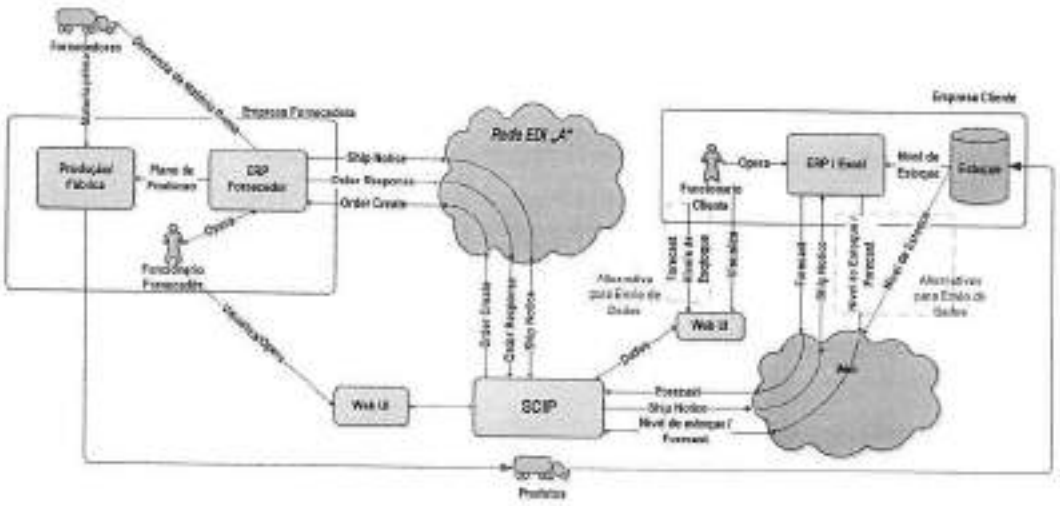


Figura 6: Modelo de Negócio - cliente sem acesso a rede EDI.

Para esta configuração, existem diversas alternativas de envio de dados por parte do cliente ao SCIP:

- Os dados de níveis de estoque podem ser enviados diretamente ao SCIP pela rede através de um sensor instalado no estoque do cliente (ex. Tanque);
- Podem ser enviados pelo ERP do cliente via Internet;
- Podem ser inseridos manualmente por um funcionário do cliente através da Interface *Web* do SCIP.

Da mesma forma, os dados de previsão de consumo (*forecast*) podem ser enviados através da interface *Web*, ou diretamente pelo sistema ERP do cliente.

O fluxo de informações entre SCIP e fornecedor permanece o mesmo.

3.3.3 Variante 2: Fornecedor e cliente sem rede EDI

Nesta variante, aborda-se o caso em que tanto o cliente quanto o fornecedor não dispõem de uma rede EDI, sendo toda a comunicação entre os sistemas realizada via Internet (Figura 7).

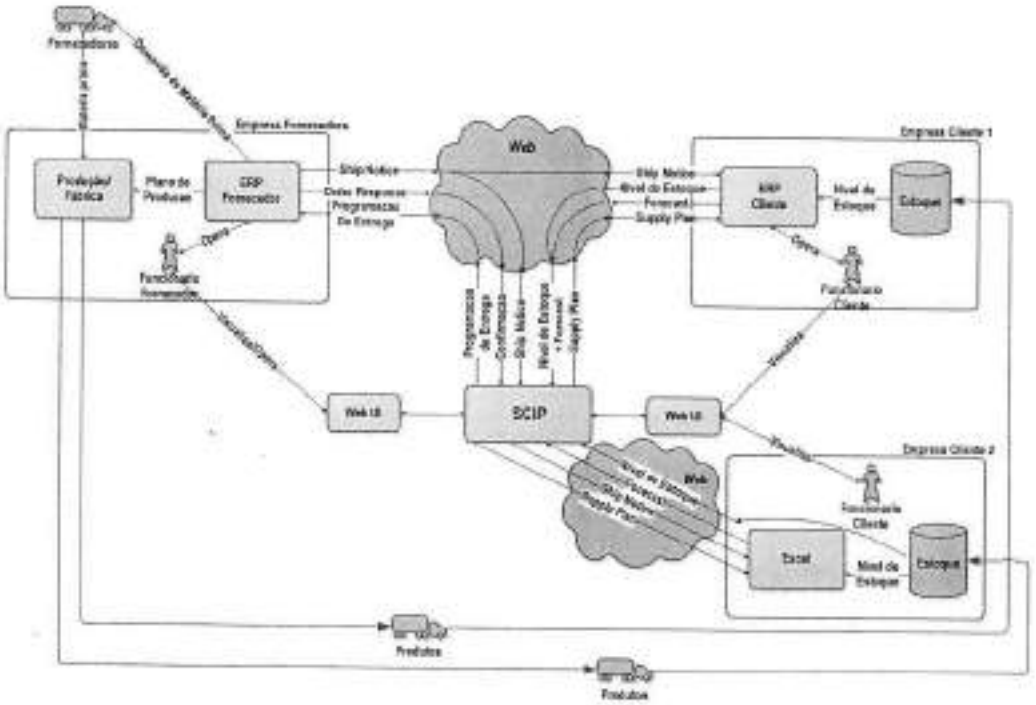


Figura 7: Modelo de Negócio - sem rede EDI.

Este modelo consolida os dois casos anteriores, substituindo a rede EDI central pela Internet. A empresa cliente 2 representa uma empresa que dispõe de um sistema ERP, enquanto a empresa cliente 3 representa uma empresa que não dispõe de um sistema ERP, sendo seu envio de dados de estoque e *forecast* realizado manualmente.

3.3.4 Variante 3: SCIP acoplado internamente ao sistema ERP do fornecedor

Nesta variante, o SCIP é administrado pela empresa fornecedora e se encontra conectado diretamente ao sistema ERP da mesma. Neste caso, a comunicação entre fornecedor e cliente prossegue da mesma forma descrita nas variantes descritas anteriormente. Esta variante compreende a total delegação do SCIP ao fornecedor, atuando assim como um módulo complementar ao sistema ERP existente (Figura 8).

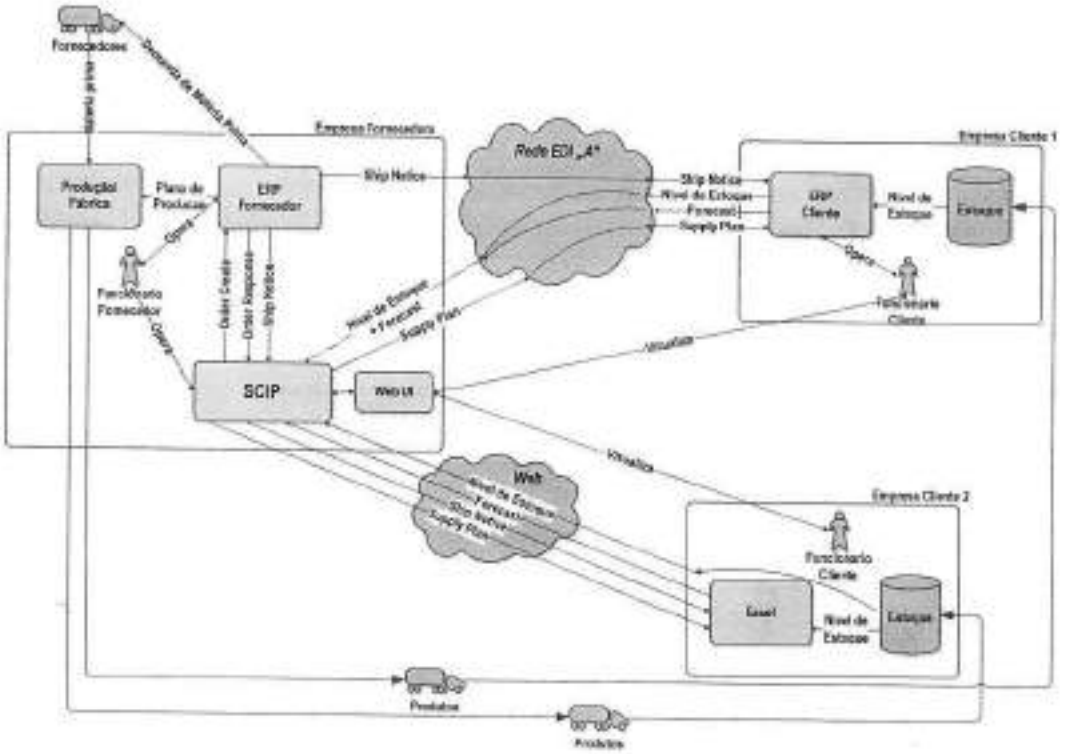


Figura 8: Modelo de Negócio - SCIP administrado pelo fornecedor.

3.3.5 Modelo de Negócio completo

O modelo de negócio completo, apresentado na Figura 9, consolida todas as variantes apresentadas anteriormente, adicionando ainda uma terceira possibilidade de conexão de um cliente – através de outra rede EDI vizinha (Figura 9).

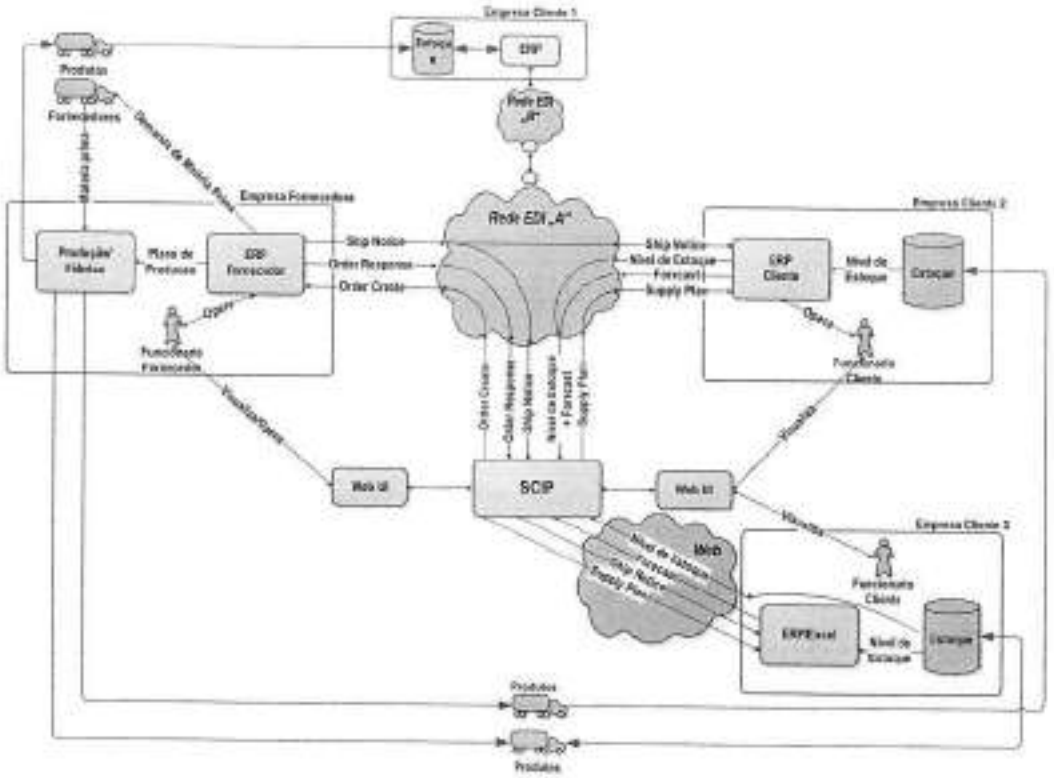


Figura 9: Modelo de Negócio completo.

3.4 Entidades de negócio envolvidas

Nesta seção, são apresentadas e descritas as entidades envolvidas no modelo de negócio do sistema SCIP. Todo processo VMI envolve um acordo de no mínimo duas empresas para o fornecimento de um ou mais produtos. Ao se iniciar um processo VMI, as empresas envolvidas redigem um contrato de fornecimento para cada produto a ser fornecido. Neste contrato são especificadas as empresas participantes, identificando seus papéis e responsabilidades. Dentre os papéis existentes neste processo, pode-se citar:

- Comprador (*BUYER*): Companhia que compra o produto;
- Fornecedor (*SUPPLIER*): Companhia que fabrica o produto e que realiza a venda;
- Recebedor de fatura (*BILL-TO*): Companhia que recebe a fatura referente à compra de produto. A companhia que recebe a fatura não necessariamente é a mesma que compra o produto;
- Pagador (*PAYER*): Companhia que paga a fatura referente à compra de produto. A companhia que paga a fatura não necessariamente é a mesma que compra o produto ou que recebe a fatura;
- Emissor de mercadoria (*SHIP-FROM*): Companhia que envia o produto fabricado pelo fornecedor. Não necessariamente é a mesma companhia que fabrica o produto (*SUPPLIER*). Em muitos casos o emissor de mercadoria é um centro de distribuição ou um armazém;
- Recebedor de mercadoria (*SHIP-TO*): Companhia que recebe o produto. O recebedor de mercadoria não necessariamente é o comprador, ou o recebedor de fatura ou o pagador.

Esta divisão em diversos papéis distintos reflete as diferentes formações jurídicas que as empresas participantes podem apresentar. Atualmente, a maior parte das grandes companhias é dividida em entidades jurídicas distintas que atuam em conjunto. Entretanto, publicamente, estas entidades se utilizam de um único nome fantasia para atuar no mercado. Este tipo de arquitetura jurídica é extremamente comum, especialmente em empresas que atuam em diversos países.

Estas divisões jurídicas são, na maior parte dos casos, contempladas nos sistemas ERP de fornecedores e clientes, sendo necessária a sua presença nas mensagens eletrônicas trocadas entre estes e o sistema SCIP. Desta forma, é de grande importância que o sistema SCIP contemple também estes conceitos. A forma utilizada para incorporá-los no sistema é mostrada na seção 3.5, e um exemplo fictício de aplicação do mesmo é mostrado na seção 3.7.

3.5 Itens de planejamento

O sistema SCIP contempla os contratos de fornecimento existentes entre fornecedores e clientes através dos chamados Itens de Planejamento. Cada Item de Planejamento corresponde a um contrato de fornecimento gerenciado pelo sistema SCIP. No caso mais simples, onde nenhuma exceção é contemplada, o Item de Planejamento consiste apenas de um produto que é vendido por um fornecedor e comprado por um cliente. Entretanto, fornecedores e clientes são empresas diferentes que gerenciam seus dados de formas distintas, assim o SCIP deve estar preparado para contemplar as variantes existentes no mercado.

Desta forma, o SCIP implementa o Item de Planejamento como um conjunto dos seguintes itens:

- Empresas: Comprador (*BUYER*), Fornecedor (*SUPPLIER*), Recebedor de fatura (*BILL-TO*), Pagador (*PAYER*), Emissor de mercadoria (*SHIP-FROM*), Recebedor de mercadoria (*SHIP-TO*);
- Produto comercializado;
- Regras de negócio necessárias para executar algoritmos de aprovisionamento do produto comercializado.

É importante ressaltar que clientes e fornecedores nem sempre utilizam um mesmo nome ou código para referenciar um produto em seu sistema interno. Desta forma o sistema SCIP deve armazenar as seguintes informações referentes ao produto de um Item de Planejamento:

- Nome do produto no fornecedor: Nome que o fornecedor designa para o produto vendido;
- Código de produto no fornecedor: Código que o fornecedor designa para o produto vendido;
- Descrição do Produto pelo fornecedor: Descrição do produto feita pelo fornecedor;
- Nome do produto no cliente: Nome que o cliente designa para o produto comprado;

- Código de produto no cliente: Código que o cliente designa para o produto comprado;
- Descrição do Produto pelo cliente: Descrição do produto feita pelo cliente.

Um detalhamento das regras de negócio contempladas pelo item de planejamento é descrito na seção 3.6 (Algoritmo de Aprovisionamento).

Os Itens de planejamento são os principais componentes do sistema SCIP para gerir os contratos de fornecimento entre fornecedores e clientes. É a partir deles que se definem as informações a serem reproduzidas nas mensagens eletrônicas, contendo todas as regras de negócio necessárias para realizar o planejamento de entregas.

3.6 Algoritmo de Aprovisionamento

A seguir será descrito o algoritmo de aprovisionamento de estoque responsável pelo planejamento de ordens de entrega entre fornecedor e cliente. O algoritmo contempla previsões de consumo e níveis de estoque do cliente juntamente com regras de negócio e restrições específicas.

3.6.1 Princípio

O algoritmo de aprovisionamento tem como principal objetivo gerar ordens de entrega a fim de manter o nível de estoque do cliente acima do estoque mínimo de segurança.

Através da análise do nível de estoque atual do cliente e da previsão de consumo, calcula-se o nível de estoque esperado para os dias subseqüentes (vide seção 2.5.3). Quando este for inferior ao estoque mínimo de segurança, uma ordem de entrega é gerada para o dia.

Para que uma ordem possa ser agendada para um dia específico, é preciso observar regras de negócio e restrições específicas da configuração fornecedor-cliente-produto. As principais restrições são:

- *Lead-Time*: o tempo necessário ao fornecedor para fabricar o produto e entregá-lo à localidade do cliente. Este deve ser inferior ao intervalo de tempo contemplado entre instante de geração da ordem e ponto de entrega definido;
- *Possibilidade de Entrega*: o ponto de entrega deve ser um dia contido dentro do calendário de recebimento do cliente e compatível com o calendário de entrega do fornecedor, descontando-se o tempo necessário para transporte do produto;
- *Ordens Confirmadas Adiadas*: quando o sistema ERP do fornecedor adia o ponto de entrega de uma ordem previamente planejada, é suposto que este não consegue realizar nenhuma entrega até a data confirmada, seja por deficiência na produção ou sobrecarga.

Caso as restrições citadas anteriormente impossibilitem a criação de uma ordem de entrega para o dia desejado, dias anteriores devem ser contemplados. Se ainda assim nenhum dia passível de entrega for identificado, a entrega deve ser agendada para uma data futura, ficando o estoque do cliente abaixo do nível de segurança pré-estabelecido. Nesta situação, está prevista a geração de alertas para ambas as partes envolvidas. Ressalta-se que esta ocorrência está relacionada a discrepâncias entre consumo real e previsão de consumo do cliente, inadequação dos parâmetros resultantes da configuração existente entre cliente e fornecedor ou eventos externos, tal como extravio de remessas.

3.6.2 Tipos de Aprovisionamento

Uma vez definido o ponto de entrega, define-se a quantidade do produto a ser entregue. Esta quantidade é influenciada pelo estilo de aprovisionamento logístico selecionado pelas empresas participantes:

- **Reposição Máxima de Estoque:**
  O estoque do cliente é reposto ao nível máximo permitido, minimizando assim a quantidade de entregas realizadas e maximizando a quantidade em cada entrega. Este aprovisionamento deve ser considerado quando os custos de transporte são muito altos comparados aos custos de se manter um grande estoque.

- **Reposição *Just-in-Time* (JIT):**

  O estoque do cliente é reposto à medida que este tem necessidade de material para produção, ficando o estoque constantemente próximo do nível de segurança. Nesta configuração, minimiza-se a quantidade de estoque mantida na localidade do cliente, conseqüentemente diminuindo custos associados, enquanto se aumenta a freqüência de entregas. Este é o algoritmo mais utilizado em relações VMOI (vide seção 2.5.1), onde o estoque é consignado. Em uma relação VMOI, o fornecedor deseja que o estoque no cliente seja o menor possível.

- **Dias de Autonomia:**

  O estoque do cliente é reposto de modo a permitir um determinado número de dias de autonomia de produção, sem uma nova entrega. A quantidade entregue depende do número de dias de autonomia desejada e da previsão de consumo para os dias contemplados. Nesta configuração, visa-se equilibrar quantidades de entrega e nível de estoque.

A quantidade planejada para entrega deve respeitar uma restrição adicional do fornecedor referente a quantidades mínimas de entrega e estoque máximo do cliente. Quando o cliente apresenta um pico de produção pode ser necessário agendar uma entrega que ultrapasse o estoque máximo do cliente, ficando a cargo deste definir se a situação pode ocorrer ou não.

3.6.3 Algoritmo de gestão de ordens

Uma vez que o algoritmo de aprovisionamento tenha planejado ordens de entrega, o algoritmo de gestão de ordens se responsabiliza por gerir o estado de cada ordem e disparar as mensagens necessárias para a comunicação com os sistemas ERP.

Uma ordem pode assumir os seguintes estados durante seu ciclo de vida:

- *Planned*: representa o estado da ordem planejada pelo algoritmo de aprovisionamento;

- *Approved*: quando uma ordem é planejada pelo sistema SCIP, esta deve ser primeiramente aprovada por um responsável do fornecedor. Se o fornecedor desejar, esta aprovação pode ser automática;
- *Waiting For Response*: uma vez aprovada, o sistema SCIP cria uma mensagem eletrônica contendo informações para criação de uma ordem no sistema ERP do fornecedor. O estado da ordem passa para *Waiting For Response*, até que se receba uma mensagem do fornecedor confirmando ou não a entrega programada;
- *Confirmed*: Quando o fornecedor confirma uma ordem de entrega a mesma passa para o estado Confirmed;
- *InTransit*: Quando o produto deixa a localidade do fornecedor, este envia uma mensagem de envio do produto. O estado da ordem é refletido para representar tal fato;
- *Delivered*: Quando o produto é entregue ao cliente, ou quando o dia da entrega corresponde ao dia atual, o estado da ordem é passado para Delivered, encerrando o ciclo de vida padrão de uma ordem;
- *Canceled*: Se por algum motivo, não for possível realizar a entrega, a ordem é cancelada.

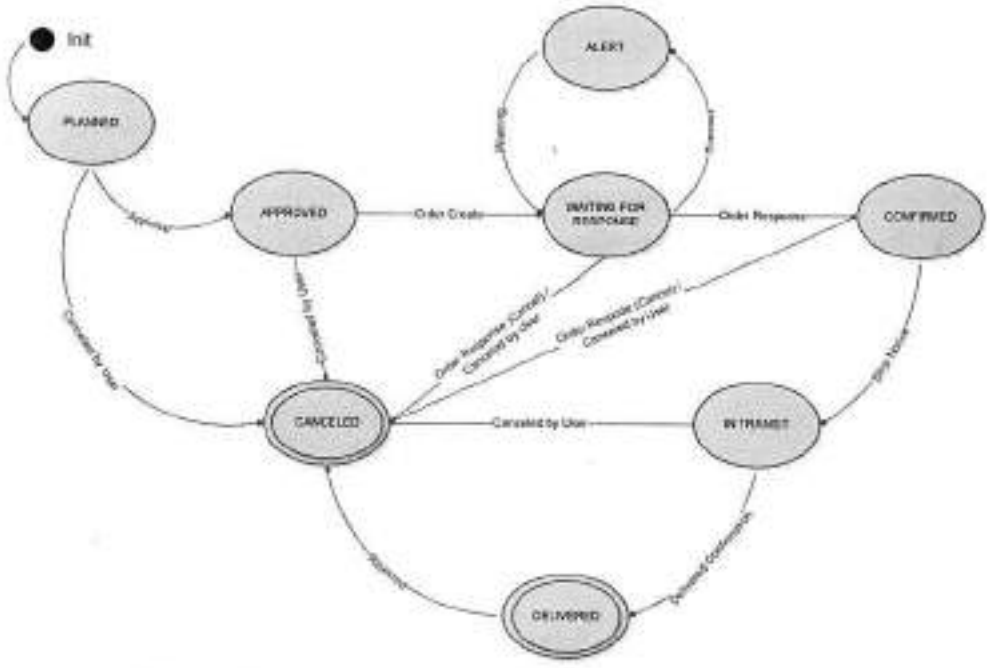


Figura 10: Diagrama de transição de estados de uma ordem.

A definição destes estados é crucial para que o sistema consiga realizar a gestão adequada das ordens e manter um histórico de cada ordem planejada no sistema.

3.7 Exemplo fictício de uma Configuração Cliente-Fornecedor

Nesta seção, é abordado um exemplo de aplicação de um Item de Planejamento no sistema SCIP. O exemplo tem como objetivo apresentar as relações existentes entre empresas que possuem um contrato VMI ou VMOI. Todas as empresas mencionadas neste item são fictícias.

3.7.1 Empresas envolvidas

A companhia fictícia Láctea Brasil vende pacotes de leite longa-vida para o supermercado fictício Mercado ABC. O Mercado ABC revende os pacotes de leite longa-vida no varejo em diversas cidades do Brasil.

3.7.1.1 *Empresa fornecedora*

A Láctea Brasil é atualmente a maior empresa brasileira produtora e envasadora de pacotes de leite longa-vida. Seu produto é reconhecido no mercado como um produto de qualidade e é líder nacional em vendas. Entretanto, a maior concorrente nacional da Láctea Brasil está com um plano de expansão agressivo no mercado e, para aumentar sua atuação, realizou a compra de diversos produtores de médio porte. A estratégia da concorrência tem se mostrado um sucesso, e a Láctea Brasil necessita pôr em prática um plano para manter sua posição no mercado.

A Láctea Brasil realizou um estudo interno e descobriu que uma das melhores alternativas para manter sua posição atual é melhorar sua integração com clientes e fornecedores. Isto permitirá a Láctea Brasil otimizar custos, melhorar seu serviço de atendimento a clientes e aumentar agilidade de processos que são realizados em conjunto com outras empresas. Desta forma, a Láctea Brasil deseja iniciar contratos VMI com todos os seus grandes clientes com o objetivo de automatizar as reposições de estoque de leite longa-vida. A Láctea Brasil realizou um estudo no mercado para verificar o interesse de seus clientes e obteve uma boa aceitação.

A Láctea Brasil possui sua sede na cidade de São Paulo. A maior parte da produção e envasamento é realizada na cidade de Frutal em Minas Gerais. A Láctea Brasil não possui centros de distribuição espalhados pelo país. Todas as entregas a clientes são feitas diretamente da fábrica.

3.7.1.2 *Empresa cliente*

Atualmente o Mercado ABC é uma das maiores redes de supermercado do Brasil e um dos maiores compradores de leite longa-vida da Láctea Brasil.

O Mercado ABC possui um escritório administrativo no Rio Grande do Sul. Todas as compras realizadas pela empresa no Sul e Sudeste brasileiro são feitas por agentes que se encontram neste escritório. Entretanto, o centro de custos da empresa se encontra na cidade de São Paulo. A empresa deseja que seus fornecedores enviem suas faturas para o centro de custos do Mercado ABC. Todas as faturas são analisadas pelo centro de custos e, após análise, são enviadas para os escritórios administrativos responsáveis pela compra. Estes, por sua vez, realizam os pagamentos aos fornecedores.

3.7.2 Cenário de integração

A Láctea Brasil e o Mercado ABC desejam realizar um projeto piloto VMI para automatizar o aprovisionamento de leite longa-vida no principal mercado da rede, localizado no centro da cidade de São Paulo.

Ambas as empresas desejam que o processo VMI seja o mais automático possível e que seja implantado de forma rápida. Tanto a Láctea Brasil quanto o Mercado ABC possuem sistemas ERP que estão conectados a redes EDI distintas que não possuem uma ligação.

Para possibilitar o processo, a Láctea Brasil deseja utilizar os serviços do sistema SCIP. O SCIP irá realizar todos os cálculos de aprovisionamento e criará automaticamente programações de entrega no sistema ERP da Láctea Brasil para aprovisionar o estoque do Mercado ABC localizado no centro da cidade de São

Paulo. O sistema SCIP atuará ainda integrando as redes EDI de ambas as empresas, uma vez que as mesmas não estão conectadas.

3.7.2.1 *Item de planejamento*

O leite longa-vida vendido pela Láctea Brasil possui a seguinte descrição interna no sistema ERP da empresa:

> **Nome do produto:** Leite Longa-Vida – 1L – A – Minas.
> **Código interno:** 01LLVAFrut.
> **Descrição:** Pacote de Leite longa-vida tipo A produzido em Minas Gerais.

O mesmo produto possui a seguinte descrição interna no sistema ERP do Mercado ABC:

> **Nome do produto:** Leite Longa-Vida Láctea Brasil – 1L.
> **Código interno:** 51396.
> **Descrição:** Pacote de leite longa-vida Láctea Brasil – tipo A.

As informações referentes ao produto são cadastradas de forma inteiramente diferente em cada sistema ERP. Desta forma, o SCIP realizará a correlação entre o produto do fornecedor e produto do comprador.

Para iniciar o projeto, um Item de Planejamento é criado no sistema SCIP para representar o contrato VMI em questão:

- Comprador (*BUYER*): Companhia Mercado ABC – Rio Grande do Sul.
- Fornecedor (*SUPPLIER*): Láctea Brasil LTDA – São Paulo.
- Recebedor de fatura *(BILL-TO)*: Companhia Mercado ABC – Centro de Custos – São Paulo.
- Pagador (*PAYER*): Companhia Mercado ABC – Rio Grande do Sul.
- Emissor de mercadoria (*SHIP-FROM*): Láctea Brasil – Centro de Produção – Frutal – Minas Gerais.

- Recebedor de mercadoria (*SHIP-TO*): Hiper Mercado ABC – Centro de São Paulo.
- Produto (Fornecedor):
    - Nome do produto: Leite Longa-Vida – 1L – A – Minas.
    - Código interno: 01LLVAFrut.
    - Descrição: Pacote de Leite longa-vida tipo A produzido em Minas Gerais.
- Produto (Comprador):
    - Nome do produto: Leite Longa-Vida Láctea Brasil – 1L.
    - Código interno: 51396.
    - Descrição: Pacote de leite longa-vida Láctea Brasil – tipo A.

3.7.2.2 *Processo de integração*

Para possibilitar o processo, diariamente o Mercado ABC enviará via EDI as seguintes mensagens eletrônicas:

- *Inventory Actual Usage*: Estoque atual do produto 51396 no mercado localizado no centro de São Paulo;
- *Demand Forecast*: Previsões de vendas diárias do produto 51396 para os próximos 45 dias.

O sistema SCIP utilizará as informações presentes nas mensagens apresentadas anteriormente para executar algoritmos de aprovisionamento e definir datas e quantidades de entrega do leite longa-vida no mercado de destino. Como resultado, o sistema SCIP realizará a criação de programações de entrega no sistema ERP da Láctea Brasil. Toda sincronização de dados entre o sistema SCIP e Láctea Brasil será realizado através da troca de mensagens *Order Create* e *Order Response*. O sistema ERP da Láctea Brasil enviará mensagens *Ship Notice* sempre que for realizada a saída de mercadoria para ser entregue no mercado de destino. A Figura 11 mostra o cenário de integração das empresas realizado pelo sistema SCIP.

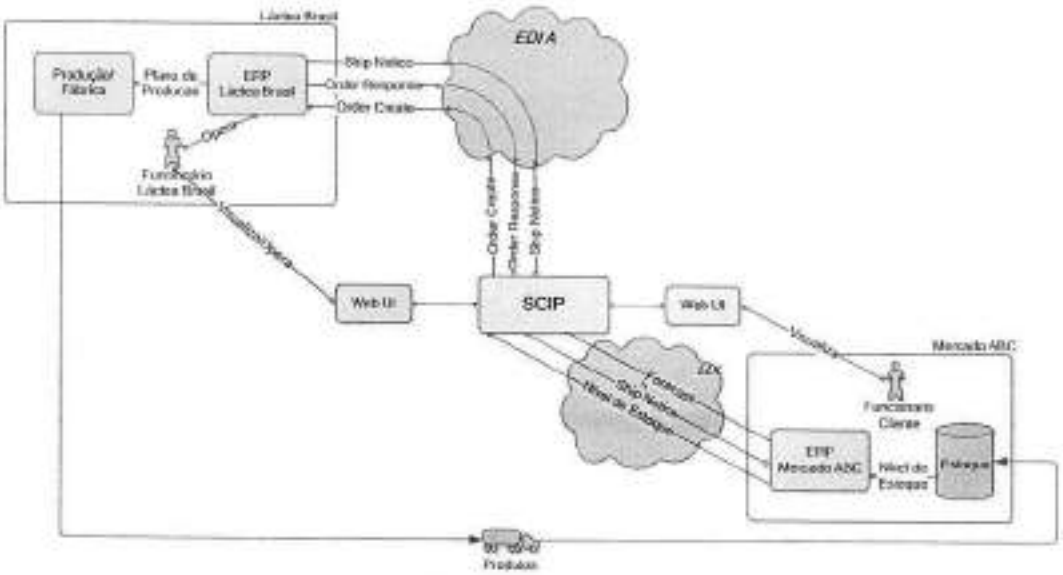


Figura 11: Cenário de Integração.

4 Conceitos e Ferramentas de Engenharia de *Software*

Neste capitulo são apresentados os conceitos e ferramentas de engenharia de *software* utilizados e aplicados nas fases de concepção e desenvolvimento do projeto. Um conjunto de técnicas e metodologias foi aplicado para permitir uma análise de requisitos eficaz e uma modelagem concisa, flexível e escalável.

O sistema proposto para desenvolvimento neste projeto desempenha um papel de integração B2B entre sistemas corporativos, através de redes privadas ou Internet, atuando como mediador e permitindo transparência nas negociações entre clientes e fornecedores dentro da cadeia de suprimentos. Desta forma, um foco especial foi dado à modularidade e flexibilidade da arquitetura do sistema, projetado para ser reutilizável e adaptável a novas especificações. Fez-se necessário, para isso, empregar o estado da arte em termos de tecnologia e desenvolvimento de *software*.

4.1 Arquitetura

Nesta seção são apresentados aspectos de engenharia de *software* contemplados durante os processos de modelagem e desenvolvimento, que influenciaram de maneira importante nos resultados alcançados.

4.1.1 Arquitetura distribuída

Projetar aplicações distribuídas não é uma tarefa simples. Muitas decisões devem ser tomadas no que tange aos aspectos arquiteturais, metodológicos e de implementação das mesmas, sendo tais decisões de impacto importante sobre as capacidades da aplicação em desenvolvimento – segurança, escalabilidade, disponibilidade e manutenibilidade são apenas algumas delas – e sobre a arquitetura, projeto e implementação da infra-estrutura em questão.

4.1.2 Objetivos da arquitetura escolhida

Entender as decisões a serem tomadas no projeto de cada camada é um passo de vital importância para se modelar eficientemente o sistema. De antemão, faz-se

necessário compreender que o projeto de uma aplicação envolve a escolha sobre a arquitetura lógica e física da mesma, bem como das tecnologias e da infra-estrutura a serem empregadas para implementar suas funcionalidades. Para realizar tais escolhas de maneira eficiente, é necessário que se tenha conhecimento profundo dos processos de negócio que a aplicação irá desempenhar (seus requisitos funcionais) e dos níveis de escalabilidade, disponibilidade, segurança e manutenibilidade requeridos (seus requisitos não funcionais).

Objetiva-se, com isso:

- Prover uma solução bem adaptada ao modelo de negócio;
- Considerar questões de segurança desde o princípio da concepção;
- Prover alto desempenho e otimização de operações comuns através das aplicações de padrões de desenvolvimento;
- Prover disponibilidade;
- Prover escalabilidade de encontro com a demanda esperada, aplicando um mínimo de recursos para se atender o maior número de usuários e atividades;
- Considerar aspectos de gerenciabilidade do sistema, permitindo-se aos operadores monitorar e corrigir eventuais falhas, dependendo do cenário;
- Prover manutenibilidade, tendo cada funcionalidade uma localização intuitiva e determinada, lançando-se mãos de padrões de desenvolvimento bem aceitos;

4.1.3 Componentização

Um exame de muitas das soluções de arquitetura encontradas na prática revela alguns componentes comumente empregados.[1] A Figura 12 mostra os componentes criados para o sistema SCIP.

Observa-se que o termo componente é usado no sentido de uma parte da solução. Isso inclui componentes de *software* compilados (bibliotecas) e outros artefatos de *software* como páginas *web* e *schedulers*.

[1] NILSSON, 2006; JEZIERSKI, 2002; McCAFFERTY (2006)

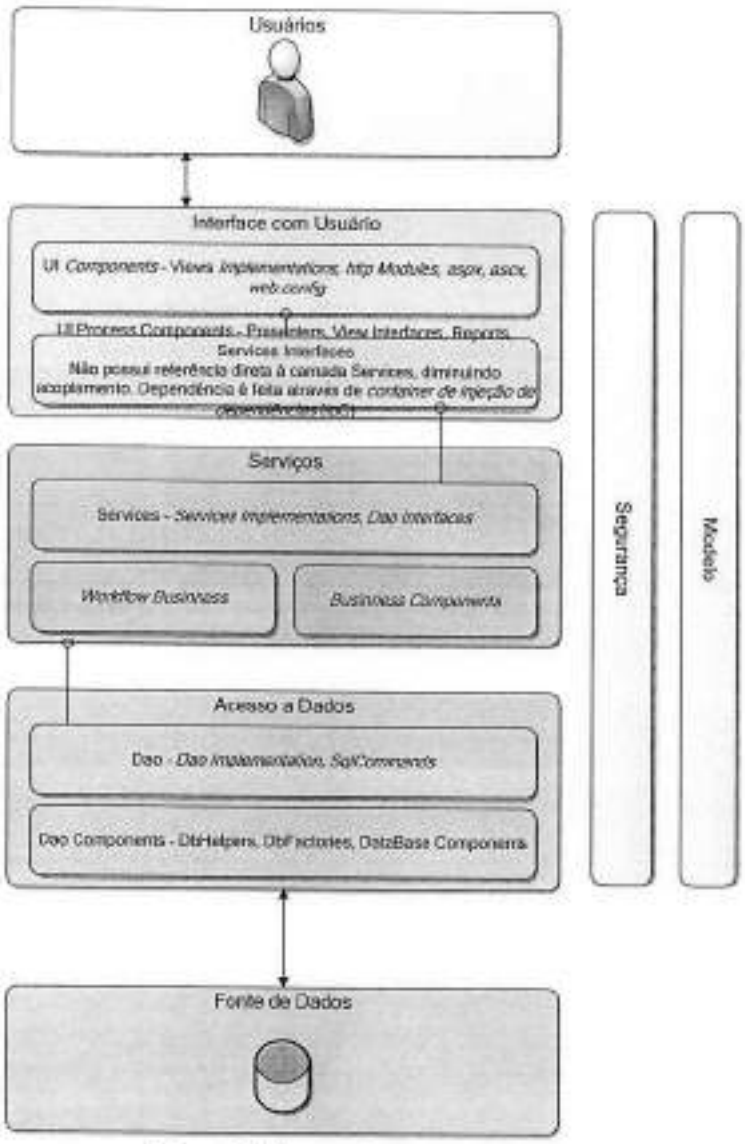


Figura 12: Visão arquitetural da aplicação.

Os componentes identificados no sistema em desenvolvimento são:

Interface com usuário

- *User interface (UI) components:* Provê um modo para que usuários interajam com a aplicação. Por exemplo, página de sinóptico de visualização permite ao usuário visualizar ordens programadas e as confirmarem manualmente. A

interface com usuário é realizada através de páginas ASP.NET, no caso do sistema SCIP.

- *User process components:* Em muitos casos, a interação com usuário segue um dado processo. A fim de orquestrar estas interações, é útil dirigir este processo através de componentes separados, fazendo com que o fluxo de processo e o gerenciamento de estado na interface com usuário não seja *hard-coded* na própria interface, e a mesma máquina de coordenação da interface possa ser reutilizada para diversos tipos de interface com usuário (*web, mobile, window forms*). Neste projeto, esta orquestração foi viabilizada pela aplicação do padrão MVP (do inglês, *Model View Presenter*) de acordo com a definição de Larman (2002), ilustrado na Figura 13.

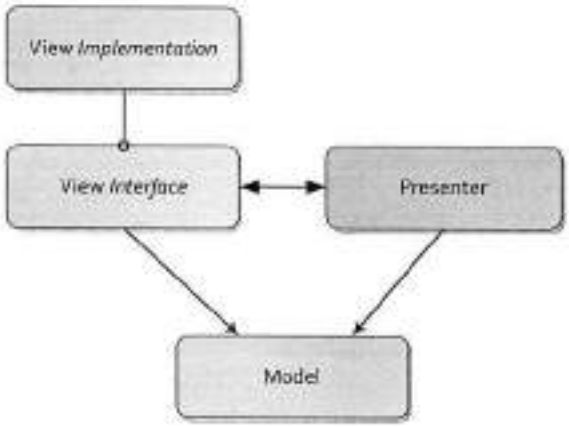


Figura 13: Ilustração do *Model View Presenter* (MVP).

O *Presenter* constitui a máquina que gerencia a interface com usuário (*View Implementation*) através da *View Interface*. Esta garante o desacoplamento entre lógica de processamento de interface e lógica de apresentação, permitindo múltiplas implementações desta última, bastando, para isso, substituir a implementação da interface (*View Implementation*).

Serviços

- *Services e Business workflows:* Depois que os dados são coletados através dos processos da interface com usuário, eles são processados pelos processos de negócio. Muitos processos de negócio envolvem múltiplos passos que devem ser ordenados em ordem correta e orquestrados. Tais

processos devem ser gerenciados, dado que podem tomar uma considerável janela de tempo para serem executados.

- *Business components*: Em despeito de um processo de negócio consistir de um fluxo de processos ou em um único processo, a aplicação demanda componentes que implementem as regras e executem as tarefas de negócio. Componentes de negócio implementam a lógica da aplicação.

Acesso a dados

- *Data Access:* Contém a implementação que transforma as estruturas de dados proprietários da linguagem (*DataTable*, *DataReader*, *Xml*, etc.) em objetos do modelo de negócio (Produto, Empresa, Ordem, etc.).
- *Data Access Components:* Durante um processo de negócio, provavelmente será necessário se acessar dados armazenados pela aplicação. Faz sentido, então, abstrair a lógica necessária para se acessar os dados contidos na fonte de dados do sistema.

Modelo

- O modelo compreende as estruturas de dados que representam entidades reais do modelo de negócio (Produto, Empresa, Ordem, etc.), que são utilizados pelas outras camadas e componentes para concentrar informações de uso comum. Isto permite que os dados armazenados em base de dados sejam passados entre componentes de forma encapsulada, ao invés de se lançar mão de estruturas proprietárias da linguagem de programação (*DataTable*, *String*, etc.)..

Segurança

- Provê gerenciamento do tratamento de exceções da aplicação tratando corretamente os erros e exceções gerados. Além disto, gerência o nível e permissão de acesso de usuários e garante a segurança dos componentes, evitando que estes gerem erros devido a dados de entrada incorretos.

Testes Unitários

- Garante testabilidade do sistema e cobertura das partes funcionais do sistema. Testabilidade é de suma relevância no desenvolvimento de qualquer sistema, garantindo a aplicação de práticas como desenvolvimento orientado a testes, explicado mais adiante neste trabalho. Este módulo contém os módulos de testes utilizados para testar a funcionalidade de cada componente, assim como os dados fictícios para cada teste.

4.2 Linha de Produto de *Software*[1]

Linha de Produtos de *Software* (do termo em inglês *Software Product Line* - SPL) se baseia no uso de técnicas de engenharia para criar um grupo de sistemas a partir de um conjunto de especificações de *software* compartilhados, usando um meio de produção comum (GOMAA, 2004).

A técnica de linha de produção vem sendo aplicada analogamente em outras áreas de engenharia há décadas para criar linhas de produtos semelhantes a partir de uma mesma fábrica ou linha de montagem. Como exemplo, cita-se o caso das automotivas que conseguem construir dezenas de milhares de variantes de modelos a partir de um conjunto de peças meticulosamente arquitetadas e uma fábrica especializada para configurar e montá-las.

O conceito de reutilização de código na manufatura de *software* é um conceito antigo, porém seu sucesso era elusivo até pouco tempo. No entanto, recentemente, a aplicação destes conceitos de modo estratégico tem produzido melhorias na qualidade, escalabilidade, custo e tempo de desenvolvimento de sistemas de uma ordem comparável ao advento da produção em massa e customização em massa observada na manufatura.

[1] A modelagem dos processos do sistema desenvolvido foi guiada por este princípio, tendo como referência a obra de Gomaa (2004).

A produção em massa aplicada ao conceito de *software* é trivial, dado que criar múltiplas cópias de um *software* é extremamente fácil. Por outro lado, customização em massa – criar diversas variantes de um mesmo produto de forma eficiente – é um grande avanço na área de engenharia de *software*. A chave para o sucesso da customização em massa é capitalização das semelhanças e gestão de variações da linha de produtos.

O SPL (KRUEGER, 2006) envolve quatro conceitos principais (vide Figura 14):

- *Especificação de software*: uma coleção de especificações, tal como requisitos, código fonte, componentes, arquiteturas e documentações que podem ser configurados e compostos de formas diferentes para criar um novo produto. Para acomodar variações, cada uma destas especificações deve ter um papel bem delineado, podendo ser opcional ou ter pontos de variação internos que podem ser configurados de modo a prover comportamentos diferentes;
- *Modelo de decisão*: descreve características opcionais e variáveis do produto, sendo cada um definido unicamente pelas decisões tomadas para cada opcional e funcionalidade no modelo de decisão;
- *Processo de Produção*: o meio pelo qual os produtos são compostos e configurados, utilizando resultado do modelo de decisão para guiar a configuração do produto a partir das especificações disponíveis;
- *Produtos de Software*: o conjunto de produtos que podem ser disponibilizados pela linha de produção, que define o seu escopo. (Gomaa)

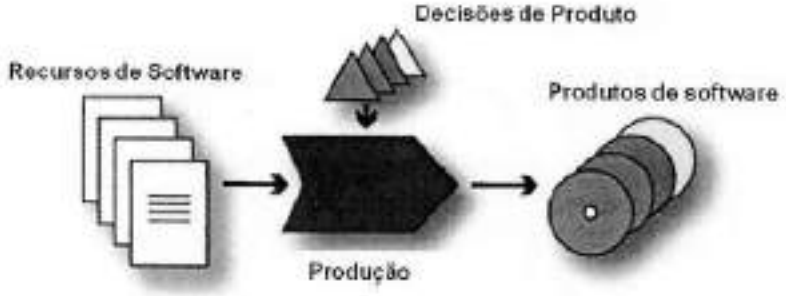


Figura 14: Conceito básico de SPL

A criação de uma estrutura para desenvolver uma linha de produtos não está contida no escopo deste trabalho, mas aplica-se o conceito de SPL na modelagem do *software*, incorporando pontos de variação aos diagramas UML, que permitem configurar diferentes funcionalidade e opcionais. Desta forma, possibilita-se que o sistema final seja posto em produção de diferentes maneiras (configurando produtos diferentes) e possa ser facilmente expandido com novas funcionalidades. A adoção de tal abordagem resulta ultimamente em um sistema mais modular e flexível a mudanças. Uma vez estabelecidas as partes fundamentais para uma linha de produto de *software*, pode-se futuramente desenvolver os conceitos de nível mais alto para uma linha de produto.

4.3 Injeção de Dependências

A injeção de dependência, segundo Fowler (2004), é um tipo específico de inversão de controle que permite atingir um alto grau de abstração entre os componentes do sistema. A seguir são explicados os conceitos necessários que levam à definição de injeção de dependência.

4.3.1 Inversão de controle

Inversão de Controle (IoC, do inglês *Inversion of Control*) é um conceito no qual o fluxo de controle é invertido em relação ao modelo de interação tradicional imperativo. Em vez de se especificar a série de eventos através de métodos durante a existência de um programa, registra-se a resposta desejada para um determinado evento delegando o controle a uma entidade externa que decide a ordem exata dos eventos.

Na programação tradicional, o programa se torna inflexível, à medida que cada cenário diferente de eventos requer uma implementação separada. Com inversão de controle, podem ser criados infinitos cenários, ganhando-se em flexibilidade e diminuindo-se o acoplamento entre os componentes da aplicação, porém com o custo de maior complexidade e perda do determinismo na programação. Aplicando IoC, torna-se difícil observar quais métodos serão chamados através da inspeção do código. É preciso considerar o código e o ambiente onde este está sendo executado

para inferir os resultados e determinar a ordem dos eventos. O código é mais difícil de ser escrito e entendido.

Quando o programa se baseia no recebimento de um parâmetro para realizar um processamento e fornecer uma resposta, sua implementação é relativamente simples. Já para programas que são executados por muito tempo e quando não é possível prever o curso dos eventos, a inversão de controle pode tornar-se bastante útil. Entre outros exemplos de aplicação estão interfaces gráficas, servidores de rede, serviços que tratam eventos, etc.

4.3.2 Programação dirigida por eventos

IoC traça um paralelo à programação dirigida por eventos, a qual se baseia no conceito em que o fluxo do programa é determinado pelas ações do usuário ou mensagens de outros programas. Cada trecho de código responsável por tratar um evento é denominado *event handler*. Muitos *frameworks* realizam sua própria implementação e esperam que o usuário proveja apenas o código para os tratadores de eventos. Cada *event handler* tem uma rotina/método associado que gera respostas para eventos aos quais o programa principal responde.

4.3.3 Princípio de inversão de controle

Geralmente é visto como sinônimo de inversão de controle. O princípio visa inverter a noção de que módulos de alto nível de um *software* dependam de módulos de um nível abaixo, pregando que módulos não devem depender uns dos outros, mas sim de abstrações. Assim, um módulo de baixo nível não precisa ser desenvolvido antes do módulo de mais alto nível. Desta forma é possível tomar uma abordagem *top-down* nas fases de concepção e desenvolvimento do projeto. O resultado consiste em objetos com baixo nível de acoplamento, modulares e maior separação de responsabilidades.

Desta forma, os objetos de alto nível podem ser testados através de *Mock Objects*[1] em lugar dos serviços de nível mais baixo e dado que existe desacoplamento de funcionalidade entre componentes de camadas diferentes, a inversão de dependência torna-se complementar à estratégia de teste unitário.

4.3.4 Injeção de dependências

Injeção de Dependências é uma técnica especifica da inversão de controle. Caracteriza-se pela abstração onde a implementação de uma classe é realizada por outra – a classe injetada. Tipicamente, existem diversas variantes da classe injetada (subclasses). A classe principal abstrai o código comum requerido por todas as implementações e delega às classes injetadas o comportamento específico.

Da mesma forma que num programa com IoC, onde o controle da execução é delegado à uma entidade auxiliar e tem-se a orientação a eventos, uma classe que utiliza Injeção de Dependência delega o controle sobre uma implementação à classe injetada, responsável pela realização do trabalho específico.

Ao construir estruturas de dados com Injeção de Dependência, consegue-se independência da implementação especifica atingindo um desacoplamento da estrutura, sendo a classe injetada intercambiável por outra implementação.

A classe abstrata apenas especifica as operações que devem estar disponíveis, mas não provê uma implementação, deixando esta tarefa às classes injetadas, dando liberdade para escolha da implementação. A classe injetada não precisa se preocupar com a implementação da classe abstrata.

4.4 Padrões de Projeto

Padrões de Projeto (em Inglês, *Design Patterns*) são soluções recorrentes a problemas de engenharia de *software* que aparecem freqüentemente em aplicações reais. *Patterns* se preocupam com o estabelecimento de uma plataforma elegante e reutilizável para soluções de programação comuns.

---

[1] Vide seção 4.4.4

No atual projeto, utilizou-se a técnica chamada *.NET optimized*, utilizando-se padrões que exploram recursos disponíveis no .NET 2.0 tais como *Generics*. Os padrões são baseados nos definidos pelo GoF[1], os quais podem ser divididos em três categorias básicas: padrões criacionais, estruturais e comportamentais. Os padrões criacionais são relacionados à criação de objetos, os estruturais tratam das associações entre classes e objetos e os comportamentais, das interações e divisões de responsabilidades entre as classes ou objetos (vide Tabela 4). Os padrões de projeto utilizados no desenvolvimento do sistema serão descritos a seguir.

Tabela 4: Resumo dos padrões de desenvolvimento definidos pelo GoF.

Creational Patterns	
Abstract Factory	Cria uma instância de um conjunto de famílias de classes
Builder	Separa construção de objeto de sua representação
Factory Method	Cria uma instância de uma série de classes derivadas
Prototype	Uma instância completa para ser copiada ou clonada
Singleton	Classe que só pode ter uma instância
Structural Patterns	
Adapter	Adequa interaces de classes diferentes
Bridge	Separa a interface de um objeto de sua implementação
Composite	Estrutura em árvore de objetos simples e compostos
Decorator	Adiciona responsabilidades para objetos dinamicamente
Façade	Uma classe única que representa um subsistema inteiro
Flyweight	Uma instância usada para compartilhamento eficiente
Proxy	Um objeto representando um outro objeto
Behavioral Patterns	
Chain of Resp.	Um modo de passar uma requisição através de uma cadeia de objetos
Command	Encapsula uma requisição de comando como um objeto
Interpreter	Um modo de incluir elementos de linguagem num programa
Iterator	Acesso sequencial de elementos de uma coleção
Mediator	Define uma comunicação simplificada entre classes
Memento	Captura e restaura o estado interno de um objeto
Observer	Modo de notificar mudanças de um dado número de classes
State	Altera o comportamento de um objeto quando seu estado muda
Strategy	Encapsula um algoritmo dentro de uma classe
Template Method	Delega passos de um algoritmo para uma subclasse
Visitor	Define uma nova operação para uma classe sem alterações

4.4.1 *Adapter*

Um *Adapter* é uma classe que permite converter a interface de uma classe em outra interface, a fim de criar uma camada de abstração que intermedia e relaciona componentes em um sistema.

[1] Do Livro *Design Patterns – Elements of Reusable Object-Oriented Software*; *Gang-of-Four*, por causa dos quatro autores que o escreveram: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.

Com este recurso é possível que classes de bibliotecas, implementações e *frameworks* diferentes possam ser utilizadas conjuntamente sem a necessidade de interferir ou estender as classes originais. Desta forma, o *Adapter* permite que a utilização das funções de uma interface sejam desacopladas de suas implementações, ao mesmo tempo que não dependem de sua estrutura interna.

Quando um cliente realiza uma chamada a um método disponibilizado pelo *Adapter*, este redireciona a chamada ao objeto alvo, que pode ser implementado de diversas formas. Tipicamente a técnica é aplicada através do uso de herança ou agregação.

Existem outros dois padrões de desenvolvimento que tangem o aspecto de interfaces entre classes: *Façades* e *bridges*.

- *Bridge* é uma técnica ex-ante que permite que a abstração e implementação variem independentemente uma da outra, enquanto um *Adapter* é uma adaptação ex-post para classes não relacionadas.

- *Façade* provê uma interface unificada para um subsistema ou diversas outras interfaces, definindo um nível hierárquico de interface mais alto, facilitando a utilização do subsistema. Este padrão pode ser aplicado para facilitar o acesso a bibliotecas de *software*, reduzir dependências de código externo quanto à implementação interna da biblioteca, e para permitir maior flexibilidade.

Um *Adapter* é usado quando se precisa respeitar uma interface particular existente e fornecer polimorfismo, provendo uma interface diferente ao objeto. Já um *Façade* define uma interface inteiramente nova.

Na Figura 15 apresenta-se um exemplo genérico para um *Adapter*. A classe *Target* provê uma interface específica para o cliente e a classe *Adapter* adapta a interface de *Adapter* para ser compatível com *Target*.

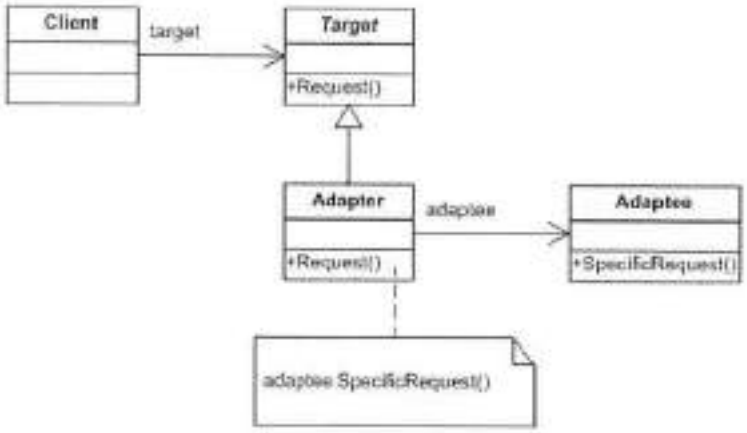


Figura 15: Diagrama de Classes exemplificado para um Adapter[1].

No projeto, utilizou-se este padrão para se obter maior flexibilidade quanto à implementação, de forma que o sistema possa facilmente ser estendido no futuro. O desacomplamento entre a interface e implementação permite que a implementação seja trocada facilmente por outra, garantindo grande modularidade ao projeto.

Um caso específico de utilização deste padrão no desenvolvimento do sistema SCIP, que foi de especial valor para se superar um desafio de implementação, consiste na adaptação dos objetos do modelo para se estender o uso de alguns controles servidor (*server controls* da plataforma .Net), os quais possuem interfaces padrões, próprios da plataforma proprietária.

4.4.2 *Decorator*

Decorators permitem adicionar responsabilidades a um objeto dinamicamente, fornecendo uma alternativa flexível à extensão de uma classe para adicionar funcionalidades. Tal técnica é adequada quando não se deseja criar uma estrutura de herança na qual o objeto fica preso a classe, mas encapsulá-lo em outro objeto (o *Decorator*) dinamicamente adicionando-se funcionalidades. O *Decorator* repassa todas as chamadas de função ao objeto encapsulado e adiciona as funcionalidades nele definidas de forma transparente.

[1] Fonte: GoF – www.dofactory.com

A Figura 16 exemplifica o uso de um *Decorator*. Na figura, o *Component* define a interface para um objeto e *ConcreteProduct* define um objeto que pode ter sua funcionalidade estendida. A classe *Decorator* mantém uma referência para componente e define uma interface conforme para ele. Por último, o *ConcreteDecorator* adiciona funcionalidades e responsabilidades ao componente.

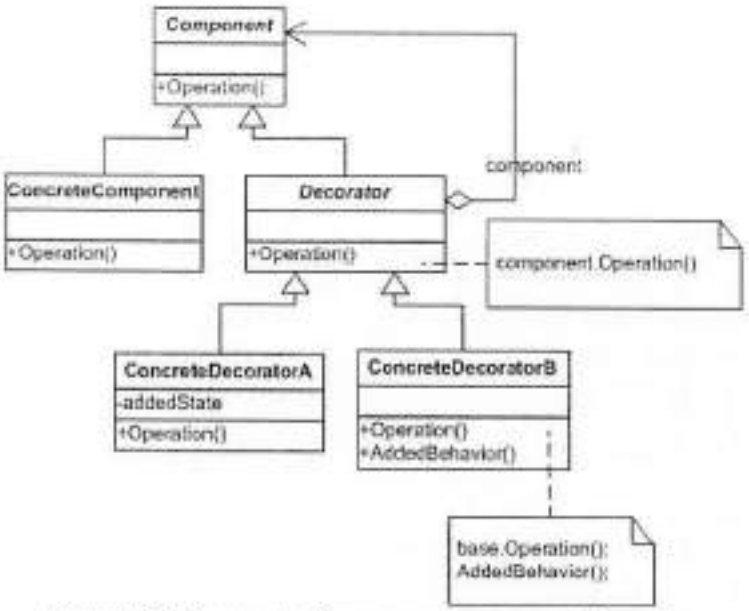


Figura 16: Diagrama de Classes genérico para um *Decorator*.

4.4.3 *Factory*

Factory permite modelar uma interface para criar um objeto, delegando a subclasses a decisão sobre qual classe deve ser instanciada no momento da criação. O termo *Factory* advém justamente de sua responsabilidade por manufaturar objetos. A *Factory* ajuda a criar o objeto apropriado a partir de um grupo relacionado de classes, permitindo que se abstraia os atributos de um objeto em subclasses específicas que os criam. Este padrão promove o desacomplamento, ao passo que elimina a necessidade de relacionar classes especificas à aplicação diretamente no código.

A Figura 17 exemplifica genericamente um *Factory*. *Product* define uma interface para o objeto a ser criado, enquanto o *ConcreteProduct* implementa esta interface. O objeto *Creator* declara o método *Factory* que retorna um objeto do tipo *Product*, ao mesmo tempo que a classe *ConcreteCreator* sobrecarrega o método *Factory* para retornar um objeto *ConcreteProduct*.

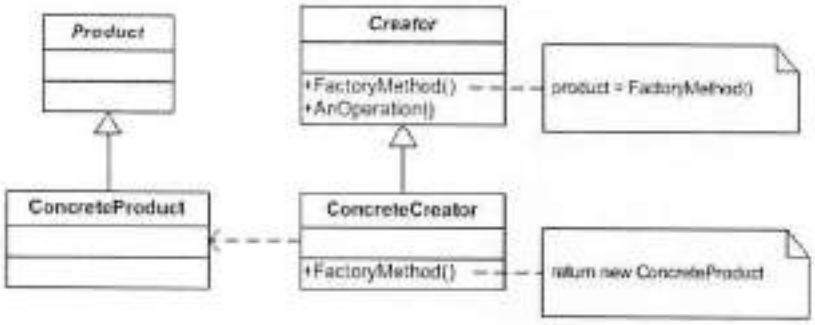


Figura 17: Diagrama de Classe genérico para um Factory[1].

4.4.4 *Mock object*

Mock Objects são entidades que mimetizam o comportamento de um objeto real. Estes são tipicamente utilizados para testar o comportamento de outro objeto real, como no caso de teste unitário definido no desenvolvimento orientado a testes.[2]

Em testes unitários, tais objetos são muito úteis quando o comportamento do objeto simulado não é determinístico, possui diversos estados possíveis ou ainda não foi desenvolvido. O objeto simulado provê a mesma interface que o objeto real, de forma que o cliente que o utiliza não percebe se este é real ou não.

O uso de desenvolvimento orientado a testes com *mock objects* permite que o desenvolvimento seja mais rápido e independente, e possa evoluir independentemente de serviços e acessos a base ainda não implementados. Quando o objeto falso é substituído pelo real, a funcionalidade fim-a-fim está, em princípio, garantida, necessitando testes de integração menos trabalhosos. Tal técnica foi largamente utilizada durante a fase de desenvolvimento do sistema SCIP.

[1] Fonte: GoF – www.dofactory.com
[2] Vide seção 4.7. Desenvolvimento Orientado a Testes.

4.4.5 *Generics*

Um método genérico é um método que é declarado com parâmetros de tipo genéricos (*typed parameters*). Classes e métodos genéricos combinam reutilização de código, eficiência e segurança de tipo de forma a permitir maior escalabilidade e concisão de implementação, que de outra forma requereria muitas linhas de código.

O parâmetro genérico é usado em diversos lugares onde se usaria um tipo concreto para definir um item ou objeto, tal como tipo de um objeto em uma lista, tipo do retorno de um método, tipo de um parâmetro, etc.

A grande vantagem fornecida por esta técnica é a definição de uma classe genérica que implementa funções genéricas, podendo esta ser estendida para uma classe específica apenas definindo-se o tipo do parâmetro. Para reutilizar o método definido, basta criar uma nova instancia com um tipo genérico diferente.

Generics são uma forma de se criar algoritmos com baixo acoplamento aos tipos que eles operam. A implementação é específica, dependendo do parâmetro passado, mas a semântica do algoritmo não se altera.

4.5 Ajax

AJAX é um acrônimo para *Asynchronous Javascript And XML*. AJAX se caracteriza pelo uso de Javascript e XML para tornar a navegabilidade em ambientes *web* mais interativa e dinâmica, realizando requisições assíncronas ao servidor (controle dos eventos de *post-back*).

Tipicamente, em uma aplicação *web* o cliente dispara solicitações ao servidor a cada ação tomada pelo usuário (clique de um botão, seleção de item numa lista ou tabela, etc.) e o servidor processa essas requisições, retornando uma nova página HTML ao cliente. Isto implica que, a cada ação do usuário, a página vista na tela do cliente (navegador) é novamente carregada, acarretando certo desconforto visual e descontinuidade na utilização da aplicação.

4.5.1 Princípios do AJAX

O AJAX tem, como um de seus princípios, a transferência de parte da lógica do aplicativo (servidor) para o navegador (cliente). Desta forma, informações sobre sessão, dados sobre operações ou até funções são implementadas diretamente no navegador.

O navegador, acessando uma aplicação utilizando AJAX, realiza chamadas assíncronas ao servidor para recuperar dados. O carregamento da página é feito uma vez, no início da sessão, após qual se realiza acesso aos dados. Como resultado, o tráfego com o servidor é mais intenso no início, e as comunicações subseqüentes com o servidor são menos freqüentes e mais eficientes.

AJAX é uma técnica baseada em *script* remoto que permite ao navegador executar requisições ao servidor sem enviar todo o conteúdo da página ao servidor. Isto permite ao cliente recuperar pequenos pacotes de dados e atualizar dinamicamente partes da página (um dado de uma tabela, uma caixa de texto, etc.). O resultado é uma melhora significativa na experiência do usuário e aplicações coerentes com os predicados da Web 2.0. (LARMAN, 2002)

A assincronia do AJAX advém do fato de que toda requisição ao servidor e o carregamento são feitos sem interferir no comportamento da página existente. As chamadas são tipicamente feitas em *javascript* utilizando o objeto *XMLHttpRequest*.

AJAX é usado pra criar aplicações *web* mais amigáveis e agradáveis ao uso. O objetivo é tornar páginas mais interativas ao trocar pequenas quantidades de dados com o servidor sem ter que recarregar a página por complete toda vez que o usuário faz uma requisição. Isto aumenta a interatividade, velocidade e usabilidade da página. É importante, no entanto, salientar que a utilização de Ajax promove um ganho considerável na complexidade da aplicação. É uma opção que deve ser avaliada e aferida como uma decisão de projeto.

4.5.2 Utilização do AJAX

AJAX é utilizado freqüentemente, dentro do escopo do sistema SCIP, em:

- Validação de formulários;
- Auto-preenchimento: sugerir palavras à medida que usuário as digita;
- Carga sob demanda: dados são carregados de acordo com necessidade do cliente
- Controles sofisticados e efeitos;
- Atualização de dados: permite atualizar dados da página sem ter que recarregá-la;
- Página *Web* como aplicação: é possível disponibilizar um aplicativo via uma única página que tem a usabilidade de um aplicativo *desktop*.

4.6 Programação por Interfaces

Programação por interface contempla a gestão cuidadosa de relações de dependência entre classes em um aplicativo. É extremamente trivial adicionar uma dependência para cumprir uma funcionalidade, mas remover uma dependência pode acarretar em trabalhosos refatoramentos (*refactoring*) ou até impossibilitar a extensibilidade do código (FOWLER, 1999).

A introdução de uma interface permite uma abstração da implementação, acarretando maior desacoplamento dos componentes do sistema (Figura 18). Ela define os métodos que um cliente precisa para que outra classe realize a implementação, podendo esta variar independentemente do cliente. Isto permite que se utilize, por exemplo, *mock objects* simples para substituir pesadas implementações de bases de dados.

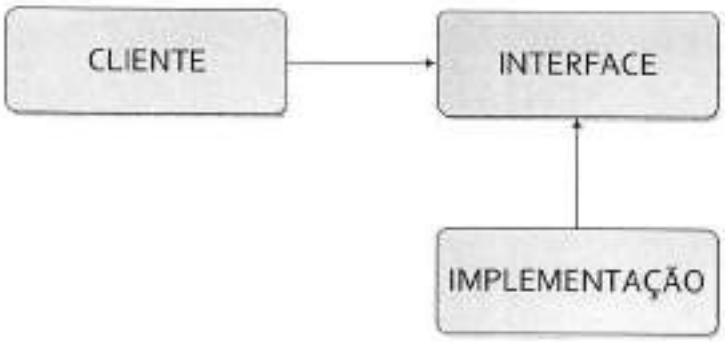


Figura 18: Interface genérica.

Esta abordagem permite maior flexibilidade ao mesmo tempo que separa o projeto da implementação, permitindo maior desacoplamento entre cliente e implementação. O uso de interfaces pode também ser substituído por classes abstratas, que em alguns casos podem fornecer ainda mais flexibilidade para evoluções futuras. Isto ocorre pois, quando se adiciona um método a uma interface, faz-se necessária a reestruturação de todos os clientes que implementam a interface, ao passo que quando se adiciona um método a uma classe abstrata, todas suas herdeiras herdam este método (Figura 19).

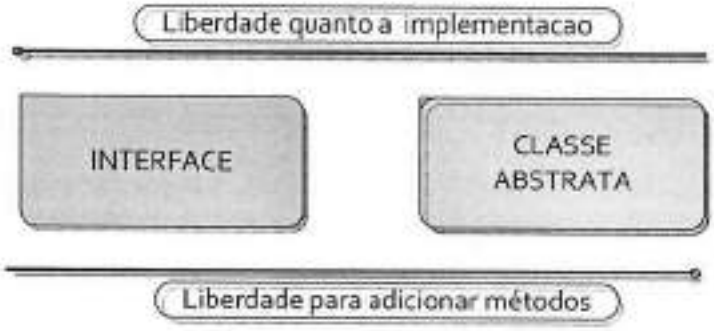


Figura 19: Ganhos de Liberdade de Interfaces e Classes abstratas.

Como conseqüência dessa inflexibilidade das interfaces, deve-se considerá-las imutáveis uma vez publicadas. Ao adicionar novos métodos a uma interface, deve-se então definir uma nova interface para não inutilizar os clientes que a implementam.
Uma interface é livre da implementação e define o vocabulário para colaboração entre objetos. Uma vez que se entenda uma interface, entende-se o sistema como um todo. O vocabulário consiste da definição dos nomes de métodos e das abstrações.

Quando um cliente se utiliza de um objeto inserido em um contexto de objetos hierárquicos, o cliente deve escrever o código orientado a interface do tipo do objeto que se encontra na parte mais abstrata da hierarquia.

Pensar em interfaces é mais importante à medida que o projeto ganha escala. Se um projeto se restringe a poucas pessoas, o refatoramento permite que todas as modificações sejam propagadas pelo código de maneira pouco custosa. Mas quando o desenvolvimento abrange equipes grandes, cada uma recebe uma responsabilidade distinta, tal como a responsabilidade sobre um componente. Cada grupo é responsável por criar sua próprias APIs e por mantê-las funcionando uma vez que tenham sido publicadas (VENNERS, 2005).

4.7 Desenvolvimento Orientado a Testes

Tipicamente, testes de código são realizados após o código ser escrito. Freqüentemente torna-se difícil realizar os testes à medida que o tempo disponível para o projeto diminui, resultando assim em testes incompletos ou parciais.

O desenvolvimento orientado a testes (TDD, do inglês *Test Driven Development*) é uma corrente de desenvolvimento de *software* que tem por objetivo gerar código de qualidade e bem testado através da definição dos testes antes da implementação do código (NILSSON, 2006).

A técnica consiste na criação de testes unitários que verificam o comportamento de uma classe ou um método, passando parâmetros, testando suas especificações e verificando o resultado obtido. Os testes unitários definem o comportamento da classe à medida que geram a necessidade de criar código para cumprir uma funcionalidade. Ela permite focar nos itens mais importantes e priorizar a codificação. Isto implica que o resultado final do *software* depende da boa definição dos testes, já que estes são a forca motriz do desenvolvimento. É uma técnica poderosa que permite maior qualidade de código garantindo cobertura das funcionalidades produzidas e maior consistência das respostas.

Testes unitários apresentam uma dificuldade em sua definição, visto que estes não devem exceder as barreiras de processo, interface e sistema. Tipicamente objetos e métodos utilizam outras classes para realizar suas tarefas; logo,ao definir um teste para um objeto específico, está de testando indiretamente todos os métodos e objetos por ele utilizados, gerando repetição e sobreposição desencessária dos testes . Uma forma de contornar isto é a utilização de *MockObjects*, que nada mais são que uma classe que emula outra classe real. Desta forma, pode-se desacoplar o desenvolvimento de parte do código de outra, por exemplo, criando um acesso falso a base de dados. Testam-se métodos que utilizam dados da base sem se preocupar à priori com a criação da base. Isto leva à criação de código mais modular, flexível e extensível, com classes menores e menos acopladas, podendo facilmente ser trocadas por novas implementações. Uma grande vantagem desta metodologia é a identificação de defeitos no estágio inicial da codificação

O TDD prova a correção do código de acordo com os casos de teste escritos. Um teste incorreto que não representa a especificação irá produzir código incorreto (semanticamente incorreto). Logo, o foco de implementação é mudado para a concepção de casos de teste, os quais mostram o enquadramento ou não do sistema dentro de sua especificação funcional. Ainda, tem-se que esta metodologia de teste não se aplica a todas as abordagens computacionais, tais como criptografia, inteligência artificial, segurança, onde testes não podem ser automatizados de forma tão simples.

4.8 Globalização e Localização

Globalização é definida como o desenvolvimento de um *software* ou aplicação de maneira que o produto seja utilizável através de culturas diversas, com padrões regionais distintos de nomenclatura, moeda, data, etc. Localização é definida como o processo de criar dados de entrada e saída em um formato específico para uma dada região ou cultura (como a representação de data no sistema americano e europeu).

Desta forma, globalização está relacionada intrinsecamente com a codificação que se faz para se desenvolver um *software*, dando suporte para mudanças que

permitam internacionalização, isto é, mudança de cultura e representações regionais. A localização tange a escolha de uma dada cultura ou região. Assim, é fácil verificar que a globalização precede a localização dentro do escopo do desenvolvimento de um *software*.

Respeitando as mais modernas tendências de desenvolvimento de *software*, o sistema SCIP suportará globalização e, para tal, alguns cuidados na especificação da interface foram tomados como a utilização de arquivos, chamados de recursos (pode ser um arquivo formatado, arquivo texto, XML, etc.), dos quais cada tela busca as informações necessárias para se adequar ao idioma e configurações regionais especificadas nas configurações da aplicação.

4.9 Agile Unified Process (AUP)

Agile Unifie Process (AUP) é uma versão simplificada do *Rational Unified Process* (RUP). A seguir apresenta-se uma breve explicação do conceito de AUP e aplicacao ao projeto.

4.9.1 Conceitos de AUP

O AUP é uma abordagem para desenvolver aplicações e sistemas usando técnicas ágeis, mas se atendo aos conceitos mais importantes do RUP (AMBLER, 2005). A Figura 20 descreve as fases e disciplinas definidas no AUP.

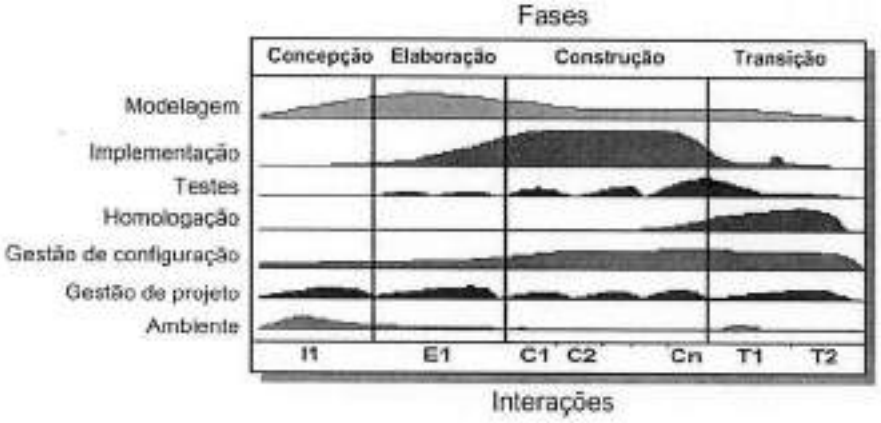


Figura 20: Ciclo de Vida do AUP.

O AUP é dividido em quatro fases:

- *Concepção*: objetiva identificar o escopo do projeto, arquitetura potencial do sistema e obter recursos e aceitação das partes envolvidas;
- *Elaboração*: objetiva validar a arquitetura do sistema;
- *Construção*: Contempla a construção do *software* num ritmo regular, incremental priorizando necessidades do projeto;
- *Transição*: corresponde a validação e homologação no ambiente de produção.

As disciplinas são abordadas de forma iterativa consistindo de:

- *Modelagem*: compreensão do modelo de negócio e dos riscos envolvidos, e identificar soluções viáveis aos problemas;
- *Implementação*: transformação de modelos em código executável e realizar testes básicos, em particular o teste unitário;
- *Teste*: avaliação objetiva para garantir qualidade. Inclui encontrar defeitos, validar funcionalidade do sistema e verificar o cumprimento de requisitos;
- *Homologação*: planejamento da homologação do sistema;
- *Gestão de configuração*: gerenciamento do acesso aos artefatos do projeto, compreendendo o rastreamento através do tempo e controle e gestão das mudanças feitas;
- *Gestão de projetos*: direcionamento das atividades do projeto, tal como gestão de risco, divisão de tarefas, e coordenação com pessoas e sistemas externos para garantir a entrega no tempo e custo previsto;
- *Ambiente*: fornecimento de suporte às outras disciplinas garantindo que o processo correto, padrões e ferramentas (HW, SW) estejam disponíveis a todos os envolvidos conforme necessidade.

O AUP permite simplicidade e agilidade no desenvolvimento, focando nos aspectos mais importantes e mantendo uma documentação concisa. A aplicação desta técnica ao longo do desenvolvimento do projeto permitiu uma modelagem mais dinâmica e iterativa do sistema.

4.9.2 Aplicação do AUP ao projeto

O trabalho de modelagem se deu de forma extremamente intensa no início do projeto, com definição dos requisitos e modelo de negócios, continuando presente ao longo do projeto inteiro, à medida que este foi sendo implementado e novas necessidades de refinamento se fizeram necessárias.

A implementação se deu de forma incremental focando nos módulos mais importantes do projeto aplicando testes unitários tal como descrito na seção 4.7. Após a integração do sistema, planejaram-se e executaram-se testes de integração para garantir o cumprimento dos requisitos, tal como descrito na seção 4.7.

A homologação do sistema não foi abordada neste projeto, visto que não está contemplada nos objetivos de um trabalho de conclusão de curso. Ademais, o acesso a sistemas corporativos em um caso real é difícil para um projeto, em princípio, de cunho acadêmico.

No início do projeto fez-se um planejamento detalhado das atividades e ponderação de tempo necessário para cada uma delas, estabelecendo-se um cronograma de projeto. À medida que o projeto evoluiu, novas atividades foram sendo definidas e re-planejadas de acordo com a necessidade. Geralmente, tal re-planejamento coincide com um entregável do projeto.

Aspectos da gestão de configuração aplicados durante o projeto são explicitados na seção 4.12.

As ferramentas necessárias ao desenvolvimento foram obtidas à medida que se mostravam necessárias para o desenvolvimento e feitas públicas a todos os envolvidos no desenvolvimento e orientadores.

4.10 Prototipagem

As técnicas e ferramentas de *software* existem para identificar requisitos ambíguos e em falta. Estes são fatores importantes no desenvolvimento de qualquer sistema de *software*. Contudo, estas questões tornam-se ainda mais complicadas com a emergente troca de requisitos de *software* ao longo do seu desenvolvimento. Projetos de larga escala estão em grande expansão nos dias de hoje e qualquer alteração no percurso da criação de um sistema pode acarretar sérios impedimentos, que levam a um aumento significativo dos custos.

A prototipagem tem vindo a ganhar uma aceitação crescente nas comunidades de Engenharia de *Software*, sendo encarada cada vez mais como um modelo credível de criação de sistemas. A prototipagem de sistemas de *software* foi usada, principalmente, para exercitar os conceitos e mostrar os requisitos do sistema. Pretende-se, assim, ajudar os clientes e os responsáveis pelo desenvolvimento do projeto a testar e a melhorar o sistema antes mesmo de este estar finalizado.

Na Figura 21 e Figura 22 são apresentados alguns dos protótipos elaborados no projeto de interface do sistema SCIP:

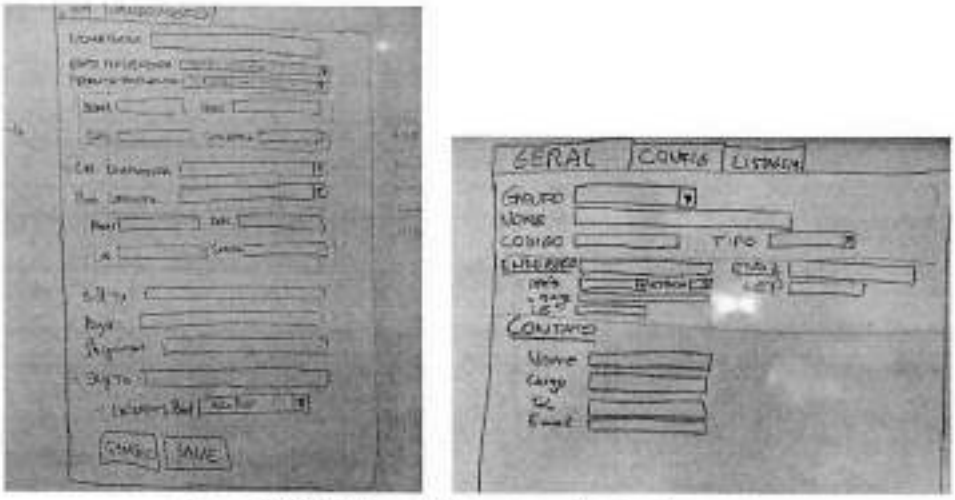


Figura 21: Protótipos de telas para cadastro de empresa.

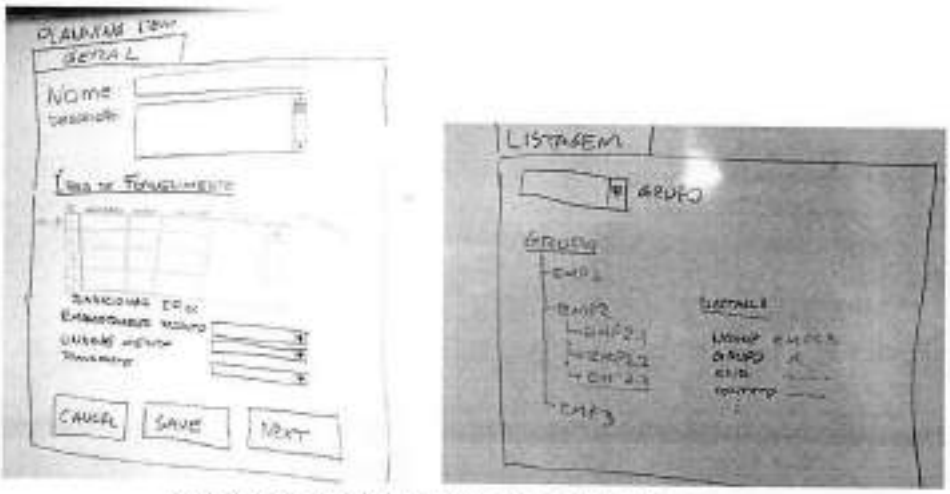


Figura 22: Protótipos de telas para cadastro de empresa.

4.11 Provas de Conceito (PoC)

Prova de Conceito (PoC, do inglês *Proof of Concept*) é o termo utilizado para denominar um modelo prático que possa provar o conceito (teórico) estabelecido por uma pesquisa, projeto ou artigo técnico.

Trata-se de um recurso muito eficaz para se testar a factibilidade de um modelo proposto, detectar suas limitações de implementação e eventuais falhas de concepção. Junto ao PoC, pode-se optar por realizar o PoT (Prova de Tecnologia, ou *Proof of Tecnology*, em Inglês).

No decorrer deste projeto, diversas provas de conceito e de tecnologia foram realizadas, as quais permitiram, a cada iteração do projeto, determinar e abrandar os riscos inerentes à implementação, sobretudo nas fases de especificação e concepção do sistema.

4.11.1 PoC de globalização e localização

Na Figura 23, é apresentada a prova de conceito realizada para testar Globalização e Localização com as tecnologias empregadas no projeto. Note-se que, ao se escolher uma nova opção de língua, a tela é recarregada com o conteúdo adaptado para a cultura específica desta língua.

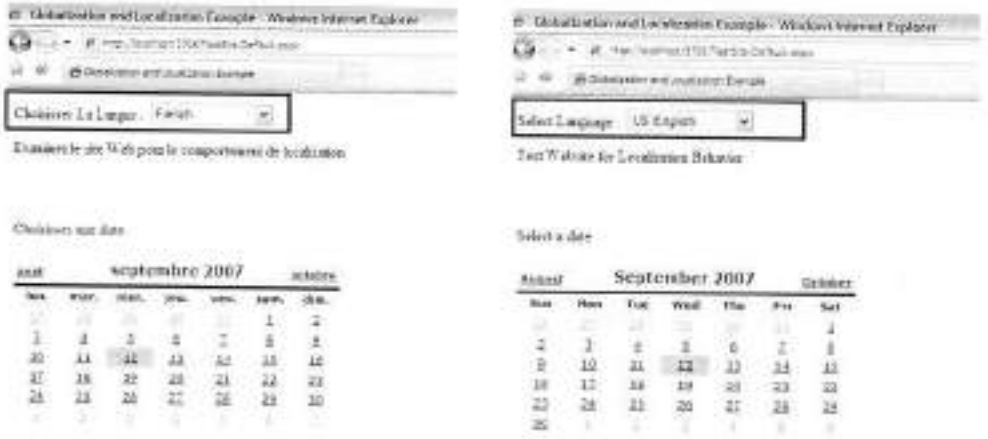


Figura 23: Prova de Conceito de Globalização e Localização para o sistema.

4.11.2 PoC de injeção de dependências

No desenvolvimento do sistema SCIP, qualidades como modularidade e baixo acoplamento entre os componentes lógicos da aplicação foram, em grande parte, favorecidos pela utilização do conceito de injeção de dependências, já explicado anteriormente neste trabalho (vide seção 4.3).

Uma questão posta durante o projeto consistiu justamente no modo de se aplicar tal conceito, visto que muitas eram as alternativas possíveis. Deste modo, partindo-se de provas de conceito realizadas com tecnologias existentes, optou-se por empregar um *container* público de inversão de controle. O resultado foi: diversas dependências entre componentes quebradas e código mais limpo e enxuto.

4.11.3 PoC de interatividade via Ajax

A interatividade e experiência com usuário são aspectos considerados de primordial importância no escopo do projeto do sistema SCIP. Desta forma, fez-se necessário utilizar Ajax (vide seção 4.5) na codificação do aplicativo, baseado em tecnologia *Web*.

Dentre as diversas alternativas existentes, desde a escrita de *javascript* puro (*raw javascript*), passando pela utilização de bibliotecas prontas (como *scriptaculous*[1]) até controles baseados em *script* servidor (Ajax .Net), optou-se pela última. Esta escolha foi motivada, após a execução de algumas provas de conceito, pela agilidade e praticidade no desenvolvimento proporcionadas pelo encapsulamento da tecnologia em controles *server-side* (prontamente integráveis ao ambiente de desenvolvimento escolhido), em detrimento da maior flexibilidade proporcionada pela escrita livre de *javascript* (consideravelmente mais complexo).

4.12 Gestão de Mudanças e Controle de Versões

O desenvolvimento de um *software* requer a interação de diversas pessoas trabalhando em paralelo ou concomitantemente para atingir o seu objetivo. Daí existe a problemática quanto ao controle das mudanças no projeto e controle de versões.

Neste projeto tomou-se a abordagem de manter um repositório de informação *on-line* (servidor SVN) contendo toda a documentação de projeto, assim como o código desenvolvido. O repositório pode ser acessado através de um cliente instalado na máquina que acessa os arquivos *on-line* guardando uma cópia de trabalho local na máquina cliente. Uma vez que um desses arquivos tenha sido modificado, o autor sincroniza as modificações com o repositório e um aviso é enviado aos outros participantes, informando que há uma nova versão disponível.

Desta forma foi possível manter uma documentação acessível e centralizada, ao mesmo tempo em que diversas pessoas puderam modificar o mesmo documento e realizar *merge* após as modificações.
Além de servir como cópia de *back-up*, o repositório permite realizar *roll-backs* sobre informações modificadas em versões anteriores e manter um histórico das mudanças efetuadas.

[1] Biblioteca de código aberto para se produzir conteúdo web com Ajax.

5 Projeto do Sistema SCIP

Este capítulo é relativo às fases de especificação e desenvolvimento do projeto, contemplando os tópicos mais relevantes do desenvolvimento do sistema SCIP. A teoria exposta no capítulo 4, somada aos processos de negócio apresentados no capítulo 3, serviu de base para a construção dos artefatos que orientaram a implementação do sistema SCIP.

5.1 Especificação de Requisitos do Sistema

Esta seção apresenta os requisitos funcionais e não-funcionais do sistema, em vista da perspectiva traçada nos capítulos anteriores deste trabalho.

5.1.1 Interfaces do Sistema

O produto deve permitir diferentes modos de interface com os sistemas dos fornecedores e clientes, através da internet, ou rede EDI, para envio de dados de demandas de entrega. Alternativamente, o produto pode estar diretamente conectado ao ERP do fornecedor. Tanto o cliente quanto o fornecedor devem estar conectados - através de conexão direta, redes EDI ou Internet - para se comunicarem com o sistema desenvolvido.

5.1.1.1 *Interfaces com Usuário*

A interface com o usuário é realizada via *web*, através de páginas de navegação dinâmicas, devendo ser intuitiva e convivial, de modo que o usuário não precise de treinamento extenso para operá-la. Sendo assim, o sistema é projetado para navegação *web*, permitindo acesso distribuído às funcionalidades do sistema.

5.1.1.2 *Interfaces de Hardware*

O sistema não faz interface com sistemas de hardware especiais, necessitando apenas de um servidor de aplicações, a partir do qual os usuários possam acessá-lo através de rede local ou Internet.

5.1.1.3 *Interfaces com Software*

Outros produtos de *software* tangem o sistema, tais como sistema gerenciador de banco de dados e sistema operacional. O sistema armazena dados da aplicação no banco de dados e utiliza os mesmos para realizar suas funções. A interface entre ambos dar-se-á por solicitações do sistema ao SGBD (sistema gerenciador de banco de dados) para armazenar, recuperar e realizar operações sobre dados.

Ainda, para a construção de gráficos, execução do processo de Inversão de Controle e log do sistema, foram utilizadas bibliotecas padrão de código aberto, disponíveis para a comunidade de desenvolvimento .Net.

5.1.1.4 *Interfaces de Comunicação*

O sistema se comunica com elementos externos através da Internet, utilizando o protocolo TCP/IP e redes EDI, que possuem um protocolo de mensagens padronizado.

Como cada rede EDI possui seu próprio padrão de comunicação e padrão de mensagens, o sistema deve ser capaz de interpretar as mensagens em cada rede e gerar mensagens no padrão adequado. Da mesma forma, sistemas ERP possuem padrões de mensagens diferentes; logo o sistema deve ser capaz de se comunicar com diversos padrões de mensagem ERP previamente configurados.

Assim, deve existir uma camada de tradução de dados que formate os dados internos do sistema adequadamente de acordo com o sistema ERP e rede EDI utilizados para a comunicação.

5.1.2 Operação

O sistema possui três modos de operação:

Modo 1. O primeiro modo, não privilegiado, provê acesso às operações necessárias do cliente para registro de produto em demanda, mudança de dados cadastrais, acesso a dados históricos de entregas, consultas e geração de relatórios do cliente.

Modo 2. O segundo modo, não privilegiado, provê acesso às operações necessárias ao fornecedor para cadastro de produtos disponíveis, mudança de dados cadastrais, acesso a dados históricos, consultas e geração de relatórios do fornecedor.

Modo 3. O modo privilegiado permite a configuração do sistema, restrição de acessos e operações de *backup* e *restore*. O *backup* deve ser realizado com auxílio de outro computador ou mídia onde os dados serão salvaguardados regularmente.

As funções de agendamento de entregas e verificação de dados do cliente devem ser realizadas periodicamente e automaticamente, sem a necessidade de intervenção de usuário, ou ser disparados por uma alteração nos dados, tal como o recebimento de novos níveis de estoque ou *forecasts*.

5.1.3 Requisitos funcionais de *Software*

Cadastros

- Cadastro de empresas, fornecedoras e clientes;
- Cadastro de produtos;
- Cadastro de grupos de empresas;
- Cadastro das regras de negócio do produto/cliente/fornecedor;
- Cadastro de usuários e perfis de acesso.

Criação de Relatórios

- Criação de relatório de entregas;
- Criação de relatórios de estatística de estoque/consumo.

Integração EDI/ERP

- Registro de níveis de estoque;
- Registro de previsões de consumo;
- Envio manual de níveis de estoque e *forecast*;

Supervisão da programação de entregas

- Visualização do calendário de entregas;
- Envio da Programação de Entregas;
- Registro de Confirmação de Programação de entrega;
- Registro de aviso de embarque;
- Confirmação de ordens de entrega;
- *Track&Trace* dos produtos enviados.

Programação de Entregas

- Estabelecimento de quantidades e datas de entrega;
- Cálculo de previsões de estoque;
- Criação de previsões de consumo.

Situações excepcionais

- Reagendamento de entregas;
- Modificação do calendário de entregas;
- Identificação de falhas no envio/recebimento de mensagens da rede EDI/Internet;
- Identificação de falhas na comunicação com ERP.

5.1.4 Recursos necessários

Paralelamente à análise e especificação de requisitos, foram identificados os principais recursos necessários ao projeto, os quais são listados a seguir.

5.1.4.1 *Desenvolvimento*

Como ambiente de desenvolvimento, optou-se pela escolha da plataforma .Net, utilizando-se C# 2.0 como linguagem de programação. Os requisitos desse ambiente são:

Sistema Operacional – Microsoft Windows XP
Versão – Edição Profissional (licença gratuita para pesquisa)
Fonte – Microsoft

Servidor Web – *Internet Information Services* (IIS)
.Net – framework 2.0.50727
Plataforma de desenvolvimento – Microsoft Visual Studio 2005
Fonte – Microsoft

SGDB – Microsoft SQL Server 2005
Versão – *Express Edition* (licença gratuita)
Fonte – Microsoft

O desenvolvimento requer que o servidor, contendo o SGBD e as máquinas de desenvolvimento estejam ligadas à internet e que os aplicativos envolvidos possuam o devido acesso à rede (portas de entrada e saída abertas).

A equipe de desenvolvimento teve contato com exemplos reais e layouts de arquivos de transmissão de dados EDI. A possessão de níveis de estoque e de previsões de consumo reais era recomendada, porém não obrigatória para a simulação do sistema ao longo do desenvolvimento.

5.1.4.2 *Utilização*

Na fase operacional, será necessário o uso de um servidor dedicado para a instalação e execução do SGDB e para armazenamento do banco de dados e um segundo servidor dedicado para a execução da aplicação. Ambos os servidores devem possuir:

- Ambiente Windows;
- Capacidade de processamento necessária para execução do aplicativo;
- Conexão à Internet confiável, segura e de alta velocidade;
- Conexão confiável e segura a redes EDI;
- Os aplicativos envolvidos devem possuir o devido acesso a redes instaladas nos servidores.

Ademais, os sistemas ERP do cliente e fornecedor devem possuir acesso seguro e confiável a redes EDI ou Internet e a interface de utilização para usuários requer conexão à internet confiável e rápida.

5.1.5 Restrições e requisitos não funcionais

5.1.5.1 *Característica dos Usuários*

O usuário que operará o sistema será funcionário da empresa do cliente ou do fornecedor, não necessariamente com conhecimentos profundos de computação. O usuário tem no mínimo formação média, treinamento básico em sistemas ERP e conhecimentos básicos de logística (demanda, estoque, *forecast*, etc.). Espera-se que o usuário esteja a par dos processos de controle, entrega e recebimento de produtos de sua empresa.

5.1.5.2 *Aspectos Legais*

Os dados de consumo, entregas e relatórios devem ser mantidos confidenciais. A distribuição desses dados para terceiros ou externos depende do contrato realizado entre cliente e fornecedor. O cliente e fornecedor devem assinar contrato formal com o provedor de serviços VMI, onde o fornecedor assume a responsabilidade de manter o estoque do cliente. O sistema SCIP, porém, assume a responsabilidade no caso de falhas que gerem prejuízo para uma das partes envolvidas. O sistema SCIP não assumirá responsabilidade sobre falhas decorrentes de erros em previsões de consumo, geradas pelo cliente ou pelo próprio sistema.

5.1.5.3 *Disponibilidade*

O sistema lida com informações essenciais sobre níveis de estoque e a falha na provisão deste implica em prejuízos de produção e impactos financeiros consideráveis. Sendo assim, o sistema deve estar disponível 24 horas por dia, sete dias por semana, salvo em períodos de parada programada e previamente notificada.

5.1.5.4 *Funções de auditoria*

O produto deve permitir a verificação da acurácea das sugestões de entregas e cálculo de previsões de entregas e consumo a partir de uma comparação com as entregas realizadas de fato e o comportamento real do estoque do cliente.

5.1.5.5 *Requisitos de linguagens de programação e ambientes de desenvolvimento*

O sistema deve ser projetado para ambiente web, onde se utilizará uma linguagem compatível com navegadores comercialmente disponíveis e que permita a geração dinâmica de dados.

5.1.5.6 *Protocolos de Comunicação*

O sistema deve ser capaz de enviar e receber mensagens em padrão próprio, em padrão proprietário ou padrão aberto de redes EDI, assim como comunicação via TCP/IP. Além disso, deve ser capaz de interpretar padrões de mensagens de diferentes sistemas ERP.

5.1.5.7 *Operações Paralelas*

O sistema deve ser capaz de receber e processar múltiplas requisições de usuários que o acessam via web, assim como processar os vários dados enviados e recebidos pelos ERPs dos clientes e fornecedores.

5.1.5.8 *Requisitos para adaptação do local*

O sistema deve incorporar todas as funções necessárias para configurar os dados de clientes, fornecedores, produtos, e regras de negócio de cada trio cliente – fornecedor – produto. As configurações para comunicação com diferentes redes EDI e sistemas ERP devem ser feitas manualmente no sistema, desenvolvido de forma modular.

5.1.5.9 *Considerações sobre a segurança*

O sistema não deve permitir acesso ou alteração indevida dos dados por pessoas não autorizadas. Os dados enviados e recebidos devem ser criptografados.

5.1.5.10 *Requisitos de confiabilidade*

O sistema deve ser capaz de enviar alertas à equipe técnica responsável pela manutenção, caso ocorra uma falha no envio/recebimento de mensagens EDI/ERP.

5.2 Modelo Estático

O modelo estático compreende a modelagem das funcionalidades do sistema e sua estrutura de dados, sendo composto pelos casos de uso, diagrama de classes e diagrama de entidade-relacionamento.

5.2.1 Casos de Uso

Os casos de uso foram modelados de acordo com a teoria de linha de produto de *software*, de acordo com Gomaa (2004), visando a reusabilidade como ferramenta para se obter modularidade e permitir a composição de diversos produtos finais diferentes. Isto se reflete basicamente na especificação da categoria de reuso, assim dividida:

- *Kernel*: especifica um caso de uso central, necessário para garantir o funcionamento essencial do sistema, sendo assim comum a todos os produtos da linha;
- *Optional*: Caso de uso opcional à configuração de um produto, permitindo configuração de produtos diferentes;
- *Alternative*: um caso de uso alternativo é utilizado para um conjunto de casos de uso que podem ser utilizados como alternativas para a configuração de um sistema.

Ademais, os casos de uso seguem a definição clássica de engenharia de *software* de acordo com UML 2.0 contendo sua descrição, evento iniciador, atores envolvidos, categoria de reuso, pré-condição, pós- condição, seqüência de eventos, extensões e inclusões.

Para os casos de uso, definiram-se quatro tipos de atores que interagem com o sistema:

- Sistema ERP do cliente;
- Sistema ERP do fornecedor;
- Usuário administrador;
- Usuário comum.

Os sistemas ERP externos são vistos aqui como atores, porque interagem com o sistema na forma de envio e recebimento de mensagens, gerando mudanças de estado e disparando processos no sistema SCIP. Definem-se também dois atores, do tipo usuário, para representar níveis diferentes de permissão de edição de informação contida no sistema através da interface gráfica *web*.

A Tabela 5 contempla todos os casos de uso descritos para o sistema SCIP, cuja especificação encontra-se disponível no APÊNDICE A.

Tabela 5: Relação dos Casos de Uso Modelados.

Casos de Uso
Caso de uso 1: Cadastro de empresas (fornecedora ou cliente)
Caso de uso 2: Cadastro de itens (produtos)
Caso de uso 3: Criação de conta de usuário
Caso de uso 4: Cadastro/Configuração de perfis de acesso para o sistema.
Caso de uso 5: Autenticação de usuário no sistema (Login)
Caso de uso 6: Executa logout de usuário
Caso de uso 7: Cadastro de um item de planejamento
Caso de uso 8: Cadastro de um item de fornecimento
Caso de uso 9: Configurar regras de negócio de um item de planejamento
Caso de uso 10: Confirmar proposições de ordem
Caso de uso 11: Criação de proposição de ordem
Caso de uso 12: Modificação de ordem (proposta ou já confirmada)
Caso de uso 13: Envio de confirmação de recebimento de mercadoria
Caso de uso 14: Recebimento confirmação de ordem proposta Order Response
Caso de uso 15: Envio de aviso de expedição Ship Notice
Caso de uso 16: Solicitação de envio de Digest
Caso de uso 17: Solicitação realização de diagnóstico
Caso de uso 18: Visualização de relatórios
Caso de uso 19: Inserir forecast
Caso de uso 20: Inserir forecast manualmente
Caso de uso 21: Inserir nível de estoque
Caso de uso 22: Inserir nível de estoque manualmente
Caso de uso 23: Envio de dados de tracking
Caso de uso 24: Consultar tracking de produtos enviados

5.2.2 Modelo de Classes

O diagrama de classes do sistema deve refletir, da melhor maneira possível, as necessidades do modelo e funcionalidades do negócio. Mantendo um enfoque forte nestes requisitos, desenvolveu-se um modelo de classes que reflete as entidades do domínio da aplicação e suas características. Especial atenção foi devotada à modularidade, para permitir maior flexibilidade quanto a modificações, extensões e variantes de configuração cliente-fornecedor possíveis.

Pelo fato de se tratar de um intermediador entre sistemas diferentes (os quais possuem parâmetros e configurações distintos), o modelo de classes – ou domínio – da plataforma em desenvolvimento deve ser suficientemente flexível para se adaptar a diferentes contextos, conforme a demanda. Isto leva à necessidade de considerar

parâmetros adicionais que podem não ser necessários para todas as configurações existentes.

O modelo de classes desenvolvido e a descrição detalhada de suas principais classes, compreendendo o pacote *SCIP.Model* previsto na componentização do sistema, pode ser visualizado no APÊNDICE C.

5.2.3 Diagrama de Entidade Relacionamento

O diagrama de entidade-relacionamento reflete a estrutura dos dados para armazenamento em um banco de dados, no caso, relacional. O diagrama reflete as entidades contidas no modelo de classes explicitando as relações de dependência. A estrutura foi modelada de acordo com as regras de normalização para eliminar redundâncias e anomalias de modificações da base. Aplicou-se normalização até a terceira forma normal, mantendo independência entre os atributos das tabelas e dependência apenas da chave primária.

O diagrama das tabelas e relações resultante pode ser consultado no APÊNDICE D deste trabalho.

5.3 Modelo Dinâmico

O modelo dinâmico provê uma visão do sistema em que controle e seqüência das ações são representados, tanto no que toca ao objeto particularmente dito (por meio de máquinas de estado finito) ou pela interação entre objetos (por meio de análise das interações entre os mesmos).

5.3.1 Processamento de Ordens de Produção

A partir do momento em que uma nova ordem é proposta pelo sistema, ela é submetida a uma série de operações envolvendo transições de estados e processamentos relativos a cada estado. A definição de cada estado pode ser consultada na seção 3.6.3 (Algoritmo de gestão de ordens).

O diagrama de transição de estados mostrado na seção 3.6.3 (Algoritmo de gestão de ordens) explicita a dinâmica de estado das ordens criadas de acordo com ações dos usuários do sistema ou criação e recebimento de mensagens dos sistemas ERP externos.

5.3.2 Diagramas de Seqüência

Os diagramas de seqüência foram elaborados de acordo com notação UML 2.0 para refletir a dinâmica das funcionalidades do sistema e das classes definidas no modelo estático.

A utilização de diagramas de seqüência permite maior clareza na definição das funções do sistema, incluindo aspectos da arquitetura, fornecendo um guia para a compreensão da interação das classes modeladas. Um bom diagrama de seqüência guia a codificação do *software*, garantindo que o resultado final esteja de acordo com as especificações do sistema. Pode-se ter a relação completa dos diagramas de seqüência no APÊNDICE E deste trabalho.

5.4 Arquitetura do Sistema

A especificação da arquitetura de *software* do sistema é orientada pela teoria exposta na seção 4.1. A partir das premissas nela levantadas, desenvolveu-se a arquitetura componentizada ilustrada pela Figura 24, descrita nas próximas seções.

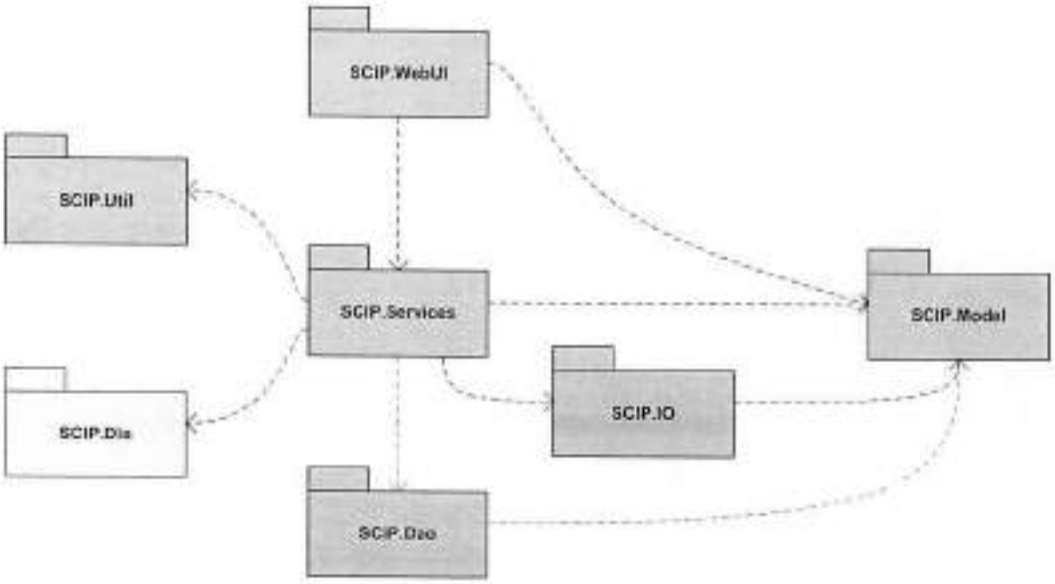


Figura 24: Arquitetura do Sistema.

5.4.1 SCIP.Model

O componente SCIP.Model contém as classes necessárias à representação das entidades do modelo de negócio envolvido no projeto – também denominadas de entidades de domínio da aplicação. Para maiores detalhes, vide seção 5.2.2.

5.4.2 SCIP.Dao

O componente SCIP.Dao implementa a camada de acesso a dados do sistema, fornecendo componentes de conexão e acesso a diversos sistemas gerenciadores de base de dados (SGBD). Fornece também as classes responsáveis pela conversão dos resultados de consultas à base de dados em objetos do domínio da aplicação – os objetos contidos na camada SCIP.Model. Toda a lógica de persistência e consulta a dados é empacotada neste componente.

5.4.3 SCIP.IO

A camada SCIP.IO define os componentes necessários para a leitura e escrita de arquivos XML. A camada Services define uma interface para a camada IO

especificando os métodos que necessita para acessar os dados contidos em arquivos XML. Desta forma, a camada IO implementa os métodos necessários, podendo facilmente ser substituída por outra implementação, cumprindo os requisitos de baixo acoplamento e modularidade.

A classe FlatFileIO define os métodos genéricos para tratar arquivos XML, tal como abertura, leitura e serialização/desserialização). Esta classe é estendida para criar classes concretas que realizam o tratamento de cada tipo de mensagem XML especificada.

5.4.4 SCIP.Services

A camada SCIP.Services contém as classes de serviço do sistema, as quais espelham os processos de negócio definidos no capítulo 3 (Processos de Negócio). A função desta camada, além de fornecer os serviços de reaprovisionamento e de processamento das mensagens de entrada e saída do sistema, é de prestar serviços à camada de interface com usuário, no que toca à lógica de validação, coleta e persistência de dados.

5.4.5 SCIP.WebUI

A camada SCIP.WebUI implementa os componentes da interface com o usuário, sendo responsável pela lógica de apresentação da aplicação. Permite a visualização dos dados pelo usuário e a interação com o mesmo.[1]

5.4.6 SCIP.Util

O componente útil engloba classes e métodos genéricos que implementam funcionalidades comuns necessárias aos outros componentes. Ultimamente, este componente funciona como uma biblioteca de classes de utilidades comuns da aplicação.

---

[1] Vide seção 4.1.3.

5.4.7 SCIP.Dis

O componente SCIP.Dis (acrônimo para *Data Integration Services*) é responsável por realizar a integração entre o sistema SCIP e sistemas externos. O sistema SCIP gera mensagens utilizando um padrão interno desenvolvido pela equipe. Pelo fato de no mercado existirem diversos padrões de mensagens EDI, é necessário um serviço de tradução para viabilizar qualquer tipo de comunicação. O componente Dis acumula a responsabilidade de traduzir as mensagens enviadas para os padrões necessários e converter as mensagens recebidas para o padrão interno. No mercado existem diversos pacotes de *software* que realizam este tipo de serviço. Desta forma, o desenvolvimento deste componente está fora do escopo deste projeto.

5.5 Algoritmo de aprovisionamento

Os diagramas que documentam o algoritmo de aprovisionamento implementado podem ser vistos no APÊNDICE G. O algoritmo foi implementado de acordo com a teoria apresentada na seção 3.6.

5.6 Processamento de Mensagens

A comunicação do sistema SCIP com sistemas e aplicações externas é realizada através do componente SCIP.DIS. O padrão das mensagens foi definido pela equipe com o intuito de servir como base para comunicação entre o componente de entrada e saída (SCIP.IO) e o componente SCIP.DIS. O padrão desenvolvido utiliza mensagens XML baseadas na norma *Chem eStandards* da CIDX.

A CIDX é uma organização sem fins lucrativos dedicada a criar padrões *eBusiness* para o setor industrial químico. A norma *Chem eStandards* especifica diversos tipos de mensagens para a troca de dados entre empresas, com o objetivo de se realizar comércio eletrônico. Dentre as diversas mensagens descritas pelo padrão, foram escolhidas as seguintes, para serem integradas ao sistema SCIP:

- *Order Create;*

- *Order Response;*
- *Ship Notice;*
- *Supply Plan.*

O padrão utilizado pelo sistema SCIP foi desenvolvido pela equipe, a partir de simplificações das mensagens originais do padrão *Chem eStandards*, mantendo apenas os dados necessários para o funcionamento do sistema. Para maiores detalhes sobre a especificação das mensagens, vide APÊNDICE H.

5.7 Projeto de Interface com Usuário

A interface com o usuário foi desenvolvida em tecnologia Asp.Net com recurso a Ajax. Seguindo as recomendações da W3C (*World Wide Web Consortion*) com respeito ao desenvolvimento de interface Web, optou-se por editar o conteúdo das interfaces (texto propriamente dito) através de folhas de estilo – CSS, do Inglês *Cascading Style Sheet* – separadas do *script* de cada tela. Os benefícios desta separação são evidentes, do ponto de vista que ela permite centralizar o padrão de estilo de toda a aplicação em alguns arquivos CSS.

No tocante à tecnologia utilizada (Asp.Net), faz-se necessário mencionar que os recursos providos pela mesma tiveram grande impacto sobre o projeto. Muitos dos conceitos salientados no Capítulo 4 (Conceitos e Ferramentas de Engenharia de *Software*) só foram passíveis de implementação, dentro do prazo determinado, devido à existência – dentro da plataforma .Net – de um ambiente que abrange muitas das melhores práticas existentes em engenharia de *software*. Podem-se aqui citar os arquivos de *resources* (permitem internacionalização de uma maneira simples e eficiente), os *skins* e *templates* (permitem aplicar um padrão determinado para toda aplicação) e, por fim, os *master pages*, recurso poderoso que permite herança de formulários e páginas *web*.

Nas seções que seguem, são reforçados outros pontos que guiaram o projeto da interface do sistema.

5.7.1 Acessibilidade

Seguindo as orientações do WAI (*Web Accessability Iniciative*), esforços[1] foram realizados no sentido de se desenvolver interfaces com usuários que respeitem alguns requisitos de acessibilidade, como menu de navegação acessível, aplicação de descritivos às imagens (*script* que descreve a imagem e que permite interpretação aos leitores de *browser* comerciais), facilidade de acesso ao conteúdo e navegação por formulários utilizando teclado.

Ainda assim, importa ressaltar que não se enquadra no escopo deste projeto (pelo menos em sua primeira versão) a criação de interfaces acessíveis no sentido amplo da palavra, dada a extensão do assunto e a janela de tempo que não permitiu o aprofundamento no mesmo.

5.7.2 Desenvolvimento Interativo

O desenvolvimento das interfaces passou por diversas etapas até atingir seu estado de maturidade. Iniciou-se pela técnica de prototipacão rápida (utilizando quadro branco, papel, etc.) para se especificar o layout e conteúdo geral das telas.

Feito isto, buscou-se auxílio de pessoas externas ao projeto (não diretamente envolvidas) para se captar opiniões, capturar os reflexos de utilização e impressões do usuário (por exemplo, pediu-se para que o usuário da interface de papel "pensasse em voz alta"). Através da observação do comportamento e reações dos usuários, a cada iteração, realizaram-se melhorias graduais no protótipo das interfaces.

Após esta etapa, passou-se à implementação das mesmas e, novamente, iniciou-se um ciclo de estudo do comportamento do usuário em vista da interface – desta vez em função do modo de apresentação do conteúdo e não do conteúdo propriamente dito. Como resultado, viabilizou-se a escolha dos melhores controles a serem

[1] Note o uso da palavra *esforços*, no sentido que alguns aspectos de acessibilidade foram estudados e aplicados, não sendo, porém, suficientes para caracterizar as interfaces desenvolvidas como acessíveis.

empregados para cada tipo de representação requerida (leia-se por controles caixa de texto, lista, caixa de checagem, lista expansível, ou seja, controles *web*).

5.7.3 Interfaces

Conforme mencionado no Capítulo 4, buscou-se inspirar as interfaces no conceito de simplicidade e usabilidade prescritos pela mais moderna tendência de desenvolvimento *web* (*Web* 2.0). Para isso, *design* de páginas *web* teve que ser pesquisado e Ajax, para se melhorar a interatividade com o usuário, foi largamente empregado. Vide o APÊNDICE I APÊNDICE Gpara se ter uma visão geral das interfaces desenvolvidas.

5.8 Plano de Testes e Validação

Como descrito na seção 4.7 (Desenvolvimento Orientado a Testes), aplicou-se, no desenvolvimento deste projeto, a metodologia de desenvolvimento orientado a testes. Desta forma, garantiu-se o funcionamento e teste dos componentes de *software* projetados para o sistema ao longo do período de implementação. Para tal, foi empregada a ferramenta *TestDriven.Net*, que possibilita a especificação de testes e a verificação da cobertura dos mesmos.

A utilização de *MockObjects* para substituir componentes reais permitiu realizar testes unitários ao longo do desenvolvimento sem que todas as camadas estivessem presentes. Ao longo do desenvolvimento, realizaram-se pequenos testes unitários para provar funcionalidade de acordo com a especificação e, à medida que se aproximava a última fase do projeto, criou-se os testes de camadas, isolando-as de suas interfaces.

Por último, realizaram-se os testes integrados, que testam o comportamento do sistema integrado e o cumprimento dos requisitos e principais funcionalidades. A cobertura dos testes unitários e o plano de testes integrados, contendo a descrição dos testes e validação, encontram-se disponíveis no APÊNDICE J.

6 Considerações Finais

6.1 Aplicações

O desenvolvimento do sistema SCIP foi motivado pela necessidade da construção de uma plataforma de serviços extremamente flexível, atuante como integradora dentro de cadeias logísticas. Deste modo, o SCIP se comunica com qualquer sistema ERP existente no mercado através de um módulo tradutor adequado, o componente DIS (*data integration service*), permitindo que fornecedores possam prover serviços VMI automatizados a seus clientes sem recorrer a plataformas de serviço externas.

O sistema SCIP apresenta-se independente de implementações de redes EDI (distintas e proprietárias), provendo serviços de integração para diversos parceiros. Como alternativa, o SCIP também pode ser utilizado para realizar a troca de informações via Internet, possibilitando a integração de empresas que não possuam acesso a redes EDI.

A adaptabilidade do sistema SCIP permite que seja instalado e homologado de forma rápida e eficiente. A configuração de um item de planejamento possui poucas etapas e pode ser realizada por completo em menos de uma hora, tendo-se em mãos todos os dados necessários para iniciar um contrato VMI.

A robustez de seu modelo de negócio, caracterizado pela ampla abrangência funcional, permite que o sistema SCIP se adeque às necessidades e processos de fornecedores e clientes dos mais variados ramos – indústria, varejo, setor agropecuário, etc.

6.2 Conclusões Finais

O objetivo deste trabalho consistiu em conceber e desenvolver um sistema *on-line*, capaz de integrar a cadeia de suprimentos de clientes e fornecedores através da monitoração e controle remoto dos estoques de produtos, aplicando o conceito de reposição continua de estoque contido no VMI. Trata-se de uma forma de reduzir custos de transação e melhorar o desempenho da cadeia produtiva, na qual o fornecedor é responsável pelo gerenciamento dos níveis de estoque de seus clientes.

A execução deste projeto contemplou a pesquisa e levantamento de requisitos e restrições para conceber um sistema que aplica o estado da arte em metodologia e técnicas de modelagem de *software*, vindo ao encontro de modernos conceitos e tecnologias de integração B2B. Igualmente, o trabalho de pesquisa realizado no domínio de logística de negócios e automação industrial foi essencial para a especificação dos processos inscritos no contexto do sistema e para sua abrangência e flexibilidade do ponto de vista de negócios.

Atenção especial foi dedicada à modelagem e a aplicação de metodologias e técnicas de engenharia de *software* que permitiram a concepção de elementos de *software* com baixo nível de acoplamento ente si, a construção de um sistema escalável, adaptável, de fácil operação e manutenção. A codificação, por sua vez, permitiu à equipe desenvolvedora deparar-se com novas tecnologias e compreender de forma holística o impacto de uma modelagem completa e eficaz sobre a modularidade e estensibilidade de um *software*. Por fim, pode-se enunciar a atenção especial que foi dedicada ao projeto da arquitetura do sistema.

O desenvolvimento apresentou-se como uma tarefa desafiadora, implicando em dificuldades maiores que as inicialmente esperadas. Entretanto, estas foram transpostas graças ao planejamento eficiente, dedicação da equipe e ótima orientação provida pelos professores orientadores.

6.3 Trabalhos Futuros

O projeto desenvolvido prevê diversos pontos de continuidade de pesquisa e desenvolvimento, os quais não foram abordados neste trabalho por restrições de escopo e tempo. Entre os pontos de continuidade identificados, pode-se citar:

- *Criação de um módulo gerador de previsões de consumo*: o sistema atual depende do cliente para envio de previsões de consumo para seu funcionamento adequado. Através de técnicas de análise estatística e inteligência artificial pode-se criar um módulo gerador de previsões de consumo que se utilize de dados de consumo histórico de uma empresa para prever o consumo futuro;
- *Adição de outros algoritmos de aprovisionamento*: o sistema atual possui um algoritmo de aprovisionamento que contém três estilos de aprovisionamento (vide seção 3.6.2). Podem-se adicionar algoritmos de aprovisionamento baseados em Pontos de Pedido (vide seção 2.5.3.1) que não dependam de previsões de consumo de enviadas pelo cliente, como alternativa ao algoritmo já desenvolvido.
- *Adição de mensagens*: Inicialmente foi prevista a possibilidade de envio da mensagem *Supply Plan* pelo sistema SCIP. O *Supply Plan* é uma mensagem baseada no padrão CIDX (vide seção 3.2) que contém um resumo de todas as ordens de venda confirmadas pelo fornecedor mas ainda não entregues ao cliente. Esta atua como uma mensagem de resumo informando ao cliente todas as entregas por vir.
- *Possibilidade de roteamento*: Adicionar ao sistema SCIP a possibilidade de rotear mensagens EDI entre diferentes redes.
- *Adição de níveis de usuários*: Inicialmente foi prevista a criação diferentes níveis de acesso de usuário. Contudo, devido ao tempo disponível não foi possível realizar esta implementação. O sistema desenvolvido contempla apenas um tipo de usuário administrador, capaz de modificar qualquer configuração existente no SCIP.
- *Melhorias de interface*: Adicionar à interface existente a possibilidade de se visualizar diversos Itens de Planejamento simultaneamente.

- *Inclusão de uma ferramenta DIS*: o componente que realiza a tradução das mensagens XML pode dar origem a um projeto inteiramente novo, que realiza a integração SOA de sistemas diferentes utilizando padrões de mensagens XML diferentes;
- *Testes com ERPs reais*: integração do sistema desenvolvido com sistemas ERP reais para testar funcionalidade em ambiente homologado.
- *Modelo de negócio orientado ao cliente*: adaptação do modelo de negócio orientado para o cliente em vez do fornecedor, funcionando como ferramenta de gestão de fornecedores.
- *Adição de criptografia das mensagens*: desenvolver mecanismo de segurança e criptografia para um sistema que recebe e envia dados de negócio críticos e sigilosos;

7 Referências

BALLOU R. H. **Logistica Empresarial**, São Paulo: Ed. Atlas, 1993

HUGOS, M. **Essentials of Supply Chain Management**, EUA: John Wiley & Sons, 2003.

PORTER, M. E. **Competitive advantage: creating and sustaining superior Performance**, New York: The Free Press, 1985.

BOWERSOX, D. J.; CLOSS, D. J.; COOPER, M. B. **Supply Chain Logistics Management**, Ed. McGraw-Hill, 2002.

ZNAMENSKY, A. **Heurísticas para o problema de distribuição com estoques geridos pelo fornecedor**, 2006. Dissertação (Doutorado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2006.

TAVARES, S. **Modelo de estoque gerenciado pelo fornecedor (VMI) aplicado ao varejo de materiais de construção no setor de revestimentos cerâmicos**, 2003. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Florianópolis, 2003.

GAPSKI, O. L. **Controle de nível de estoque no setor varejista com base no gerenciamento do inventário pelo fornecedor**, 2003. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Florianópolis, 2003.

WALLER, M. et al., **Vendor-Managed Inventory in the Retail Supply Chain.** Journal of Business Logistics, vol. 20, nº 1, 1999.

GOMAA, H. **Designing Software Product Lines With UML – From Use Cases to Pattern-Based Software Architectures**, Ed. Addison Wesley, 2004.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**, Ed Addison Wesley, 1997.

LARMAN, C. **Applying UML and Patterns**, Ed Prentice Hall, 2002.

NILSSON, J. **Applying Domain-Driven Design and Patterns**, Ed Addison Wesley, 2006.

FOWLER, M. **Refactoring: Improving the Design of Existing Code**, Ed. Addison Wesley,1999.

FOWLER, M. **Inversion of Control Containers and the Dependency Injection Pattern**, 2004. Disponível em <http://martinfowler.com/articles/injection.html>. Acesso em 04/11/2007.

EINI, O. **Inversion of Control and Dependency Injection: Working with Windsor Container**, 2006. Disponível em <http://msdn2.microsoft.com/en-us/library/aa973811.aspx>. Acesso em 04/11/2007.

AMBLER, S **The Agile Unified Process**, 2005. Disponível em <http://www.ambysoft.com/unifiedprocess/agileUP.html>. Acesso em 10/11/2007.

JEZIERSKI, E. A. **Application Architecture for .NET: Designing Applications and Services**, 2002. Disponível em <http://msdn2.microsoft.com/en-us/library/ms954595.aspx>. Acesso em 04/11/2007.

McCAFFERTY, B. **Default ASP.NET Architecture**, 2006. Disponível em <http://devlicio.us/blogs/billy_mccafferty/archive/2006/10/05/_2800_My_2900_>-Default-ASP.NET-Architecture.aspx. Acesso em 04/11/2007.

VENNERS, B. **Design Principles from Design Patterns**, 2005. Disponível em <http://www.artima.com/lejava/articles/designprinciples.html>. Acesso em 04/11/2007.

KRUEGER, C. **Software Product Lines**, 2006. Disponível em <http://www.softwareproductlines.com/index.html>. Acesso em 04/11/2007.

TROTTA, G. **B2B: Why You Need It And How To Achieve It**, 2003. Disponível em < http://www.ebizq.net/topics/b2b/features/3504.html>. Acesso em 11/11/2007.

APÊNDICE A – PLANEJAMENTO

A.1 Estrutura Analítica do Projeto

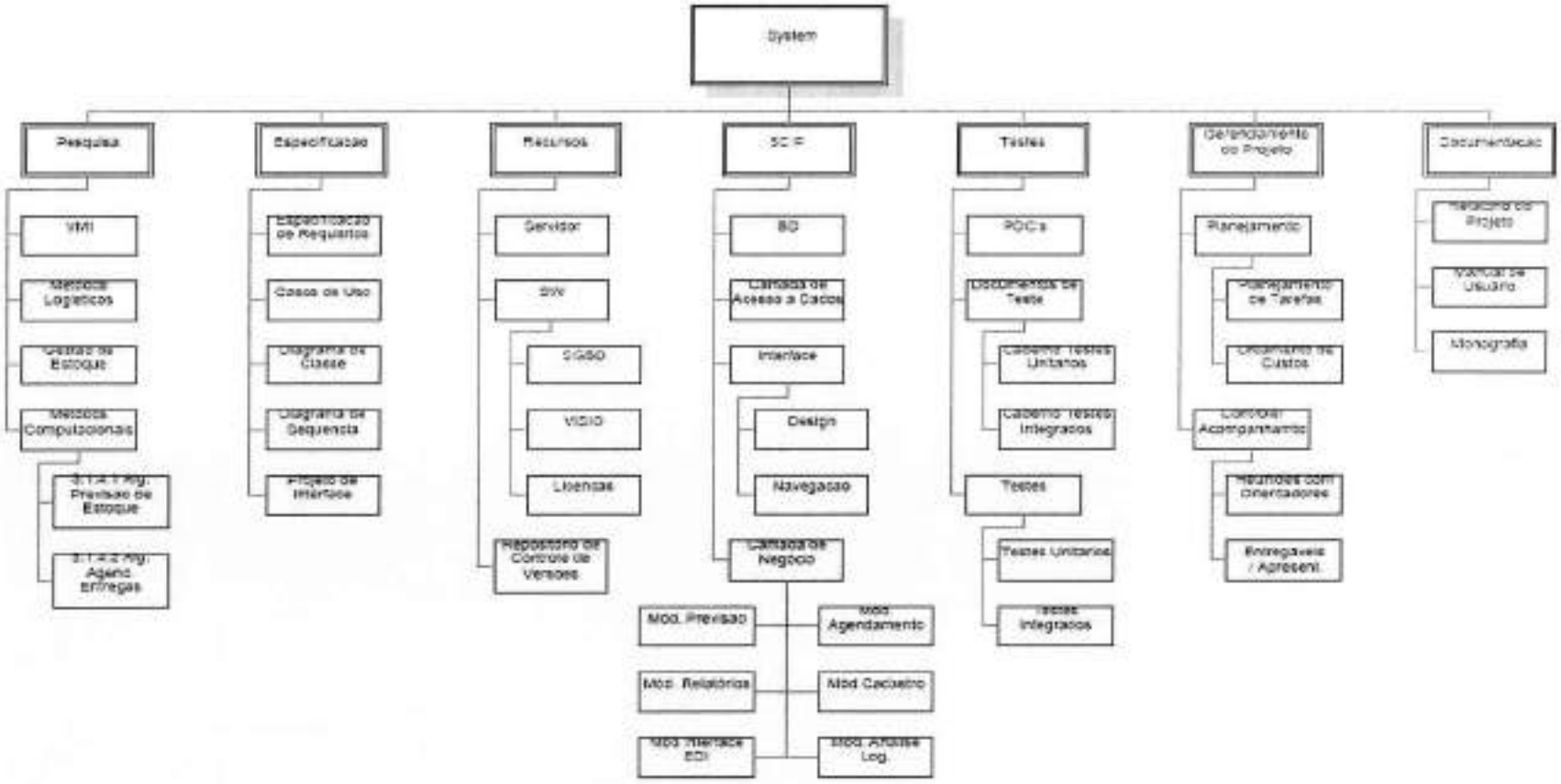
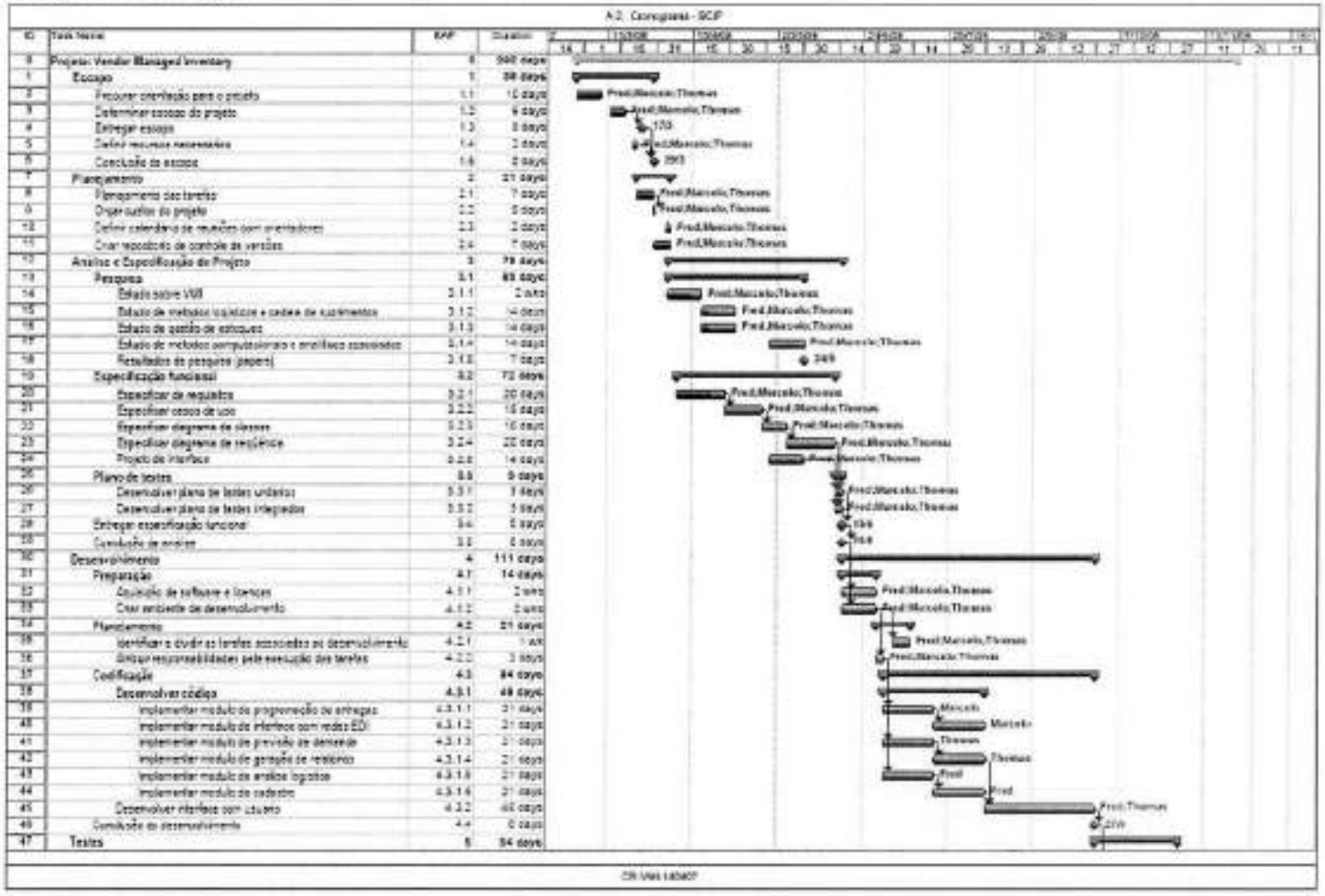
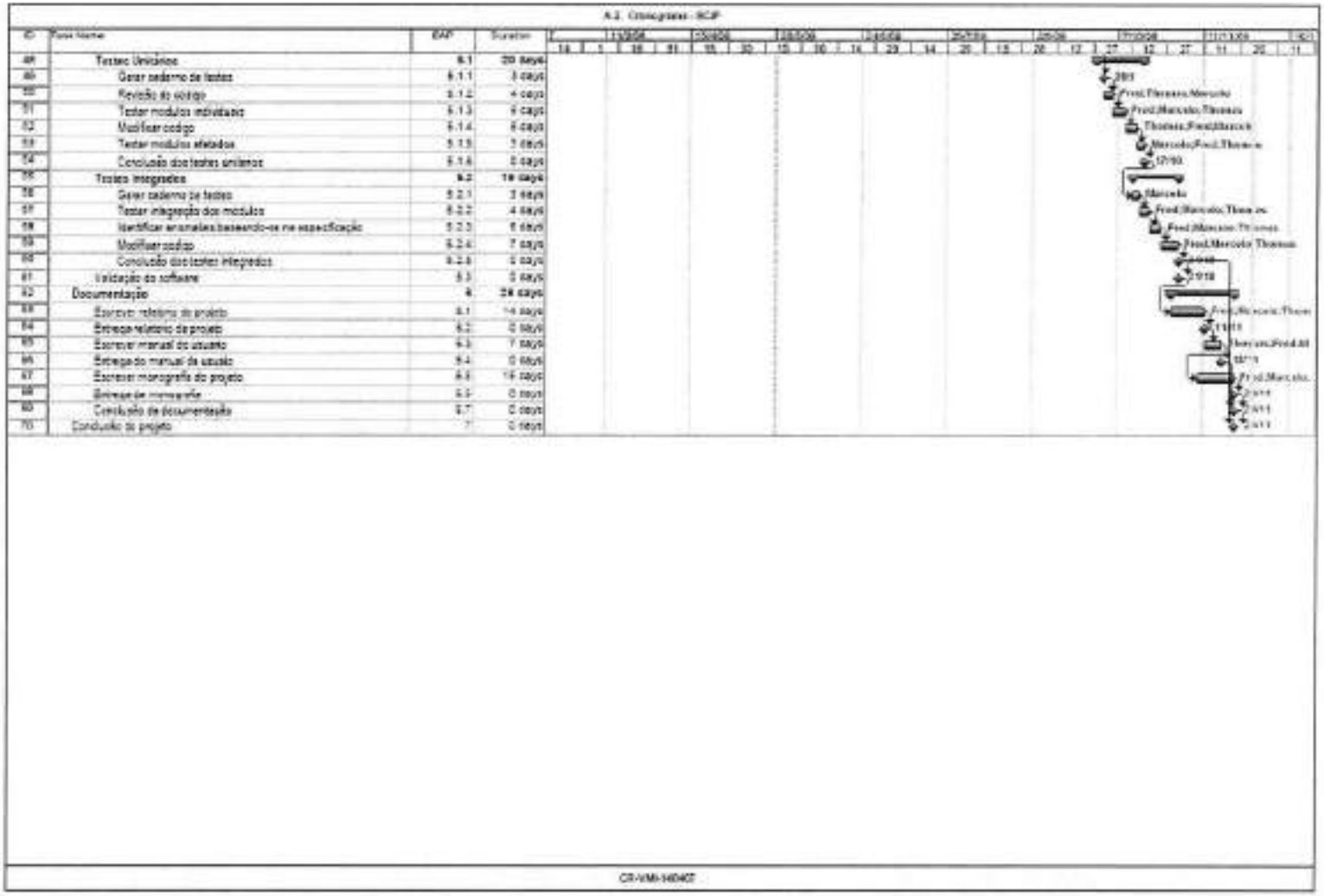


Figura A.1: Estrutura Analítica do Projeto.

A.2 Cronograma do Projeto




A.2. Cronograma - RUP
ID
Task Name
EAP
Duration
Testes Unitários
Gerar caderno de testes
Revisão do código
Testar módulos individuais
Modificar código
Testar módulos afetados
Conclusão dos testes unitários
Testes integrados
Gerar caderno de testes
Testar integração dos módulos
Identificar anomalias baseando-se na especificação
Modificar código
Conclusão dos testes integrados
Validação do software
Documentação
Escrever relatório do projeto
Entrega relatório do projeto
Escrever manual do usuário
Entrega do manual do usuário
Escrever monografia do projeto
Entrega da monografia
Conclusão da documentação
Conclusão do projeto
CR-WM-140407

A.3 Distribuição de Custos do Projeto

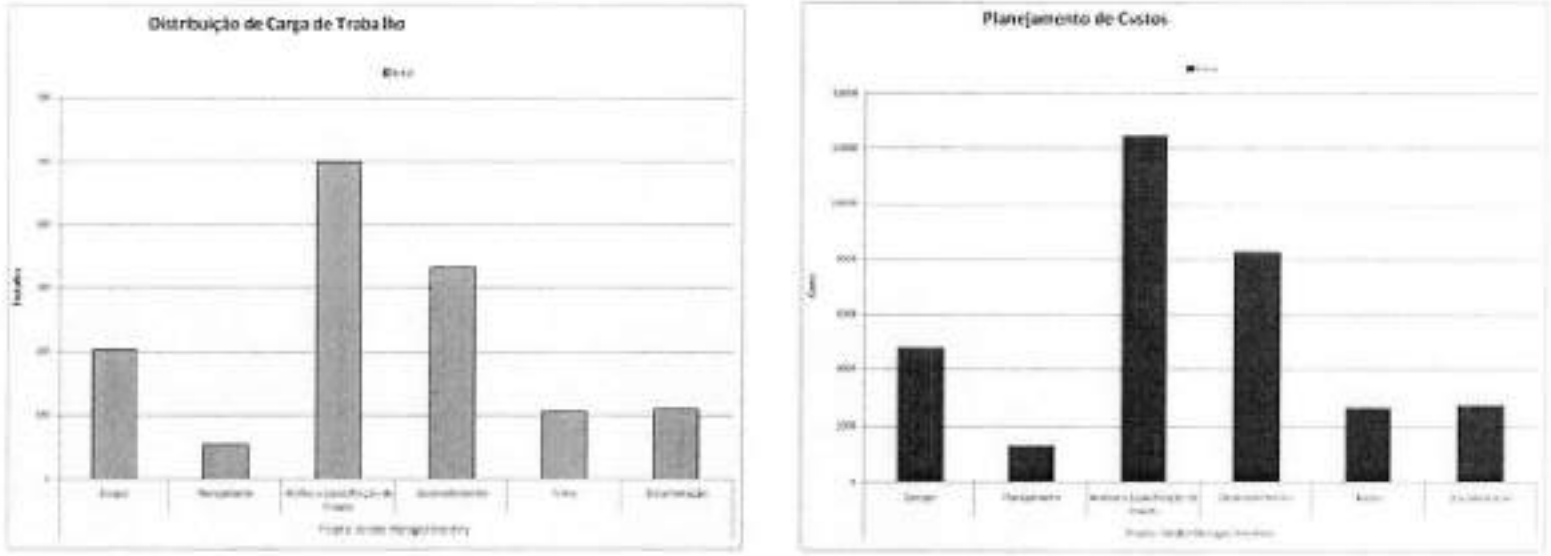


Figura A.2: Distribuição dos custos do projeto por carga de trabalho (esquerda) e custo de HH (direita).

A.4 Alocação da Equipe

Matriz de Alocação de Recursos do Projeto	Recurso				
Atividade	Selma	Reginaldo	Frederico	Thomas	Marcelo
Pesquisa					
Estudo sobre VMI	I	I	A	A	R
Estudo de métodos logísticos e cadeia de suprimentos	I	I	A	R	A
Estudo de gestão de estoques	I	I	A	A	R
Estudo de métodos computacionais e analíticos associados	C	C	R	A	A
Resultados da pesquisa (papers)	I	I	A	R	A
Especificação funcional					
Especificar de requisitos	C	C	A	R	A
Especificar casos de uso	C	C	R	A	A
Especificar diagrama de classes	C	C	R	A	A
Especificar diagrama de seqüência	C	C	A	A	R
Projeto de interface	C	C	A	R	A
Preparação					
Criar repositório de controle de versões	I	I	R	A	A
Aquisição de software e licenças	A	A	A	A	R
Criar ambiente de desenvolvimento	I	I	R	A	A
Identificar e dividir as tarefas associadas ao desenvolvimento	I	I	A	R	A
Desenvolver código					
Implementar módulo de programação de entregas	I	I	A	A	R
Implementar módulo de interface com redes EDI	I	I	A	A	R
Implementar módulo de previsão de demanda	I	I	A	R	A
Implementar módulo de geração de relatórios	I	I	A	R	A
Implementar módulo de análise logística	I	I	R	A	A
Implementar módulo de cadastro	I	I	R	A	A
Desenvolver interface com usuário	I	I	R	A	A
Testes					
Gerar cadernos de testes	I	I	A	A	R
Testes Unitários	I	I	A	R	A
Testes Integrados	I	I	R	A	A
Documentação					
Escrever relatório do projeto	C	C	R	A	A
Escrever manual do usuário	C	I	A	A	R
Escrever monografia do projeto	I	C	A	R	A

Recurso	Alocação (HH)
Selma	50
Reginaldo	50
Frederico	292
Thomas	292
Marcelo	292

Figura A.3: Alocação da equipe de trabalho.

A.5 Matriz de Riscos do Projeto

Avaliacao de Impacto do Risco sobre Objetivos do Projeto							
Probabilidade	Objetivos de Projeto	Definicao	Muito Baixo	Baixo	Moderado	Alto	Muito Alto
			0,05	0,1	0,2	0,4	0,8
0,9	Interface com EDI	Interface com ERPs atraves de redes EDI ou Internet	Interfaces requer pequenos com ajustes e configuracoes especificas	Interface EDI requerem alta customizacao	Interface EDI falha, necessidade de interfaces alternativas	Comunicacao com ERPs nao confiavel/falha	Comunicacao e interface indisponivel
0,7	Cronograma	Tempo previsto para atividades e inicio e fim de cada atividade	Pequenos Atrasos	Atraso de 5%	Atraso de 10%	Atrasos de 20%	Atraso maior que 20%
0,5	Gestao de Estoque	Qualidade da Previsao de estoque e agendamento de entregas	Estoque gerido com poucos quebras do limite minimo de estque	Quebras de limite de estoque minimo freqentes	Estoque em falta ocasionalmente	Estoque nao é gerido adequadamente	Item final é inútil
0,4	Dados de Teste	A validacao do produto final depende da qualidade dos dados disponiveis para testes, no caso os dados sao os niveis de	Dados de teste para poucas areas da industria	Dados de teste especificos para uma area/ pouco historico	Dados de teste nao correspondem a niveis de estoques reais	Dados de teste nao possibilitam validacao do SW	Impossibilidade de simular os dados para teste
0,4	Interface ERP	Funcionalidade de comunicacao e troca de dados com ERPs do cliente e fornecedor	ERP simulado desvia um pouco de ERPs reais	ERP simulaod simplifica ERP real	ERP siulado nao corresponde a ERP real, dificultando aplicacao real do	Falhas na simulacao do ERP	Impossibilidade de sequer simular o ERP
0,3	Escopo	Escopo do projeto, no caso cumprir a funcao de intergacao e controle de estoque	Projeto cumpre escopo com pequenas simplificacoes	Areas do escopo afetadas/ limitadas	Areas do escopo nao cumpridas	Reducao de Escopo degrada funcionalidade principal	Item final é inútil, nao adequado ao escopo
0,1	Recursos	Disponibilidade de recursos	recursos adquiridos com pouco atraso para inicio do desenvolvimento	atraso em um recurso especifico	atraso consideravel de diversos recursos	projeto desenvolvido com recursos inferiores e precarios	ecursos indisponiveis para desenvolvimento do projeto

Figura A.4: Matriz de riscos do projeto.

Pontuacao de Risco para cada Risco do Projeto					
Probabilidade	**Pontuacao de Risco (Probabilidade X impacto)**				
0,9	0,045	0,09	0,18	0,36	0,72
0,7	0,035	0,07	0,14	0,28	0,56
0,5	0,025	0,05	0,1	0,2	0,4
0,4	0,02	0,04	0,08	0,16	0,32
0,3	0,015	0,03	0,06	0,12	0,24
0,1	0,005	0,01	0,02	0,04	0,08
	0,05	**0,1**	**0,2**	**0,4**	**0,8**
	Impacto do Risco sobre Objetivos				

Figura A.5: Impacto de riscos sobre objetivos do projeto

APÊNDICE B – CASOS DE USO

Caso de uso 1: Cadastro de empresas (fornecedora ou cliente)

Descrição: Este caso de uso descreve o processo de cadastro de empresas no sistema SCIP.
Evento iniciador: Abertura da tela de cadastro de empresas no aplicativo.
Atores: usuário administrador do sistema (iniciador)
Categoria de reuso: kernel
Pré-condição: usuário autenticado no sistema, com perfil de administrador.
Seqüência de eventos:

1. Usuário seleciona, através do menu do aplicativo, a tela de cadastro de empresas.
2. Sistema exibe a tela de cadastro de empresas, com formulário para cadastrar uma empresa.
3. Usuário preenche o formulário com dados da empresa.
4. Usuário especifica a relação da empresa cadastrada com outras empresas já cadastradas.
5. Usuário confirma o cadastro da empresa.
6. Sistema empacota estes dados, executa validação dos mesmos e os registra em sua base de dados.

Pós-condição: empresa cadastrada no sistema
Extensões:

1. Usuário altera cadastro de uma empresa: o sistema exibe o formulário com os dados registrados para empresa selecionada (passo 3).
2. Usuário remove cadastro de uma empresa: o sistema exibe o formulário com os dados registrados para empresa selecionada e usuário escolhe opção de remoção da empresa (passo 2).
3. Sistema não autoriza (não valida) o cadastro: o sistema cancela a cadastro e retorna à tela de cadastro (passo 2).
4. Erro de temporização na comunicação com a base de dados: o sistema cancela o cadastro e retorna mensagem de erro ao usuário, retornando à tela de cadastro (passo 2).

Inclusões:

Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 2: Cadastro de Itens (produtos)

Descrição: Este caso de uso descreve o processo de cadastro de itens (produtos).
Evento iniciador: Abertura da tela de cadastro de itens no aplicativo.
Atores: usuário administrador do sistema (iniciador)
Categoria de reuso: kernel
Pré-condição: usuário autenticado no sistema, com perfil de administrador.
Seqüência de eventos:

1. Usuário seleciona, através do menu do aplicativo, a tela de cadastro de itens para uma empresa.
2. Sistema exibe a tela de cadastro de itens, com menu para escolha da empresa a qual o produto será relacionado.

3. Usuário seleciona a empresa a qual o produto pertence
4. Sistema habilita formulário para cadastro do produto.
5. Usuário preenche o formulário com dados do item.
6. Usuário confirma o cadastro do item.
7. Sistema empacota estes dados, executa validação dos mesmos e os registra em sua base de dados.

Pós-condição: item da empresa cadastrado no sistema

Extensões:

1. Usuário altera cadastro de um item: o sistema exibe o formulário com os dados registrados para o item selecionado (passo 3).
2. Usuário remove cadastro de um item: o sistema exibe o formulário com os dados registrados para o item selecionado e usuário escolhe opção de remoção do item (passo 2).
3. Sistema não autoriza (não valida) o cadastro: o sistema cancela a cadastro e retorna à tela de cadastro (passo 2).
4. Erro de temporização na comunicação com a base de dados: o sistema cancela o cadastro e retorna mensagem de erro ao usuário, retornando à tela de cadastro (passo 2).

Inclusões:

Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 3: Criação de conta de usuário

Descrição: Este caso de uso descreve o processo de cadastro de um usuário no sistema SCIP.

Evento iniciador: Abertura da tela de cadastro de usuários no aplicativo.

Atores: usuário administrador do sistema (iniciador)

Categoria de reuso: kernel

Pré-condição: usuário autenticado no sistema, com perfil de administrador.

Seqüência de eventos:

1. Administrador seleciona, através do menu do aplicativo, a tela de cadastro de usuários.
2. Sistema exibe a tela de cadastro de usuários, com formulário para cadastrar um usuário.
3. Administrador preenche o formulário com dados do usuário.
4. Administrador especifica o perfil de acesso do usuário.
5. Administrador confirma o cadastro do usuário.
6. Sistema empacota estes dados, executa validação dos mesmos e os registra em sua base de dados.

Pós-condição: item cadastrado no sistema

Extensões:

1. Administrador altera cadastro de um usuário: o sistema exibe o formulário com os dados registrados para o usuário selecionado (passo 3).
2. Administrador remove cadastro de um usuário: o sistema exibe o formulário com os dados registrados para o usuário selecionado e administrador escolhe opção de remoção do usuário (passo 2).
3. Sistema não autoriza (não valida) o cadastro: o sistema cancela a cadastro e retorna à tela de cadastro (passo 2).

4. Erro de temporização na comunicação com a base de dados: o sistema cancela o cadastro e retorna mensagem de erro ao usuário, retornando à tela de cadastro (passo 2).

Inclusões: Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 4: Cadastro/Configuração de perfis de acesso para o sistema.

Descrição: Este caso de uso descreve o processo de cadastro de perfis de acesso para usuários do sistema SCIP.
Evento iniciador: Abertura da tela de cadastro de perfis no aplicativo.
Atores: usuário administrador do sistema (iniciador)
Categoria de reuso: kernel
Pré-condição: usuário autenticado no sistema, com perfil de administrador.
Seqüência de eventos:

1. Administrador seleciona, através do menu do aplicativo, a tela de cadastro de perfis.
2. Sistema exibe a tela de cadastro de perfis, com formulário para cadastrar um perfil.
3. Administrador preenche o formulário com dados do perfil.
4. Administrador especifica as permissões de acesso conferidas ao perfil.
5. Administrador confirma o cadastro do perfil.
6. Sistema empacota estes dados, executa validação dos mesmos e os registra em sua base de dados.

Pós-condição: item cadastrado no sistema
Extensões:

1. Administrador altera cadastro de um perfil: o sistema exibe o formulário com os dados registrados para o perfil selecionado (passo 3).
2. Administrador remove cadastro de um perfil: o sistema exibe o formulário com os dados registrados para o perfil selecionado e administrador escolhe opção de remoção do perfil (passo 2).
3. Sistema não autoriza (não valida) o cadastro: o sistema cancela a cadastro e retorna à tela de cadastro (passo 2).
4. Erro de temporização na comunicação com a base de dados: o sistema cancela o cadastro e retorna mensagem de erro ao usuário, retornando à tela de cadastro (passo 2).

Inclusões:
Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 5: Autenticação de usuário no sistema (*Login*)

Descrição: Este caso de uso descreve o processo de autenticação de usuários do sistema SCIP.
Evento iniciador: Abertura do aplicativo do sistema ou tentativa de acesso a qualquer tela do aplicativo.
Atores: qualquer usuário cadastrado no sistema (iniciador)
Categoria de reuso: kernel
Pré-condição: iniciar o aplicativo do sistema SCIP.
Seqüência de eventos:

1. Usuário inicia o aplicativo de SCIP ou tenta acessar alguma tela do mesmo diretamente.

2. Sistema verifica se o usuário está autenticado.
3. Caso não esteja autenticado, tela de *login* (autenticação) é proposta ao usuário.
4. Usuário preenche campos com nome de usuário e senha.
5. Sistema executa autenticação do usuário e em caso de validação, dá acesso para usuário à aplicação.

Pós-condição: usuário autenticado no sistema

Extensões:

1. Caso esteja autenticado, leva o usuário direto para a tela principal do aplicativo, ou para a tela para o qual o usuário solicitou acesso.
2. Sistema executa autenticação do usuário e em caso não validação, redireciona usuário para tela de *login* novamente.
3. Erro de temporização na comunicação com a base de dados: o sistema cancela o cadastro e retorna mensagem de erro ao usuário, retornando à tela de *login* (passo 2).

Caso de uso 6: Executa *logout* de usuário

Descrição: Este caso de uso descreve o processo de execução de *logout* de usuários do sistema SCIP.

Evento iniciador: Escolha da opção de logout, presente em todas as telas do aplicativo do sistema.

Atores: qualquer usuário do sistema (iniciador)

Categoria de reuso: kernel

Pré-condição: Estar autenticado no sistema

Seqüência de eventos:

1. Usuário opta por executar *logout* do sistema
2. Sistema apaga sessão do usuário.
3. Sistema redireciona usuário para a tela de *login*.

Pós-condição: Usuário não autenticado no sistema

Extensões:

1. Erro de temporização na comunicação com a base de dados: o sistema cancela o cadastro e retorna mensagem de erro ao usuário.

Caso de uso 7: Cadastro de um item de planejamento

Descrição: Permite que usuário cadastre um novo item de planejamento no sistema.

Evento iniciador: Abertura da tela de cadastro de item de planejamento no aplicativo.

Atores: usuário administrador do sistema.

Categoria de Reuso: Kernel

Pré-condição:

1. Usuário autenticado no sistema;
2. Tela de cadastro de item de planejamento aberta.

Seqüência de eventos:

1. Usuário seleciona, através do menu do aplicativo, a tela de cadastro de itens de planejamento.
2. Sistema exibe a tela de cadastro de itens de planejamento, com formulário para cadastrar um item de planejamento.

3. Usuário preenche o formulário com dados do item de planejamento.
4. Usuário cadastra um Item de fornecimento novo para este item de planejamento.
5. Usuário confirma o cadastro do item de planejamento.
6. Sistema empacota estes dados, executa validação dos mesmos e os registra em sua base de dados.

Pós-condição: Item de planejamento é criado. Sistema retorna para tela de configuração do item atual.

Extensões:

1. Usuário altera cadastro de um item de planejamento: o sistema exibe o formulário com os dados registrados para o item selecionado (passo 3).
2. Usuário remove cadastro de um item de planejamento: o sistema exibe o formulário com os dados registrados para o item selecionado e usuário escolhe opção de remoção do item (passo 2).
3. Sistema não autoriza (não valida) o cadastro: o sistema cancela a cadastro e retorna à tela de cadastro (passo 2).
4. Erro de temporização na comunicação com a base de dados: o sistema cancela o cadastro e retorna mensagem de erro ao usuário, retornando à tela de cadastro (passo 2).

Inclusões:

1. Cadastro de um novo item de fornecimento para o item de planejamento em questão. (Caso de uso 25)

Caso de uso 8: Cadastro de um item de fornecimento

Descrição: Permite que usuário cadastre um novo item de fornecimento em um item de planejamento no sistema.

Evento iniciador: Tela de cadastro de item de planejamento aberta; usuário seleciona a inserção de um novo item de fornecimento.

Atores: usuário administrador do sistema.

Categoria de Reuso: Kernel

Pré-condição:

1. Usuário autenticado no sistema;
2. Tela de cadastro de item de fornecimento aberta.

Seqüência de eventos:

1. Usuário seleciona, através da tela de cadastro do item de planejamento, a tela de cadastro de itens de fornecimento.
2. Sistema exibe a tela de cadastro de itens de fornecimento, com formulário para cadastrar um item de fornecimento.
3. Usuário preenche o formulário com dados do item de fornecimento.
4. usuário relaciona empresas participantes e seus papéis.
5. Usuário seleciona os produtos que cada empresa compra e vende estabelecendo a relação produto do vendedor e comprador
6. Usuário confirma o cadastro do item de fornecimento.
7. Sistema empacota estes dados, executa validação dos mesmos e os registra como parte do item de planejamento.

Pós-condição: Item de fornecimento é criado. Sistema retorna para tela de configuração do item de planejamento atual.

Extensões:

1. Usuário altera cadastro de um item de fornecimento: o sistema exibe o formulário com os dados registrados para o item selecionado (passo 3).
2. Usuário remove cadastro de um item de fornecimento: o sistema exibe o formulário com os dados registrados para o item selecionado e usuário escolhe opção de remoção do item (passo 2).
3. Sistema não autoriza (não valida) o cadastro: o sistema cancela a cadastro e retorna à tela de cadastro (passo 2).
4. Erro de temporização na comunicação com a base de dados: o sistema cancela o cadastro e retorna mensagem de erro ao usuário, retornando à tela de cadastro (passo 2).
5. Usuário cadastra um novo produto para uma empresa na própria tela de cadastro de item de fornecimento.

Inclusões: nenhuma.

Caso de uso 9: Configurar regras de negócio de um item de planejamento

Descrição: Permite que usuário configure parâmetros das regras de negocio de um item de um fornecedor para um cliente.
Evento iniciador: Usuário seleciona item a ser configurado.
Atores: usuário administrador do sistema ou usuário funcionário do fornecedor
Categoria de Reuso: Kernel
Pré-condição:

1. Usuário autenticado no sistema;
2. Tela de configuração de itens aberta.

Seqüência de eventos:

1. Usuário seleciona através um clique o item de planejamento desejado;
2. Sistema exibe os parâmetros possiveis a serem configurados e suas configurações atuais;
3. Usuário modifica e insere parâmetros nas configurações;
4. Usuário seleciona, através de um clique, a opção de salvamento;
5. Sistema exibe informação que parâmetros foram salvos;

Pós-condição: Parâmetros das regras de negócio são modificados ou inseridos no sistema. Sistema retorna para tela de configuração do item atual.
Extensões:

1. Usuário fecha tela do navegador ou aperta botão de cancelamento. Nenhuma alteração é salva nos parâmetros das regras de negócio do item;
2. Erro de temporização na comunicação com o sistema: usuário levou tempo demais para realizar modificações, o sistema cancela as modificações feitas e não salvas.

Inclusões: nenhuma.

Caso de uso 10: Confirmar proposições de ordem

Descrição: Permite que usuário aprove e confirme proposição de entrega de produto.
Evento iniciador: Usuário seleciona proposição de entrega.
Atores: usuário administrador do sistema ou usuário funcionário do fornecedor

Categoria de Reuso: Kernel
Pré-condição:

1. Usuário autenticado no sistema;
2. Tela principal de reaprovisionamento aberta, mostrando lista de itens que o usuário tem direito a visualizar e interagir.

Seqüência de eventos:

1. Usuário seleciona através um clique a proposição de ordem a ser confirmada;
2. Sistema exibe a proposição de ordem e seus parâmetros (ex: data de entrega, quantidade, etc.);
3. Usuário modifica e insere parâmetros na proposição;
4. Usuário seleciona, através de um clique, a opção de confirmação;
5. Sistema exibe informação que proposição foi confirmada e exportada para sistema ERP do fornecedor;

Pós-condição: Proposição de entrega é transformada em entrega confirmada e sistema SCIP exporta mensagem *"Order Create"* para sistema ERP do fornecedor. Sistema retorna para tela reaprovisionamento de itens mostrando todas as proposições de entrega e entregas confirmadas.
Extensões:

1. Usuário recusa a proposição de ordem. Ordem é cancelada.
2. Usuário fecha tela do navegador ou aperta botão de cancelamento logo após o passo numero '1'. Neste caso nenhuma modificação é realizada na proposição de entrega e a mesma não é confirmada. Caso o usuário tenha apertado o botão de cancelamento o sistema retorna para tela de reaprovisionamento, mostrando lista de itens que o usuário tem direito a visualizar e interagir.
3. Erro de temporização na comunicação com o sistema: usuário levou tempo demais para realizar operação, o sistema cancela as modificações feitas e não salvas.

Inclusões: Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 11: Criação de proposição de ordem

Descrição: Permite que usuário crie proposição de entrega de produto.
Evento iniciador: Usuário seleciona dia em que deseja adicionar proposição de entrega.
Atores: usuário administrador do sistema ou usuário funcionário do fornecedor
Categoria de Reuso: Kernel
Pré-condição:

1. Usuário autenticado no sistema;
2. Tela principal de reaprovisionamento aberta, mostrando lista de itens que o usuário tem direito a visualizar e interagir.

Seqüência de eventos:

1. Usuário seleciona através um clique o dia em que deseja adicionar proposição de ordem;
2. Sistema exibe todas as proposições de ordem existentes para o dia, todas as ordens confirmadas para o dia e um resumo de seus parâmetros (ex: data de entrega, quantidade, etc.);
3. Usuário pressiona botão de adição de proposição de ordem;

4. Sistema mostra tela de adição de proposição de ordem com campos a serem preenchidos;
5. Usuário preenche todos os campos necessários para a criação da ordem;
6. Usuário seleciona, através de um clique, a opção de confirmação de criação de proposição;
7. Sistema exibe informação que proposição foi criada;

Pós-condição: Proposição de entrega é criada. Sistema retorna para tela reaprovisionamento de itens mostrando todas as proposições de entrega e entregas confirmadas.

Extensões:

1. Usuário fecha tela do navegador ou aperta botão de cancelamento logo após o passo numero '4'. Neste caso nenhuma proposição de entrega é criada no sistema. Caso o usuário tenha apertado o botão de cancelamento o sistema retorna para tela de reaprovisionamento, mostrando lista de itens que o usuário tem direito a visualizar e interagir.
2. Erro de temporização na comunicação com o sistema: usuário levou tempo demais para realizar operação, o sistema cancela as modificações feitas e não salvas.
   Inclusões: Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 12: Modificação de ordem (proposta ou já confirmada)

Descrição: Permite que o usuário altere as programações de entregas (ainda propostas ou já confirmadas) do sistema.

Evento iniciador: Usuário seleciona a ordem a ser alterada.

Atores: usuário administrador do sistema ou usuário funcionário do fornecedor

Categoria de Reuso: Kernel

Pré-condição:

1. Usuário autenticado no sistema;
2. Tela principal de reaprovisionamento aberta, mostrando lista de itens que o usuário tem direito a visualizar e interagir.

Seqüência de eventos:

1. Usuário seleciona através um clique a ordem a ser modificada;
2. Sistema exibe a ordem e seus parâmetros (ex: data de entrega, quantidade, etc.);
3. Usuário modifica e insere parâmetros;
4. Usuário seleciona, através de um clique, a opção de salvar alterações;
5. Sistema exibe informação que ordem foi alterada. Se a ordem já tiver sido exportada para sistema ERP do fornecedor então mensagem de modificação de ordem é enviada para ERP informando modificações.

Pós-condição: Ordem (proposição de entrega não confirmada ou já confirmada) é modificada. Caso proposição já tenha sido exportada para sistema ERP então mensagem "Order Change" contendo dados de modificação é enviada para sistema ERP.

Extensões:

1. Usuário fecha tela do navegador ou aperta botão de cancelamento logo após o passo numero '2'. Neste caso nenhuma modificação é

realizada no sistema. Caso o usuário tenha apertado o botão de cancelamento o sistema retorna para tela de reaprovisionamento, mostrando lista de itens que o usuário tem direito a visualizar e interagir.

2. Erro de temporização na comunicação com o sistema: usuário levou tempo demais para realizar operação, o sistema cancela as modificações feitas e não salvas.

Inclusões: Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 13: Envio de confirmação de recebimento de mercadoria

Descrição: Confirmação de recebimento de produto pelo cliente e inserida no sistema

Evento iniciador: Sistema ERP do cliente envia mensagem informando que cliente recebeu produto

Atores: Sistema ERP do cliente ou funcionário (administrador, funcionário do cliente ou funcionário do fornecedor).

Categoria de Reuso: Opcional

Pré-condição:

1. Sistema do cliente conectado com SCIP
   OU:
2. Funcionário autenticado no sistema e tela de inserção de recebimento de mercadoria aberta.

Seqüência de eventos:

1. Sistema do cliente envia mensagem contendo informação de recebimento de mercadoria;
2. Sistema SCIP processa a informação de recebimento de mercadoria;
3. Sistema SCIP notifica fornecedor.

Pós-condição: Sistema SCIP apresenta informação de recebimento de mercadoria na tela principal de reaprovisionamento.

Extensões:

1. Caso fornecedor já tenha recebido a mensagem de recebimento de mercadoria, o passo 3 não é executado;
2. Informação de confirmação de recebimento de mercadoria pelo cliente é feito manualmente por um usuário. Neste caso, a pré-condição número '2' deve estar satisfeita. Usuário adiciona informação de recebimento de mercadoria e pressiona o botão 'salvar'.

   Inclusões: Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 14: Recebimento confirmação de ordem proposta *Order Response*

Descrição: Sistema ERP do fornecedor envia confirmação de entrega apos a criação de uma ordem ou modificação de uma ordem em seu sistema ERP (uma ordem é criada quando ERP recebe mensagem "Order Create" do sistema SCIP).

Evento iniciador: Sistema ERP do fornecedor envia mensagem "Order Response"

Atores: Sistema ERP do fornecedor

Categoria de Reuso: Kernel

Pré-condição:

1. Sistema do fornecedor conectado com SCIP

Seqüência de eventos:

1. Sistema do fornecedor envia mensagem de confirmação de previsão de entrega "Order Response"
2. Sistema SCIP recebe informação. Se dados presentes na confirmação forem diferentes da informação presente no momento de exportação (ex: sistema ERP do fornecedor modificou data ou quantidade de produto a ser entregue), sistema SCIP atualiza dados e recalcula programações futuras;

Pós-condição: Sistema SCIP adiciona no seu banco de dados informações presentes na confirmação da ordem (ex: são adicionados na ordem já existente no sistema o preço, o número de pedido no sistema ERP do fornecedor, etc.).

Extensões: nenhuma

Inclusões: nenhuma.

Caso de uso 15: Envio de aviso de expedição *Ship Notice*

Descrição: Sistema ERP do fornecedor envia aviso de expedição de produto

Evento iniciador: Sistema ERP do fornecedor envia mensagem "Ship Notice"

Atores: Sistema ERP do fornecedor

Categoria de Reuso: Kernel

Pré-condição:

1. Sistema do fornecedor conectado com SCIP

Seqüência de eventos:

1. Sistema do fornecedor envia mensagem de aviso de expedição de produto.
2. Sistema SCIP recebe informação. A proposição de entrega confirmada recebe status de "em transito".
3. Sistema SCIP notifica cliente

Pós-condição: Sistema SCIP adiciona no seu banco de dados informações presentes no aviso de expedição (ex: são adicionados na proposição de entrega confirmada dados sobre a quantidade exata de produto em transito, numero do veiculo de transporte, etc.). Sistema SCIP mostra informações sobre mercadoria "em transito" na tela principal de reaprovisionamento.

Extensões:

1. Caso cliente já tenha recebido a mensagem de recebimento de mercadoria, o passo 3 não é executado;

Inclusões: nenhuma.

Caso de uso 16: Solicitação de envio de *Digest*

Descrição: Sistema SCIP cria mensagem com resumo de programações de entrega e o envia para sistema do cliente

Evento iniciador: Timer do sistema envia pedido de criação de "Digest" para um determinado cliente.

Atores: Timer

Categoria de Reuso: Opcional

Pré-condição: nenhuma

Seqüência de eventos:

1. Timer do sistema envia pedido de criação de "Digest" para um determinado cliente;
2. Sistema SCIP cria mensagem com resumo de programações de entrega;
3. Mensagem é encaminhada para sistema do cliente.

Pós-condição: Sistema SCIP realiza "log" indicando que mensagem foi enviada para cliente.
Extensões: nenhuma
Inclusões: nenhuma.

Caso de uso 17: Solicitação realização de diagnóstico

Descrição: Timer envia para sistema SCIP pedido de diagnostico interno, a fim de se descobrir possíveis erros ou incoerências nos sistema (ex: detecção de não recebimento de dados de nível de estoque)
Evento iniciador: Timer do sistema envia pedido de diagnostico interno
Atores: Temporizador
Categoria de reuso: Kernel
Pré-condição: nenhuma
Seqüência de eventos:

1. Timer do sistema envia pedido de criação de realização de diagnostico;
2. Sistema SCIP realiza rotina de diagnostico e cria lista com problemas internos;
3. Lista com problemas é encaminhada para todos os usuários administradores.

Pós-condição: Sistema SCIP realiza "log" listando todos os erros encontrados.
Extensões: nenhuma
Inclusões: nenhuma.

Caso de uso 18: Visualização de relatórios

Descrição: Este caso de uso descreve a visualização de relatórios por usuários do sistema de acordo com seu perfil de acesso.
Evento iniciador: Solicitação de visualização de relatórios
Atores: Qualquer usuário cadastrado no sistema (funcionário)
Categoria de reuso: opcional
Pré-condição: 1. Funcionário autenticado no sistema
Seqüência de eventos:

1. Funcionário solicita visualização de relatórios
2. Funcionário seleciona um dos relatórios disponíveis no sistema de acordo com o perfil de acesso do funcionário.
3. Sistema gera o relatório selecionado e o mostra para o usuário.
4. O funcionário solicita salvar o documento
5. Usuário seleciona pasta onde relatório deve ser salvo, e o relatório é salvo.

Pós-condição: relatório gerado
Alternativas:

1. Funcionário fecha relatório voltando ao menu de relatórios.

Extensão:

Não existe.

Inclusão: Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 19: Inserir *forecast*

Descrição: Descreve a inserção de dados de *forecast* de um produto(s) da empresa cliente no sistema

Atores: sistema do cliente

Categoria de reuso: opcional

Pré-condição: 1. Sistema do cliente conectado com SCIP

Seqüência de eventos:

1. Sistema cliente envia dados de *forecast*
2. Dados de *forecast* são recebidos e sistema confirma o recebimento dos dados.
3. Sistema agenda replanejamento de ordens futuras;

Pós-condição: dados de *forecast* inseridos na base de dados

Extensão:

1. Funcionário insere *forecast* manualmente

Caso de uso 20: Inserir *forecast* manualmente

Descrição: Descreve a inserção de dados de *forecast* de um produto(s) da empresa cliente no sistema manualmente

Atores: qualquer usuário cadastrado no sistema (funcionário)

Categoria de reuso: opcional

Pré-condição: Funcionário autenticado no sistema e tela de inserção de *forecast* aberta

Seqüência de eventos:

1. Funcionário requisita a inserção de *forecast* no sistema manualmente
2. Sistema mostra empresas cliente para as quais o funcionário tem direito de acesso
3. Usuário seleciona uma empresa específica
4. Sistema mostra itens de planejamento, aos quais o funcionário tem acesso contendo o *forecast* atual para cada um dos produtos
5. Usuário altera *forecast* de consumo para um produto especifico.
6. Usuário salva modificações e sistema atualiza *forecast* dos produtos;
7. Sistema agenda replanejamento de ordens futuras;

Pós-condição: dados de *forecast* inseridos na base de dados

Extensão:

1. Funcionário seleciona inserção de *forecast* por meio de *upload* de arquivo
2. Funcionário seleciona o arquivo a ser carregado no sistema de seu *desktop*
3. Usuário confirma a operação e dados contidos no arquivo são atualizados para empresas e produtos, de acordo com perfil de acesso do funcionário.

Inclusão: Caso de uso 5: Autenticação de usuário no sistema (Login)

Caso de uso 21: Inserir nível de estoque

Descrição: Descreve a inserção de dados de nível de estoque de produto(s) da empresa cliente no sistema
Atores: sistema cliente
Categoria de reuso: kernel
Pré-condição: Sistema do cliente conectado com SCIP
Seqüência de eventos:

1. Sistema cliente envia dados de nível de estoque
2. Dados de estoque são recebidos e sistema confirma o recebimento dos dados.
3. Dados de nível de estoque são inseridos na base de dados
4. Sistema agenda replanejamento de ordens futuras;

Pós-condição: dados de nível de estoque inseridos na base de dados
Alternativa:

1. Funcionário insere nível de estoque manualmente

Extensões: não há
Inclusão: não há

Caso de uso 22: Inserir nível de estoque manualmente

Descrição: Descreve a inserção de dados de nível de estoque de um produto(s) da empresa cliente no sistema manualmente
Atores: qualquer usuário cadastrado no sistema (funcionário)
Categoria de reuso: opcional
Pré-condição: Funcionário autenticado no sistema e tela de inserção de nível de estoque aberta
Seqüência de eventos:

1. Funcionário requisita a inserção de nível de estoque no sistema manualmente
2. Sistema mostra empresas cliente para as quais o funcionário tem direito de acesso
3. Usuário seleciona uma empresa especifica
4. Sistema mostra itens de planejamento, aos quais o funcionário tem acesso contendo o nível de estoque histórico atual para cada um dos produtos da empresa
5. Usuário altera nível de estoque de consumo para um produto especifico para uma data específica.
6. Usuário salva modificações;
7. Sistema agenda replanejamento de ordens.

Pós-condição: dados de nível de estoque inseridos na base de dados
Extensão:

1. Funcionário seleciona inserção de nível de estoque por meio de *upload* de arquivo
2. Funcionário seleciona o arquivo a ser carregado no sistema de seu *desktop*

3. Usuário confirma a operação e dados de estoque contidos no arquivo são atualizados para empresas e produtos, de acordo com perfil de acesso do funcionário.

Inclusão: não há

Caso de uso 23: Envio de dados de *tracking*

Descrição: Este caso de uso descreve o envio de dados de track&trace do produto despachado do fornecedor para o cliente.
Evento iniciador: *tracker* envia dados de posição a sensores existentes em *warehouses* e centro de distribuição logistica
Categoria de reuso: opcional
Atores: *tracker*
Pré-condição: 1.produto previamente cadastrado no sistema com status enviado.
2.*Tracker* em alcance de um sensor
Seqüência de eventos:

1. *Tracker* envia dados de identificação do produto ao sistema
2. *Tracker* envia dados do produto e sua posição para o sistema
3. Sistema atualiza a posição atual do produto enviado.

Pós-condição: dados de *track&trace* enviados e registrados
Extensão: Não existe.
Inclusão: não há

Caso de uso 24: Consultar tracking de produtos enviados

Descrição: Este caso de uso descreve a apresentação de *track&trace* do produto despachado do fornecedor para o cliente ao funcionário
Evento iniciador: Funcionário requisita visualizar *tracking* do produto
Categoria de reuso: opcional
Atores: funcionário
Pré-condição:

1. Produto previamente cadastrado no sistema com status enviado.
2. Funcionário autenticado no sistema e com a tela de estoque do produto aberta, mostrando as quantidades de cada produto em transito

Seqüência de eventos:

1. Funcionário requisita visualização do *tracking* do produto
2. Sistema mostra posição atual do produto, data de saída e data esperada de chegada.
3. Funcionário encerra a operação

Pós-condição: tela de estoque dos produtos da empresa aberta.
Extensão:
Não existe.
Inclusão: não há

APÊNDICE C – DIAGRAMA DE CLASSES

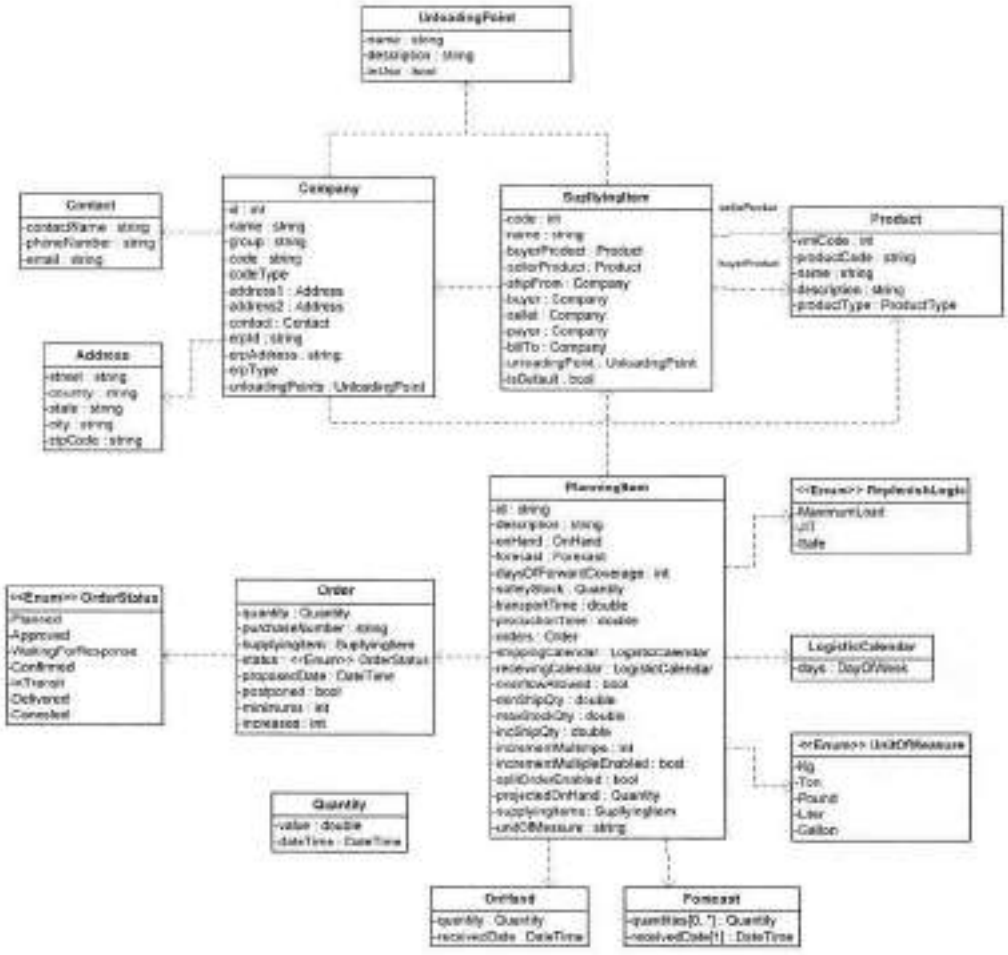


Figura C.1: Diagrama de classes do sistema.

C.1 Company

Tabela C.1: Classe Company.

Atributo	Descrição
name	Nome da empresa.
code	Identificador da empresa, esta informação é utilizada para identificar a empresa dentro do sistema ERP dela.
codeType	Identifica o tipo de código: DUNS ou código interno. O DUNS é um código único de nove dígitos usado para identificar mais de 100 milhões de empresas ao redor do mundo.
group	Grupo a qual a empresa pertence. Este atributo permite agrupar empresas sob uma mesma sigla, código ou nome.
address1	Endereço primário da empresa.
address2	Endereço secundário da empresa.
erpAddress	Endereço do ERP na rede (IP)
erpId	Código de identificação do ERP do fornecedor.
erpType	Tipo do código ERP (DUNS o outro tipo).
unloadignPoints	Pontos de descarregamento de produtos.

A classe *Company* (Empresa) pode assumir tanto o papel de fornecedora quanto de cliente, dependendo do contexto que assumir na classe Item de Fornecimento, e contém todos os dados cadastrais para identificar uma empresa. A fim de permitir agrupamento lógico, por exemplo, empresas podem filiar-se a grupos ou conglomerados.

Uma empresa é relacionada com produtos, que são fornecidos ou adquiridos por ela. Características específicas da empresa, tais como pontos de descarga de produtos (*unloading point*), foram separados em classes próprias a fim de conferir maior modularidade ao sistema.

C.2 Contact

Tabela C.2: Classe Contact.

Atributo	Descrição
contactName	Nome do contato da empresa.
phoneNumber	Número de Telefone
email	Email para contato.

Cada empresa deve possuir contatos para que os gestores do sistema SCIP possam contatá-los em caso de necessidades de configuração do sistema. Logo, a classe *Contact* vem de encontro a esta necessidade, contendo as informações necessárias para definir um contato de uma empresa, podendo esta possuir um ou mais contatos.

C.3 Address

TabelaC.3: Classe Address.

Atributo	Descrição
street	Rua
country	Pais
state	Estado
city	Cidade
zipCode	Código postal.

A classe *Address* define o endereço de uma empresa, que pode assumir um ou mais endereços para correspondência. Por este motivo, o endereço foi abstraído da classe empresa em uma classe própria afim dede permitir mais modularidade e flexibilidade quanto ao numero de endereços.

C.4 Product

Tabela C.4: Classe Product.

Atributo	Descrição
name	Nome do produto.
productCode	Identificador do produto, esta informação é utilizada para identificar o produto no sistema ERP da empresa.
description	Descrição do produto.
productType	Categoria do produto. Permite atribuir um produto a uma categoria de produtos.

A classe *product* representa um item fornecido ou adquirido por uma empresa. O produto tem uma relação direta com a empresa (fornecedor/cliente), sendo que cada empresa pode cadastrá-lo de modo diferente em seu sistema ERP, designando

códigos, nomes e descrições diferentes para o mesmo produto. Por este motivo, decidiu-se que cada produto numa relação cliente-fornecedor teria um cadastro duplo - um relacionado ao cliente e outro ao fornecedor, sendo estabelecida a relação de unicidade entre estes dois cadastro através do Item de Fornecimento.

C.5 UnloadingPoint

Tabela C.5: Classe UnloadingPoint.

Atributo	Descrição
name	Nome do ponto de descarga.
Description	Descrição do ponto de descarga.
inUse	Especifica se está em uso ou não.

A classe *Unloading Point* representa um ponto de descarga de produtos de uma empresa. Como uma empresa pode possuir um ou mais pontos de descarga, e como uma forma de possibilitar maior flexibilidade, criou-se esta classe. Um ponto de descarga pode estar relacionado unicamente a uma empresa, mas pode estar relacionado a diversos Itens de Fornecimento. Ou seja, diversos produtos podem ser entregues para o mesmo ponto.

C.6 SupplyingItem

Tabela C.6: Classe SupplyingItem.

Atributo	Descrição
name	Nome ou descrição do Item de Fornecimento.
seller	Relação com a empresa fornecedora.
buyer	Relação com a empresa cliente.
sellerProduct	Relação com o produto cadastrado junto à empresa fornecedora.
buyerProduct	Relação com o produto cadastrado junto a empresa cliente.
billTo	Relação com a empresa que recebe a fatura
payer	Relação com a empresa que paga a fatura.
shipFrom	Relação com a empresa da qual o produto será enviado. Na maioria dos casos é a mesma empresa relacionada em Fornecedor. Pode ser diferente quando identifica um centro de distribuição da empresa fornecedora.
shipTo	Relação com a empresa que recebe o produto. Na maioria dos casos é a mesma empresa relacionada em Cliente. Pode ser diferente quando identifica um centro de recebimento de produtos da empresa cliente.
unloadingPoint	Relaciona o ponto de descarga da empresa (relacionada em shipTo) que recebe o produto.
isDefault	Especifica se item de Fornecimento é o padrão do item de planejamento, ou se é secundário.

O Supplying Item (Item de Fornecimento) contém as principais informações do sistema para configurar uma relação cliente-fornecedor. Um objeto desta classe contém necessariamente uma relação com uma empresa fornecedora, uma empresa cliente e os respectivos produtos cadastrados para cada uma delas em seus sistemas. Isto é, cadastros diferentes para um mesmo produto de cada empresa são relacionados para identificar um produto em comum nesta classe. Ademais, a classe relaciona quem é a empresa pagante, a empresa que deve receber a fatura da entrega, e a empresa de origem e destino da entrega.

C.7 Planning Item

Tabela C.7: Classe PlanningItem.

Atributo	Descrição
id	Identificação do item de planejamento
description	Descrição do item de planejamento.
onHand	Nivel de estoque atual do cliente.
forecast	Vetor de previsão de consumos do cliente.
daysOfForwardCoverage	Dias de cobertura do algoritmo de aprovisionamento.
safetyStock	Estoque mínimo de segurança.
transportTime	Tempo necessário ao transporte do fornecedor ao cliente
productionTime	Tempo necessário para produção de ordens de entrega a partir
orders	Vetor contendo ordens passadas e futuras do item de
shippingCalendar	Calendário de entregas da empresa fornecedor.
receivingCalendar	Calendário de recebimento da empresa cliente.
overflowAllowed	Especifica se o estoque máximo do cliente pode ser ultrapassado
minShipQuantity	Quantidade mínima para realizar uma entrega do produto
incShipQuantity	Incremento mínimo na quantidade do produto para uma ordem de
incrementMultiple	Especifica quantos incrementos podem ser dados em uma ordem
incrementMultipleEnabled	Especifica se o incrementMultiple deve ser usado para geração
splitOrdersEnabled	Especifica se cada incremento deve ser criado em uma ordem
projectedOnHand	Estoque projetado do cliente a partir da previsão de consumo.
supplyingItems	Itens de fornecimento deste item de planejamento.
unitOfMeasure	Unidade de media a ser usada nas ordens.

O *Planning Item* representa a entidade central do sistema, o item de planejamento. Este relaciona todas as informações necessárias para a criação de ordens e administração do estoque do cliente. É dele que são inferidas as características e regras para a criação de ordens de reposição de estoque.

Inicialmente, concebeu-se o Item de Planejamento contendo todas as informações explicitadas no Item de Fornecimento. Num segundo momento, notou-se que alguns casos excepcionais requerem uma flexibilidade maior do item de planejamento, de forma que se criou o Item de Fornecimento como uma classe a parte. Desta forma um item de planejamento pode estar relacionado a um ou mais itens de

fornecimento, mas representando o mesmo contrato de fornecimento entre cliente e fornecedor para um produto.

Esta configuração permite a existência de um caso onde uma empresa fornecedora possui duas localidades diferentes das quais envia seus produtos para suprir o estoque do cliente, de forma que a cada entrega uma localidade diferente possa ser selecionada.

Adicionalmente, a classe relaciona o calendário de entrega do cliente e o calendário de fornecimento do fornecedor através da classe *LogisticCalendar*, e todas as variáveis para o planejamento de ordens, tais como tempos para entrega e produção, níveis mínimos e máximos de estoque, regras de negócio e restrições específicas.

C.8 Order

Tabela C.8: Classe Order.

Atributo	Descrição
purchaseNumber	Numero de identificação da ordem no sistema ERP do
proposedDate	Data de entrega da ordem
Quantity	Quantidade do produto a ser entregue.
supplyingItem	Item de fornecimento relacionado a esta ordem.
status	Estado atual da ordem.
minimus	Quantidade de mínimos.
increases	Quantidade de incrementos.

A classe *Order* representa uma ordem de entrega de um produto que o sistema gera para enviar ao sistema ERP do fornecedor. Esta classe está intimamente ligada ao Item de Planejamento, a partir do qual os parâmetros para criação de ordens são buscados. Ao enviar a ordem para o sistema ERP do fornecedor, as informações variáveis da classe ordem (data, quantidade, etc.) são combinadas com os parâmetros imutáveis da classe item de planejamento (unidade de medida, cliente, etc.).

C.9 Quantity

Tabela C.9: Classe Quantity.

Atributo	Descrição
value	Valor que expressa quantidade.
dateTime	Data de referencia da quantidade.

A classe *Quantity* representa uma estrutura de dados para relacionar um valor de algum produto a uma data. Este é utilizado para construir as classes *Forecast* e *OnHand*.

C.10 Forecast

Tabela C.10: Classe Forecast.

Atributo	Descrição
quantity	Quantidade do tipo Quantity
receivedDate	Data de recebimento da previsão de consumo

A classe *Forecast* reapresenta uma previsão de consumo de produto do cliente para uma dada data, contido em um item de planejamento. Para tal, ela utiliza a classe *Quantity* para especificar o valor do consumo previsto para qual data. Ademais, ela contém um parâmetro para indicar a data na qual a previsão de consumo foi recebida. Esta classe contém as informações recebidas do ERP do cliente na mensagem de envio de previsão de consumo.

C.11 OnHand

Tabela C.11: Classe OnHand.

Atributo	Descrição
quantity	Quantidade do tipo Quantity
receivedDate	Data de recebimento do nível de estoque.

A classe *OnHand* representa o nível de estoque do cliente para certo produto ou item de planejamento para uma data especifica. Esta classe conte os dados recebidos do sistema ERP do cliente na mensagem de envio de nível de estoque.

C.12 LogisticCalendar

Tabela C.12: Classe LogisticsCalendar.

Atributo	Descrição
days	Dias habilitados para recebimento/entrega

A classe *LogisticsCalendar* contém a relação de dias habilitados para entrega e recebimento de produtos para uma empresa cliente ou fornecedora dentro de um item de planejamento. O algoritmo de aprovisionamento utiliza as informações contidas nesta classe juntamente com regras de negocio do item de planejamento para definir as data ideais dos planejamentos de entrega.

C.13 ReplenishLogic

Esta enumeração explicita os tipos de aprovisionamento possíveis no sistema SCIP. As possibilidades para tipos de aprovisionamento estão explicitadas na seção 3.6.2 (Tipos de Aprovisionamento). A enumeração torna a aplicação mais flexível quanto à definição de novos tipos de aprovisionamento de estoque.

C.14 UnitOfMeasure

A enumeração contém todas as unidades de medida contemplada para a criação de ordens de entrega respeitando convenções especificas de clientes e fornecedores.

C.15 OrderStatus

A enumeração *OrderStatus* contém os estado que uma ordem assume durante o seu ciclo de vida, tal como definido no seção 3.6.3 (Algoritmo de gestão de ordens).

APÊNDICE D – DIAGRAMA DE ENTIDADE RELACIONAMENTO

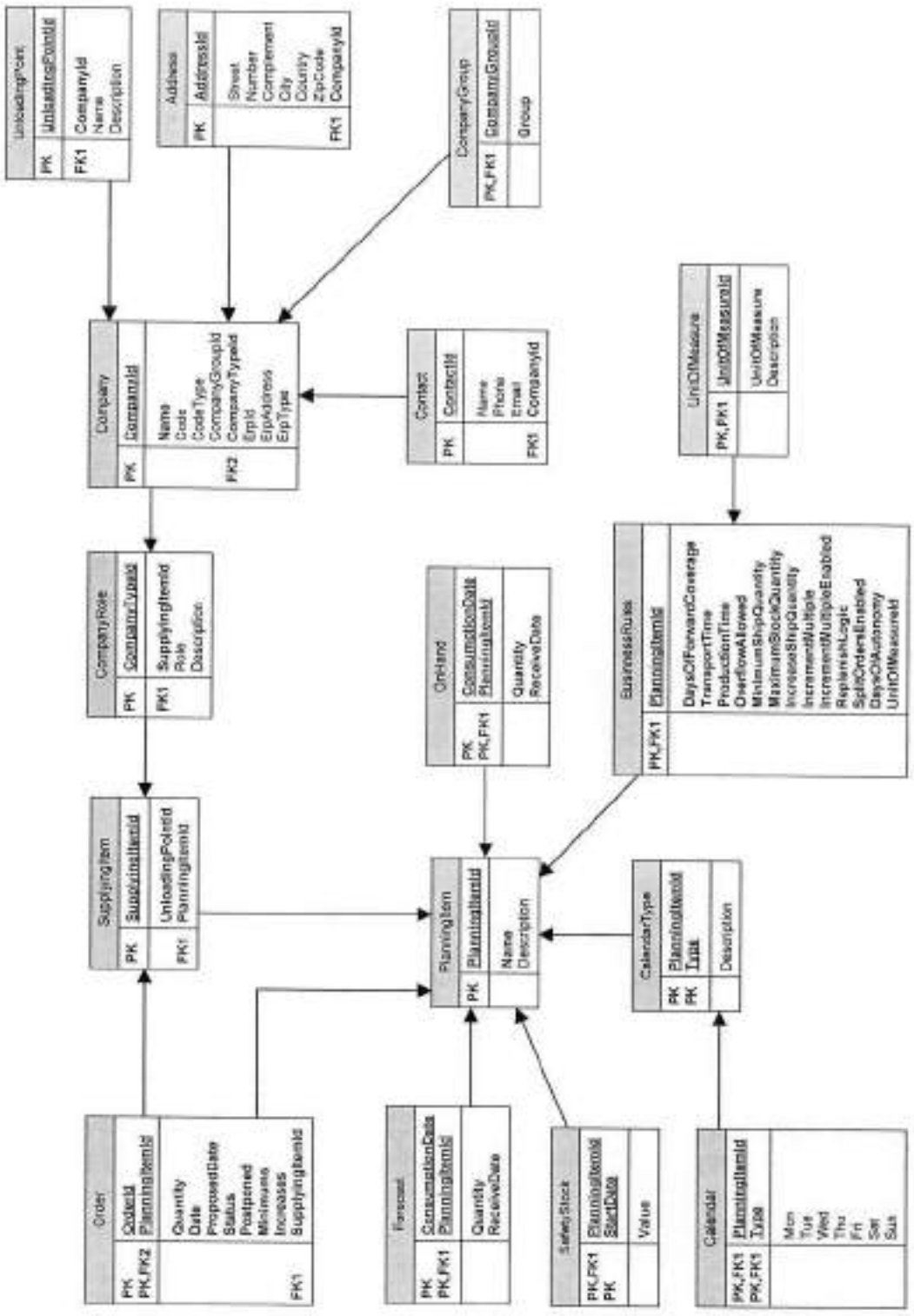


Figura D.1: Diagrama de Entidade-Relacionamento.

APÊNDICE E – DIAGRAMAS DE SEQÜÊNCIA

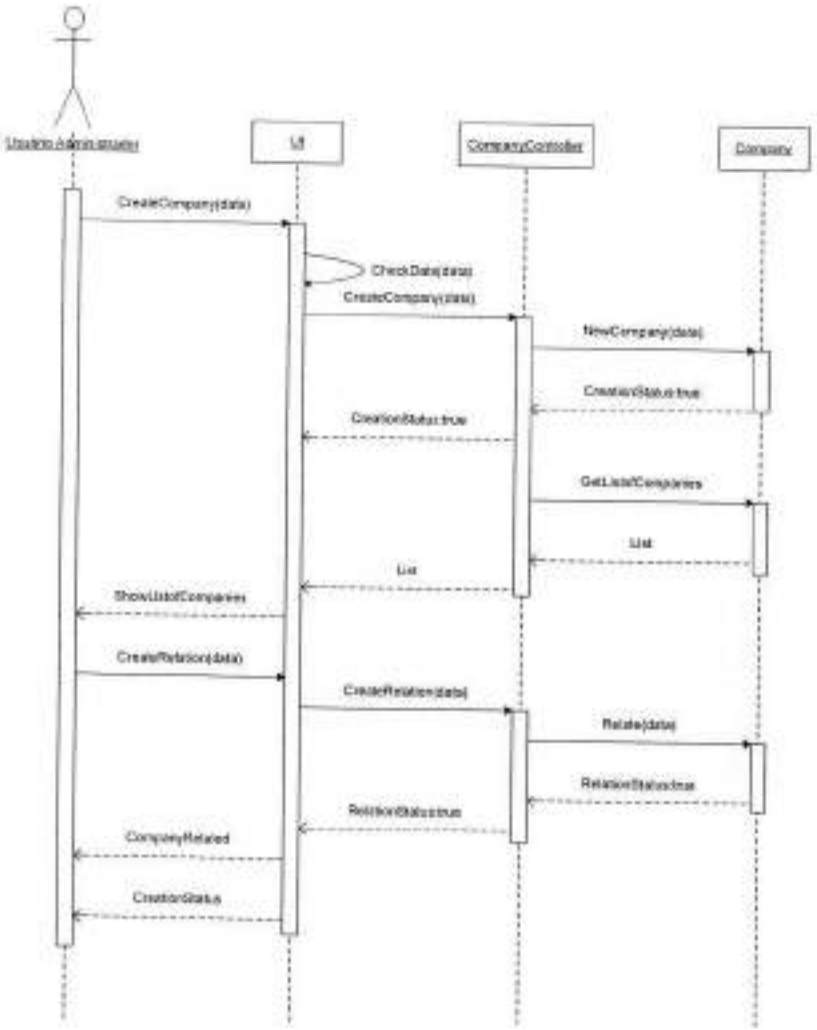


Figura E.1: Caso de Uso 1.

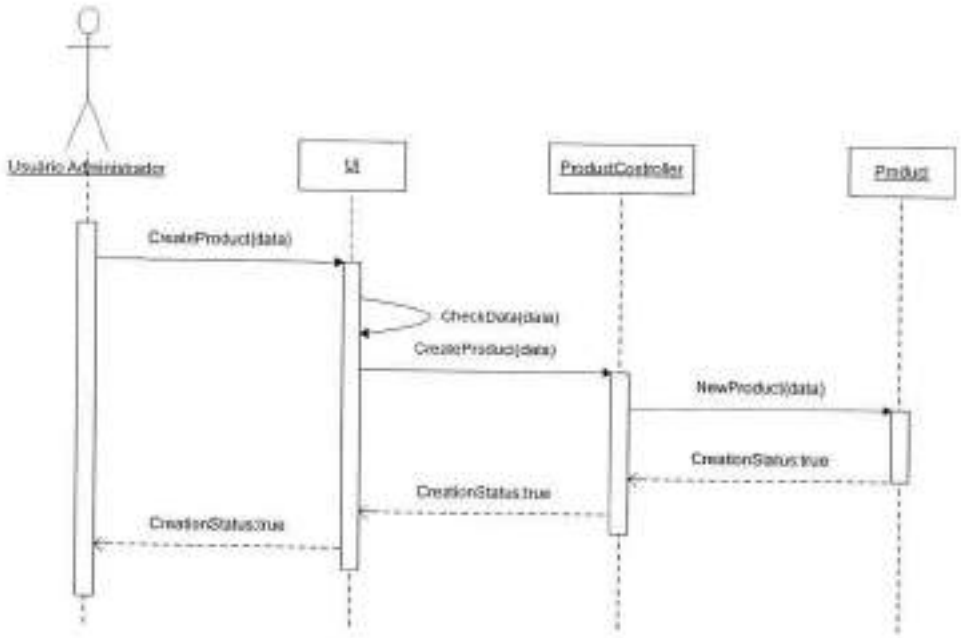


Figura E.2: Caso de Uso 2.

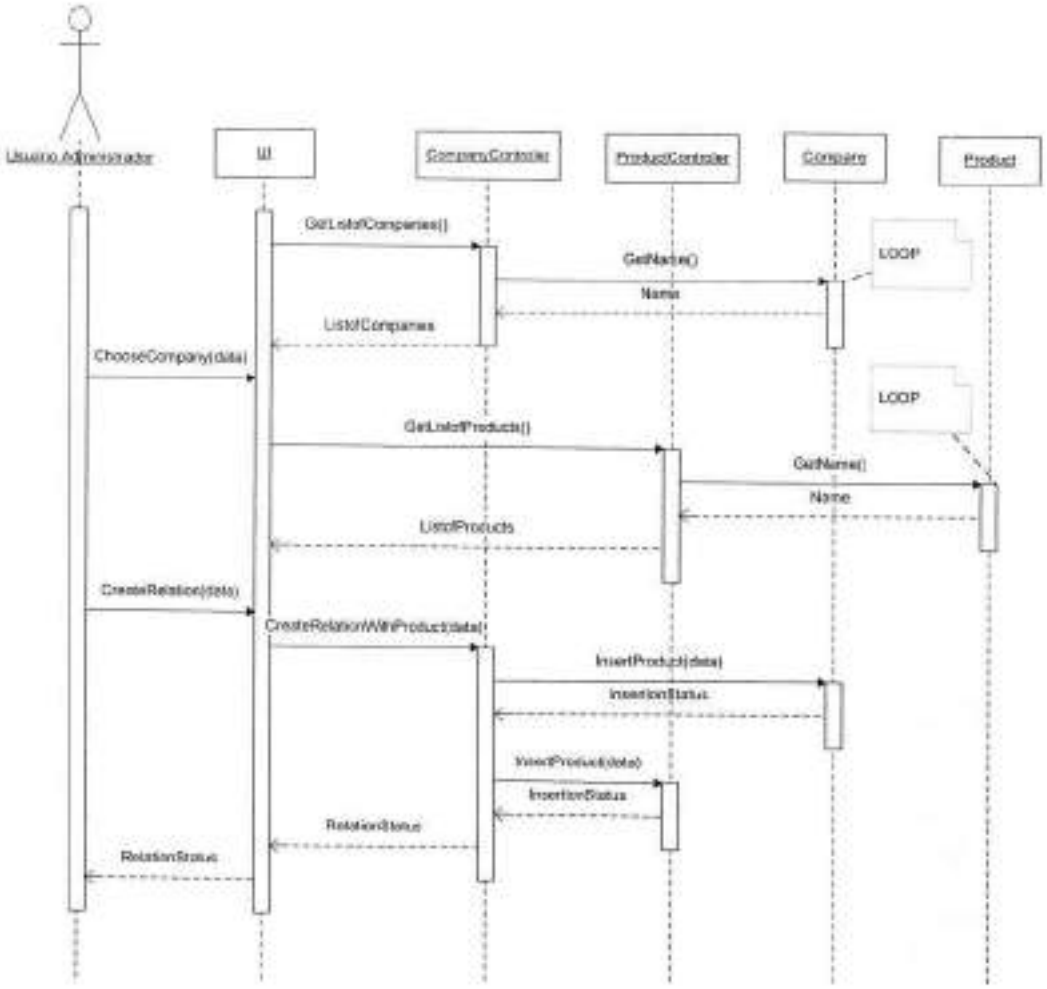


Figura E.3: Caso de Uso 3.

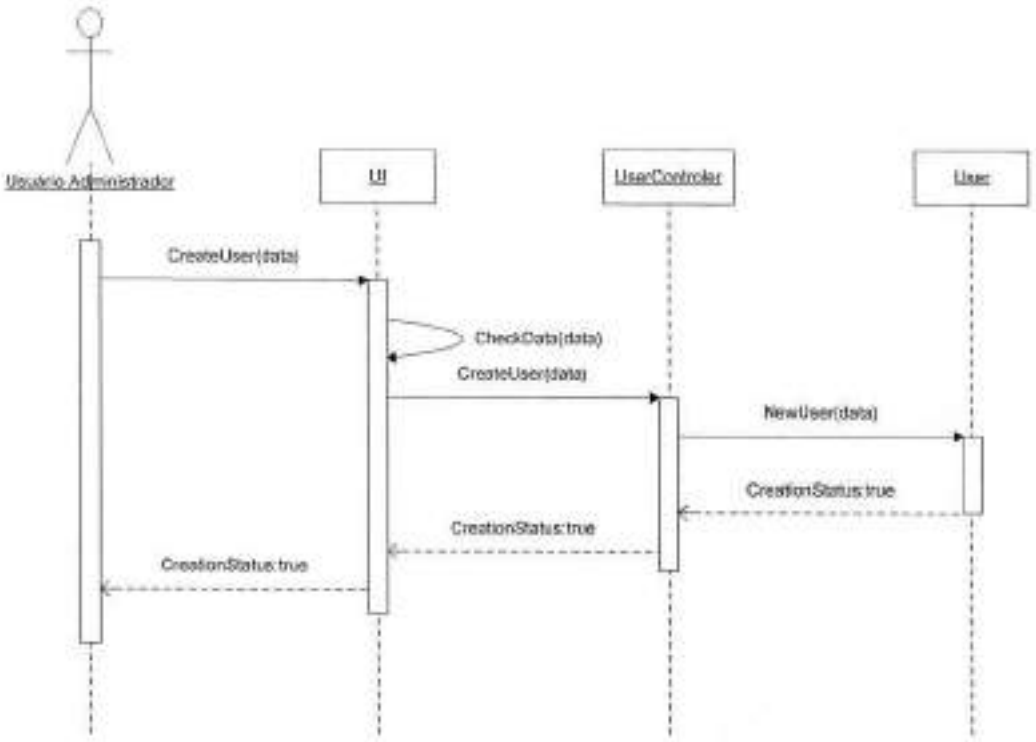


Figura E.4: Caso de uso 4.

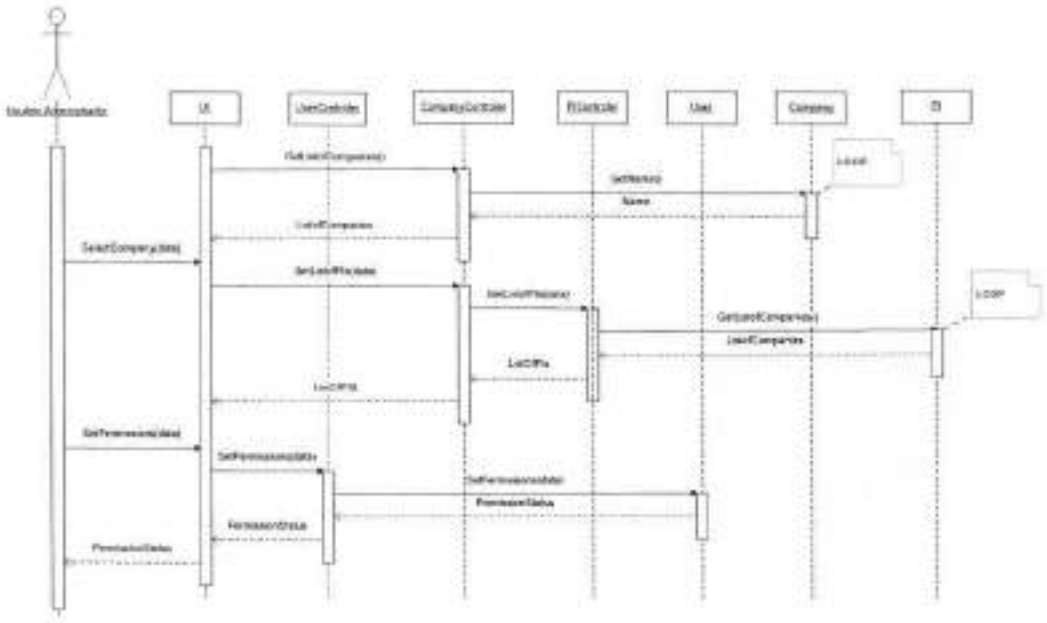


Figura E.5: Caso de Uso 5.

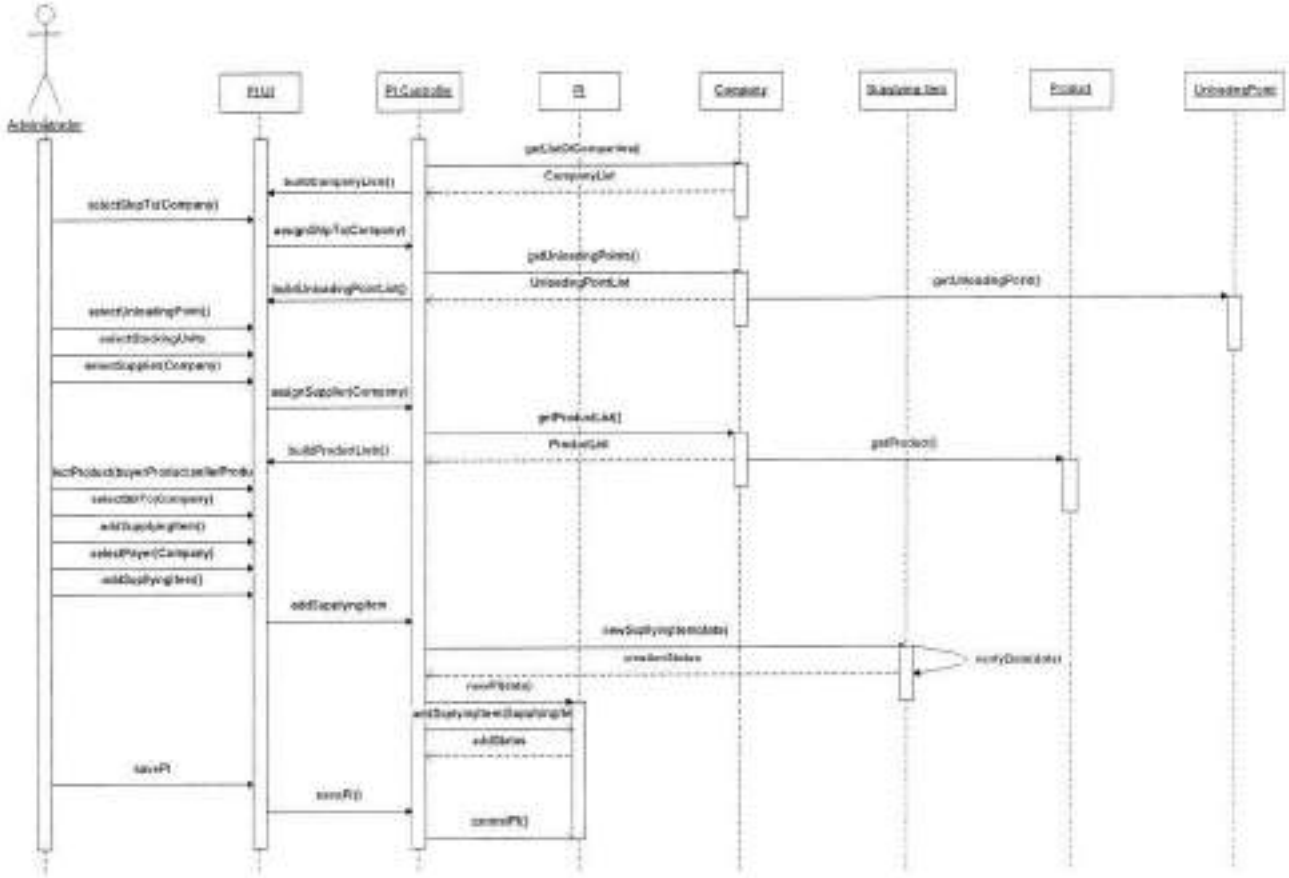
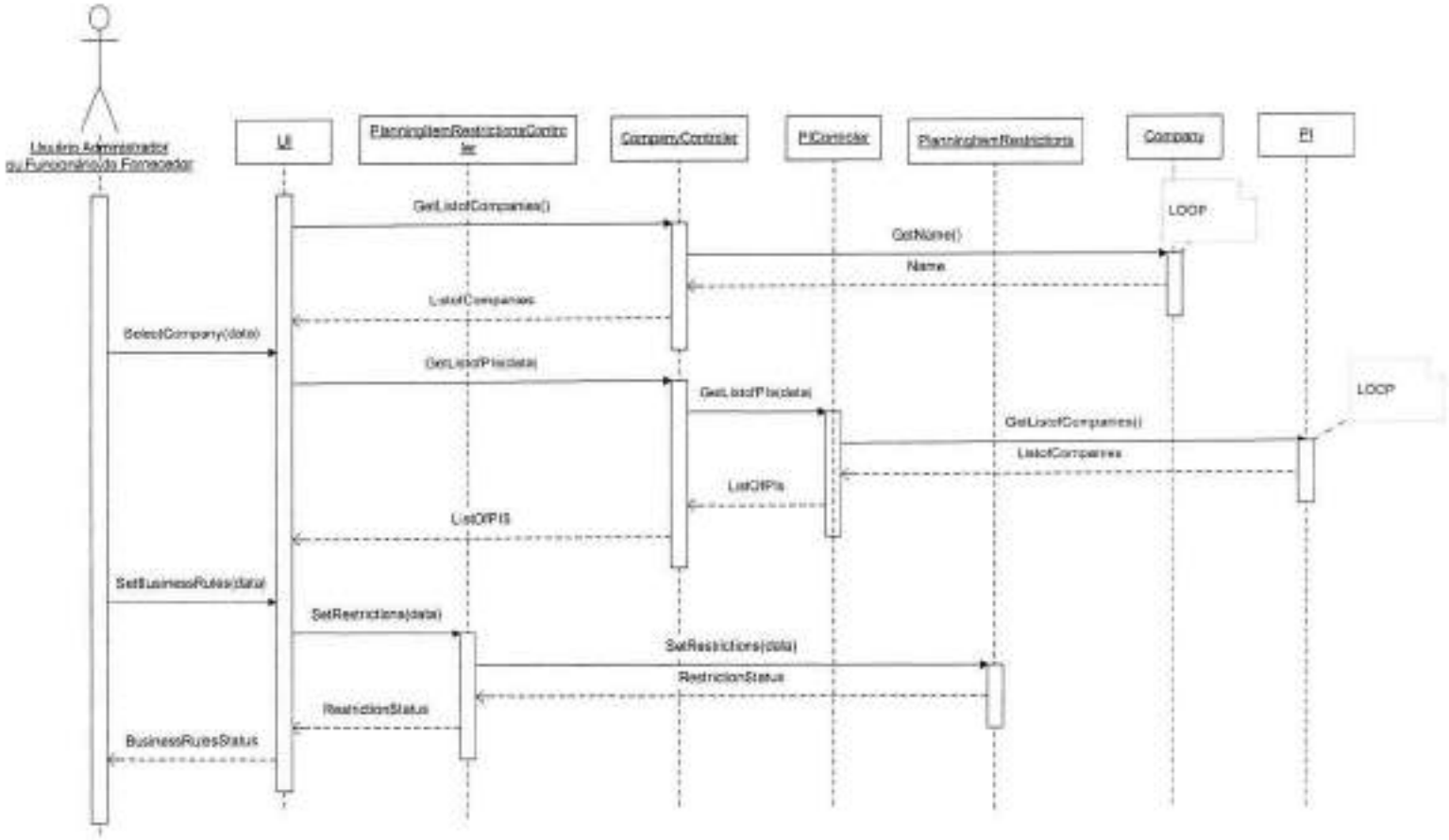


Figura E.6: Caso de Uso 8.



FiguraE.7:Caso de Uso 9.

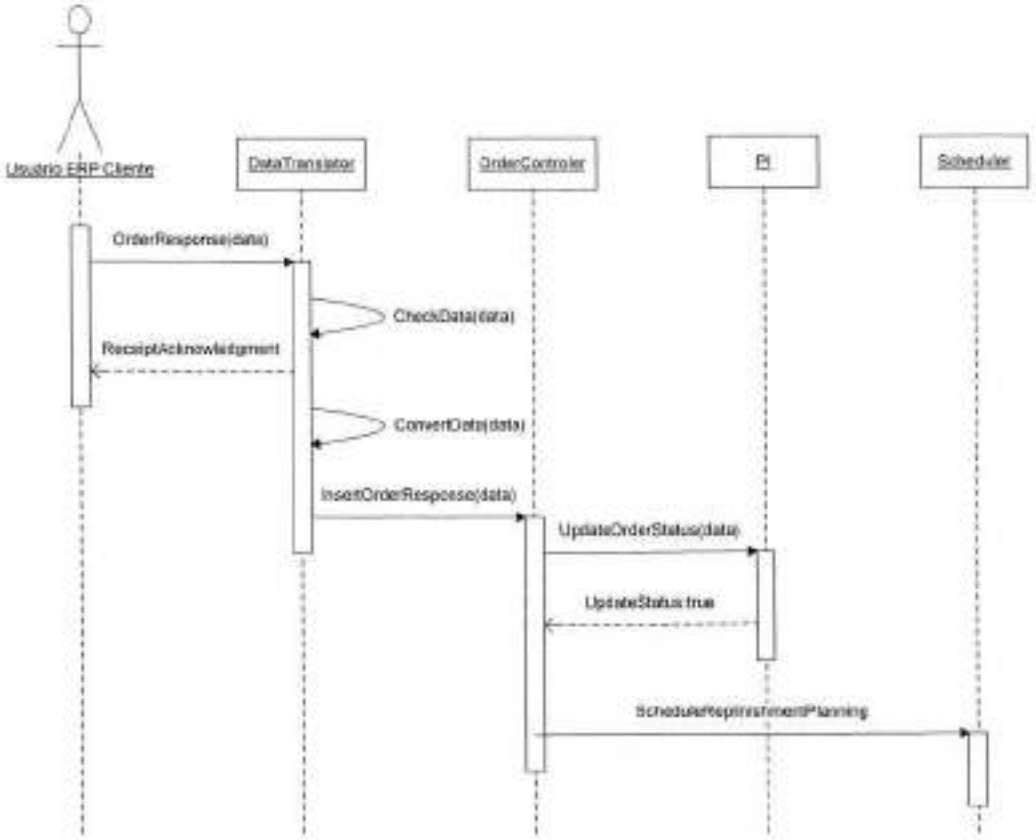


Figura E.8: Caso de Uso 14.

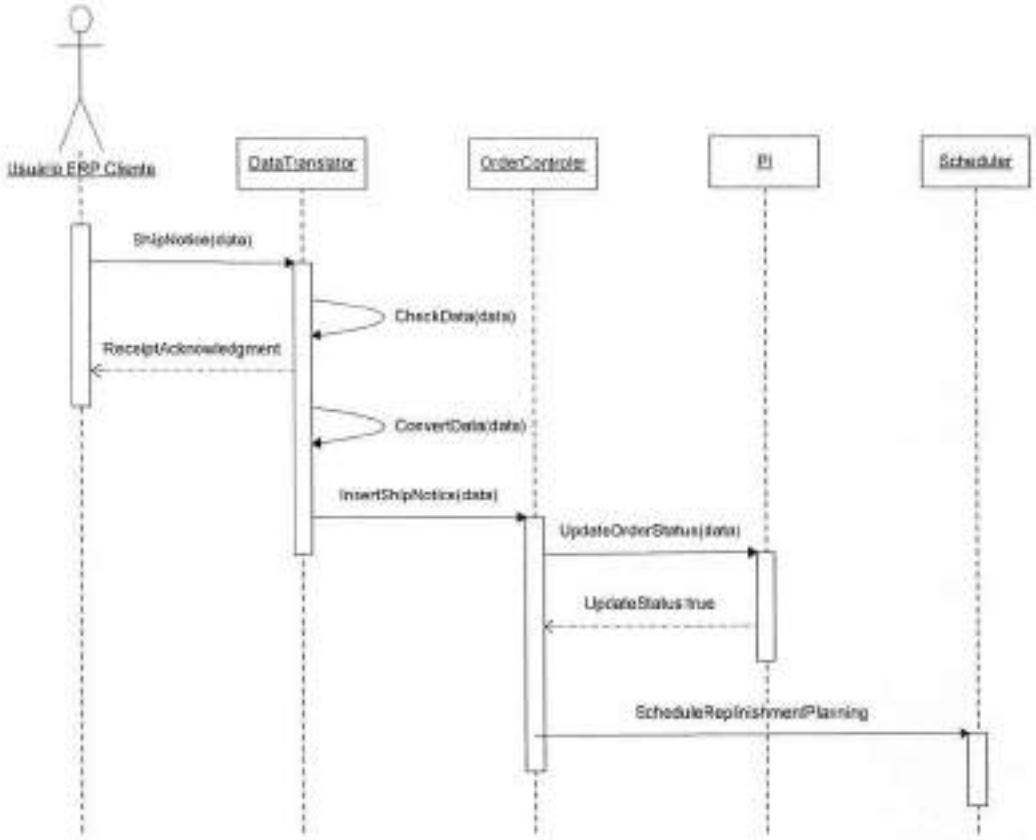


Figura E.9: Caso de Uso 15.

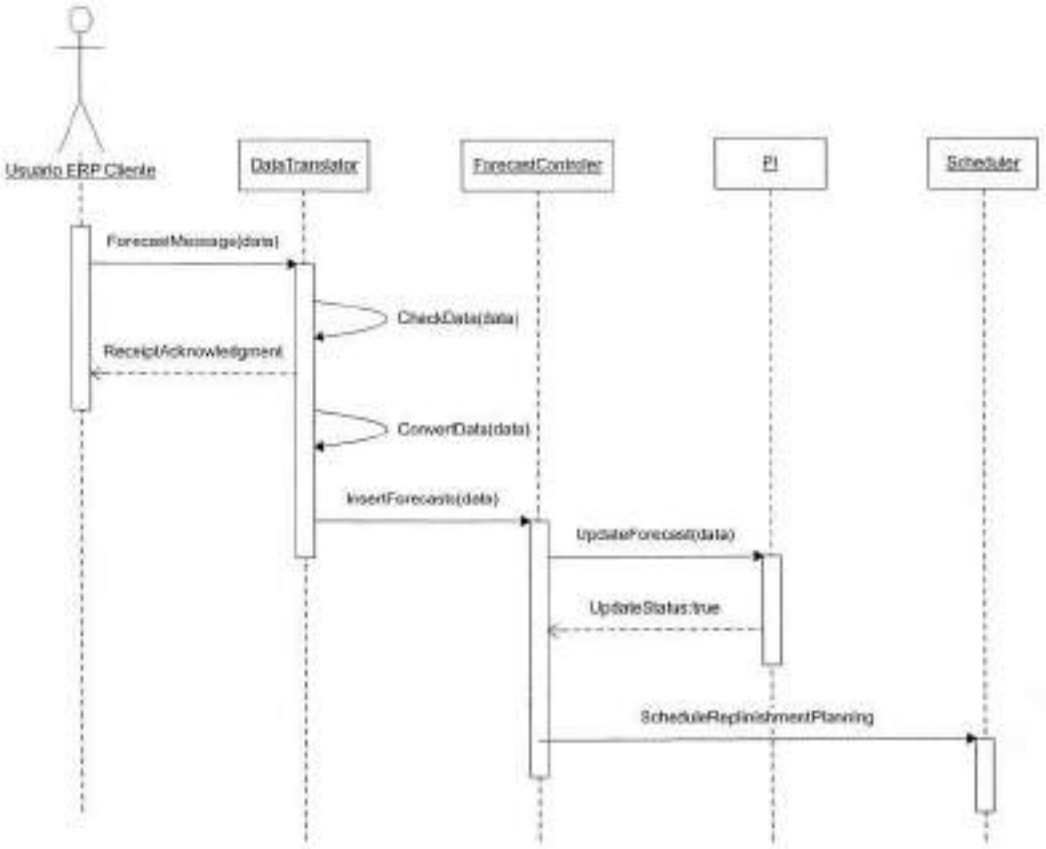


Figura E.10: Caso de Uso 19.

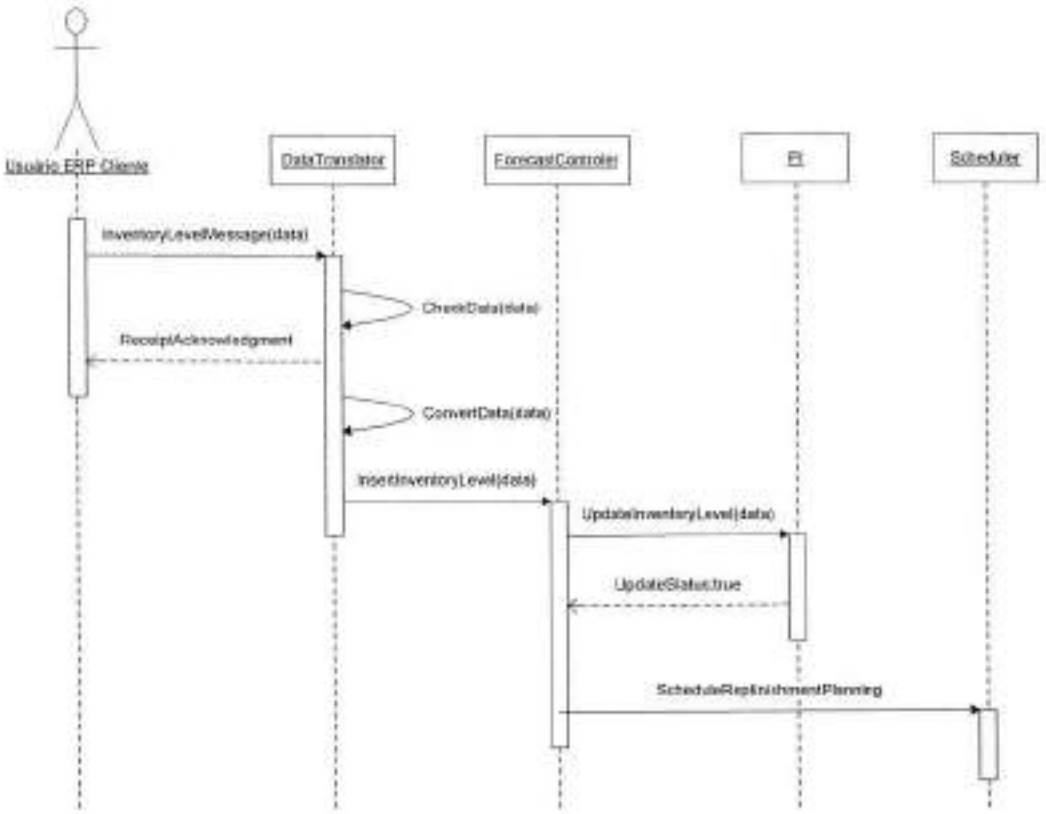


Figura E.11: Caso de Uso 21.

APÊNDICE F – VISÃO DOS COMPONENTES DO SISTEMA

Este apêndice apresenta os diagramas de classe simplificados dos componentes de software centrais do sistema SCIP.

F.1 Visão Geral

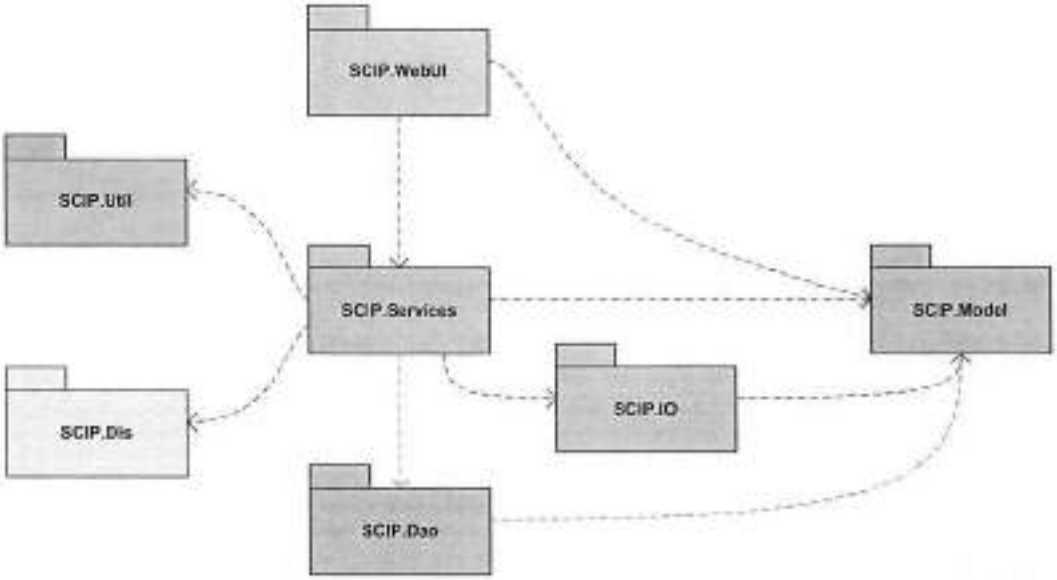


Figura F.1: Arquitetura de software do sistema.

F.2 SCIP.WebUI

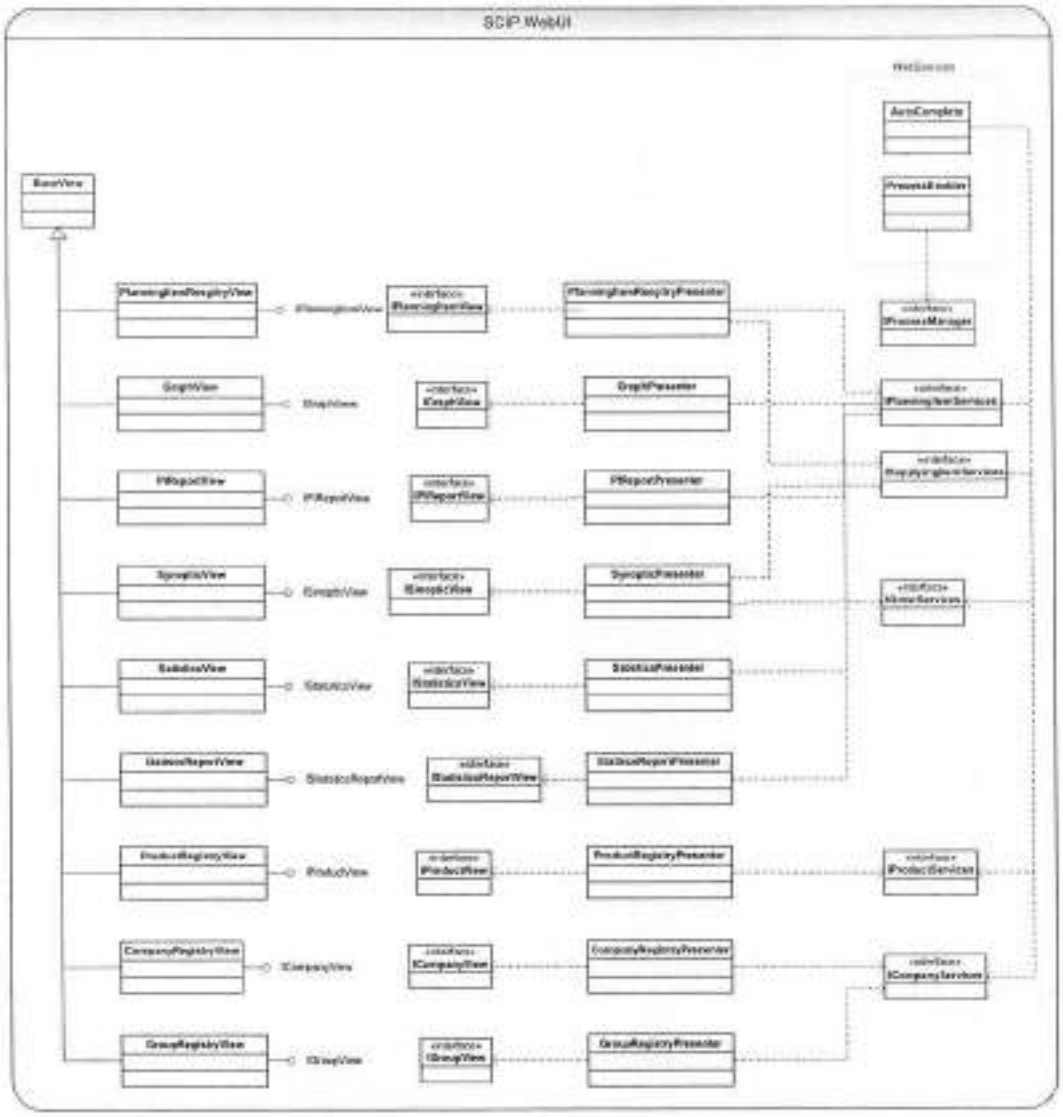


Figura F.2: Componente SCIP.WebUI.

As classes Registry (MVP) provêm a apresentação do conteúdo às classe de visualização, definido as interfaces que devem ser providas pela camada de serviço pra apresentação dos dados. OS MVPs também definem as interfaces que as classe de visualização devem implementar, obtendo assim uma abstração da implementação da visualização, facilmente intercambiável. Finalmente, as classes View implementam a visualização dos dados apresentados, estendendo funcionalidades comuns especificadas na abstração BaseView.

BaseView fornece os métodos genéricos para a população de listas, tratamento correto de entradas e saídas e a apresentação de tabelas (grids).

F.3 SCIP.Services

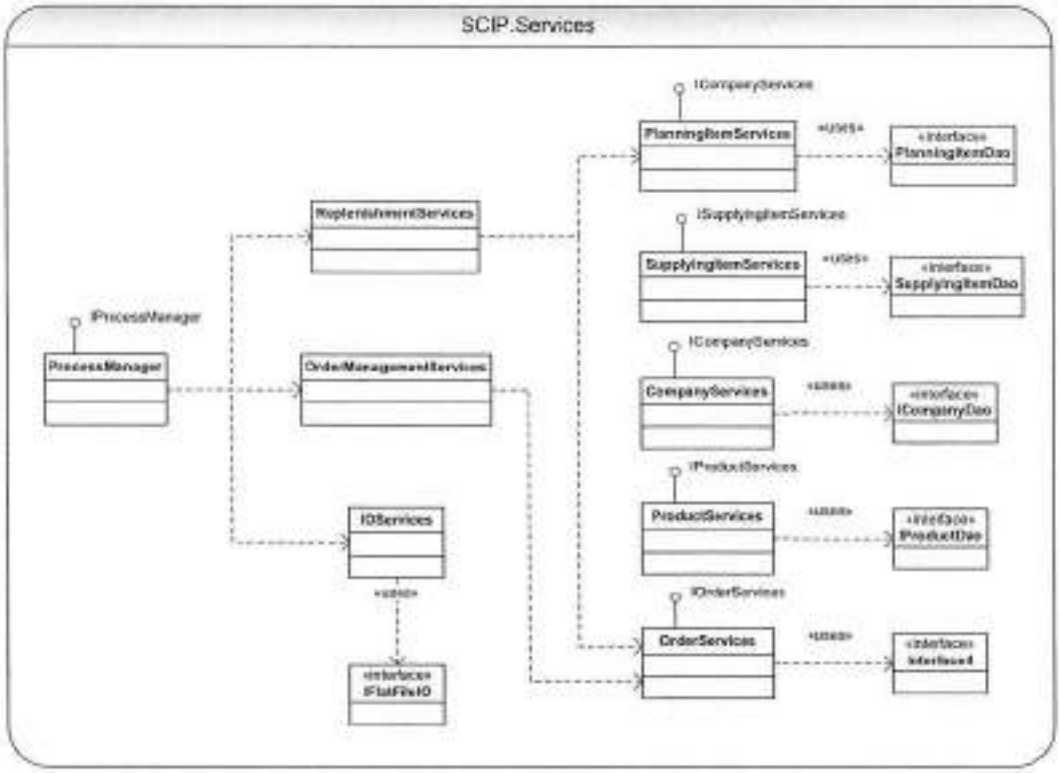


Figura F.3: Componente SCIP.Services.

O acesso e manipulação dos componentes de negócio é feito através das classes OrderServices, ProductServices, PlanningItemServices,SupplyingItemServices e CompanyServices. Estes definem na interface os métodos que a camada Dão deve prover para acesso aos componentes.

A classe ReplenishmentService implementa o algoritmo de aprovisionamento que realiza ao planejamento de entregas de produto. A classe OrderManagement Service orquestra a transição de estados das ordens do sistema, recebendo e tratando de forma adequada o conteúdo das mensagens XML.

A classe IOServices prove tratamento pra acesso e criação de arquivos de entrada e saída, definido uma interface para que a camada IO implemente a leitura e escrita.

Note-se que IOServices torna-se independente do tipo de arquivo que se está acessando na camada IO através da abstração possibilitada pela interface.

Finalmente, a classe Processmanager orquestra a execução dos outros serviços, coordenando a ordem em que são executados e o fluxo de informação entre os serviços. Esta classe também implementa os métodos definidos pela camada de interface usuário para prover conteúdo.

F.4 SCIP.Dao

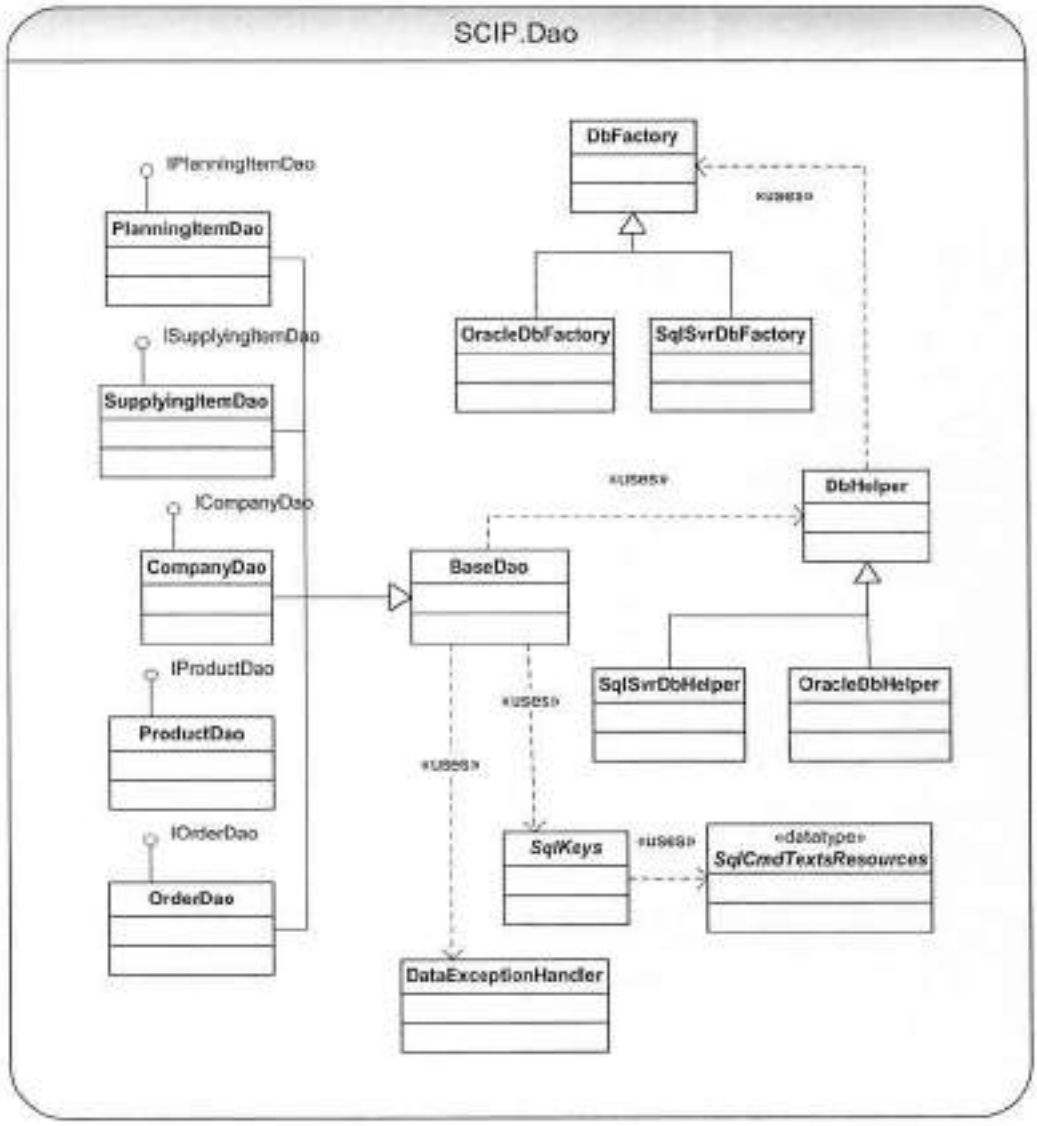


Figura F.4: Componente SCIP.Dao.

O acesso a dados é feito aplicando-se *Factory*[1] e herança para criar uma camada que abstraia a lógica de conexão e transação com o SGBD. Dessa forma, DbFactory – a classe abstrata – é estendida por classes como SqlSvrDbFactory e OracleDbFactory, as quais especializam funcionalidades da classe base para realizar a lógica de interface com SGBDs diferentes. Esta estrutura permite que este componente seja flexível quanto à escolha do SGBD.

A classe DbHelper é uma classe abstrata utilizada para definir métodos genéricos de acesso a camada de dados - tais como inserção, atualização e remoção - e a passagem de parâmetros das consultas. Esta classe é estendida para criar uma implementação específica para um tipo de bases de dados, no caso a classe SqlSvrDbHelper.

A classe BaseDao corresponde ao AbstractCreator definido no design pattern. Este define os métodos para implementar as chamadas a base de dados especificadas na BaseDao passando os parâmetros específicos para uma base de dados SqlSvr especificados na classe SqlKeys. A sqlKeys acessa os comandos e procedimentos específicos para uma base SqlServer através do arquivo SqlCmdTextResources. Este contém a especificação dos comandos para cada operação de inserção, atualização e remoção da base. A classe BaseDao utiliza então o dbHelper e SqlKeys para realizar as consultas a base de dados e implementa métodos para geração de objetos genéricos.

As classes companyDao, productDao, OrderDao, SupplyingItemDao e PlanningItemDao estendem a classe abstrata BaseDao para implementar os ConcreteCreators. Estes sobrecarregam e estendem métodos genéricos da classe BaseDao para a partir de consultas obter objetos do tipo específico correspondente as classe definidas na camada Model (Company, Product, Order, SupplyingItem, PlanningItem, respectivamente).

[1] Vide seção 4.4.3.

Observe-se que a definição de um ConcreteCreator torna-se trivial, necessitando apenas estender a classe BaseDao para especificar uma cesso a dados de outra classe. Estes implementam a interface definida pela camada acima (Services) para acesso aos objetos específicos da base.

A classe Dao atinge abstração e modularidade quanto a tipo de base de dados utilizada, seu métodos e chamadas específicos e tipos de objeto que podem ser instanciados.

F.5 SCIP.IO

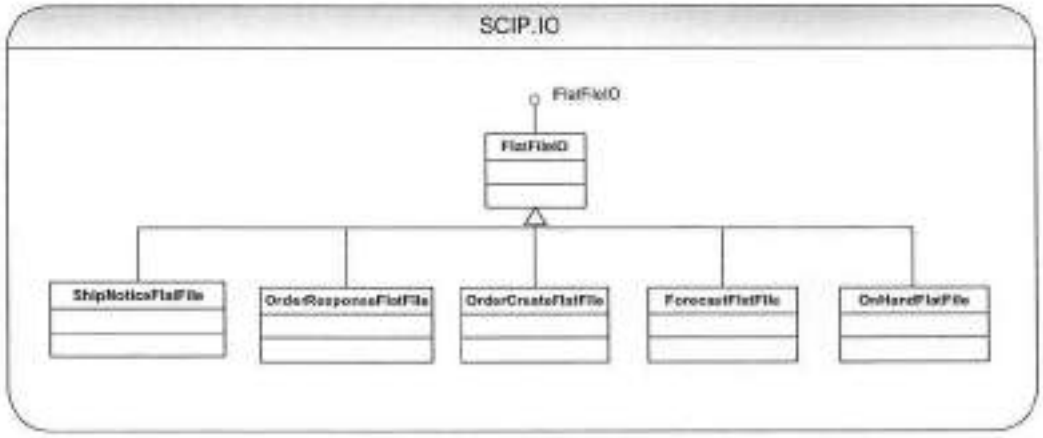


Figura F.5: Componente SCIP.IO.

APÊNDICE G – ALGORITMO DE APROVISIONAMENTO

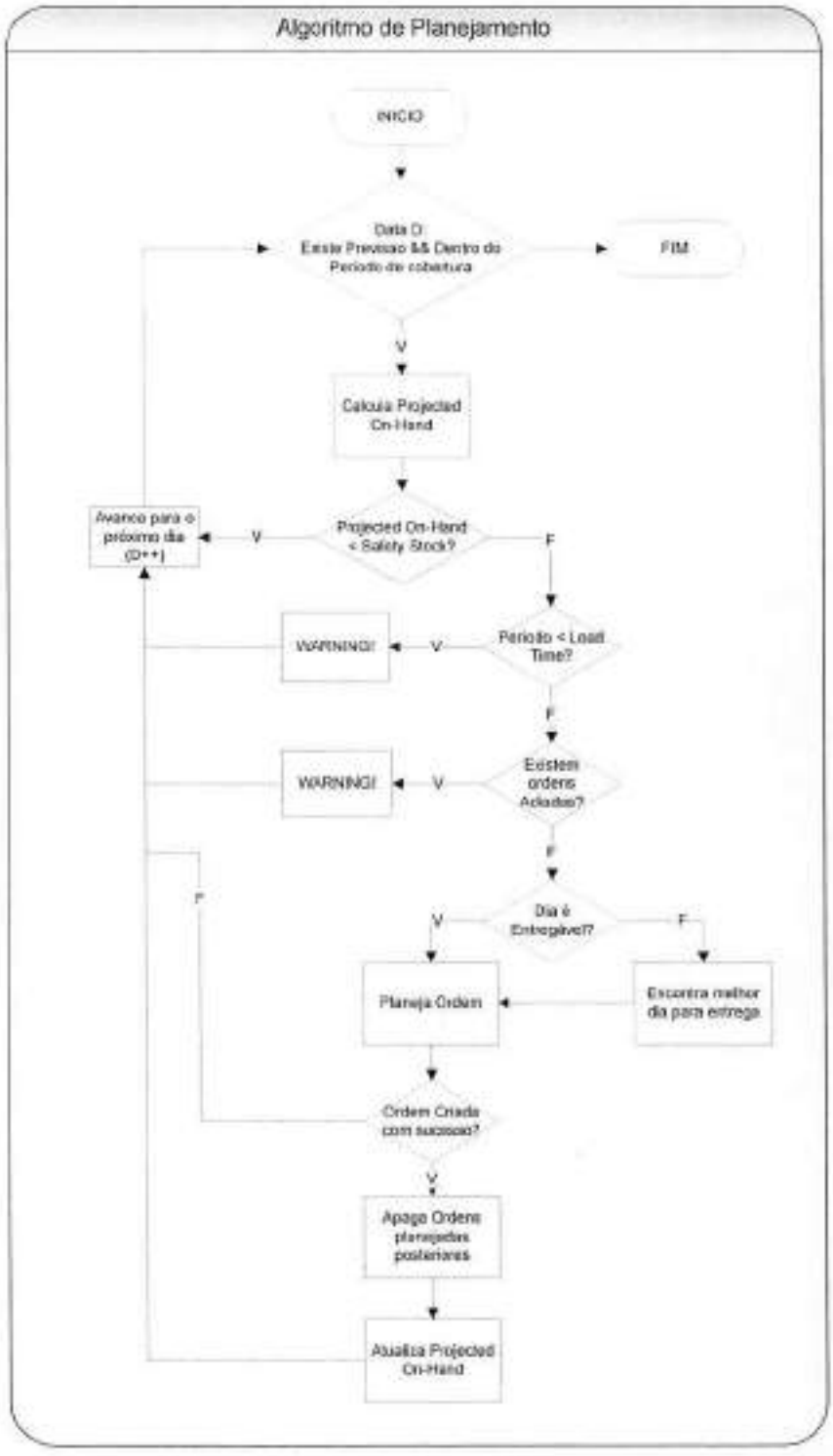


Figura G.1: Algoritmo de aprovisionamento.

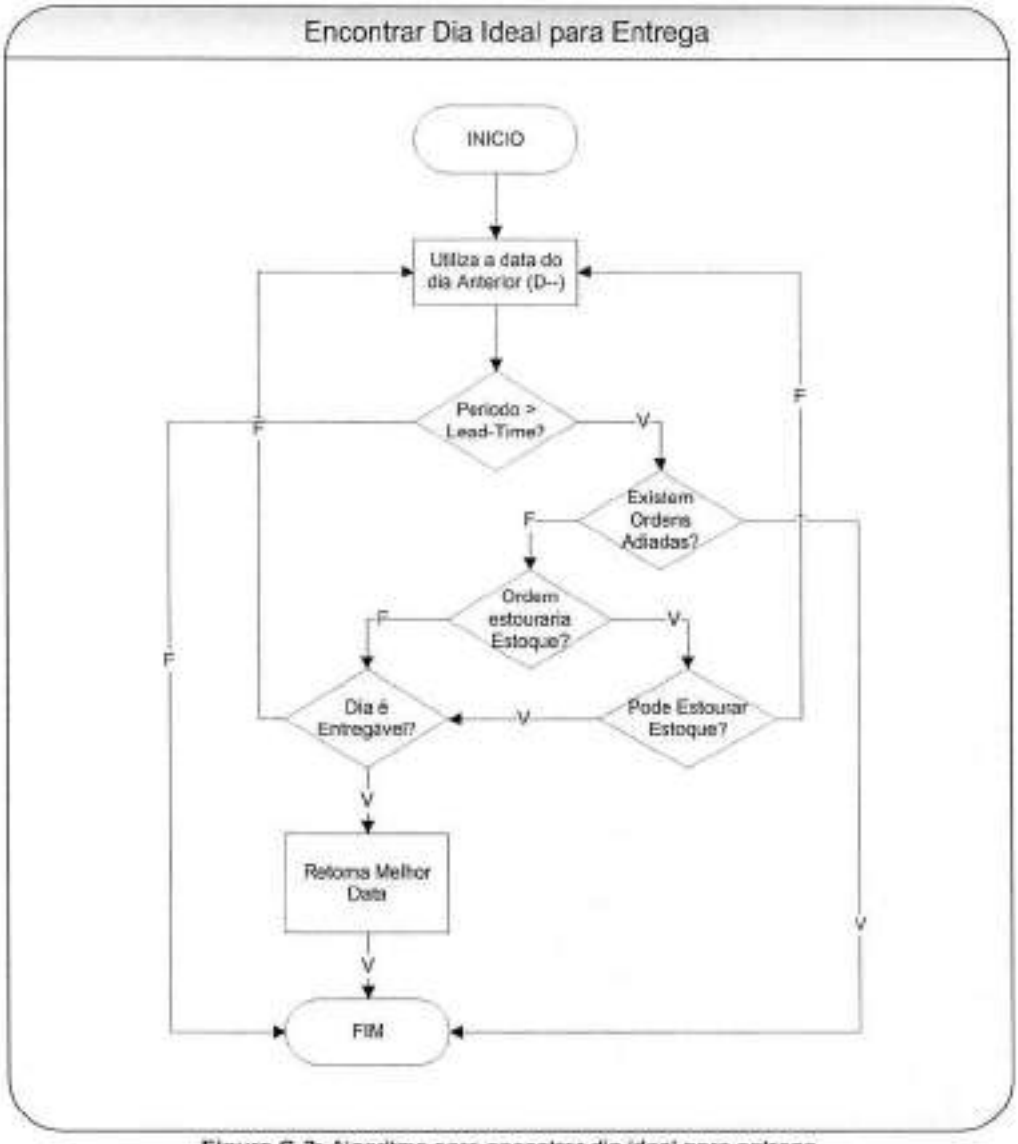


Figura G.2: Algoritmo para encontrar dia ideal para entrega.

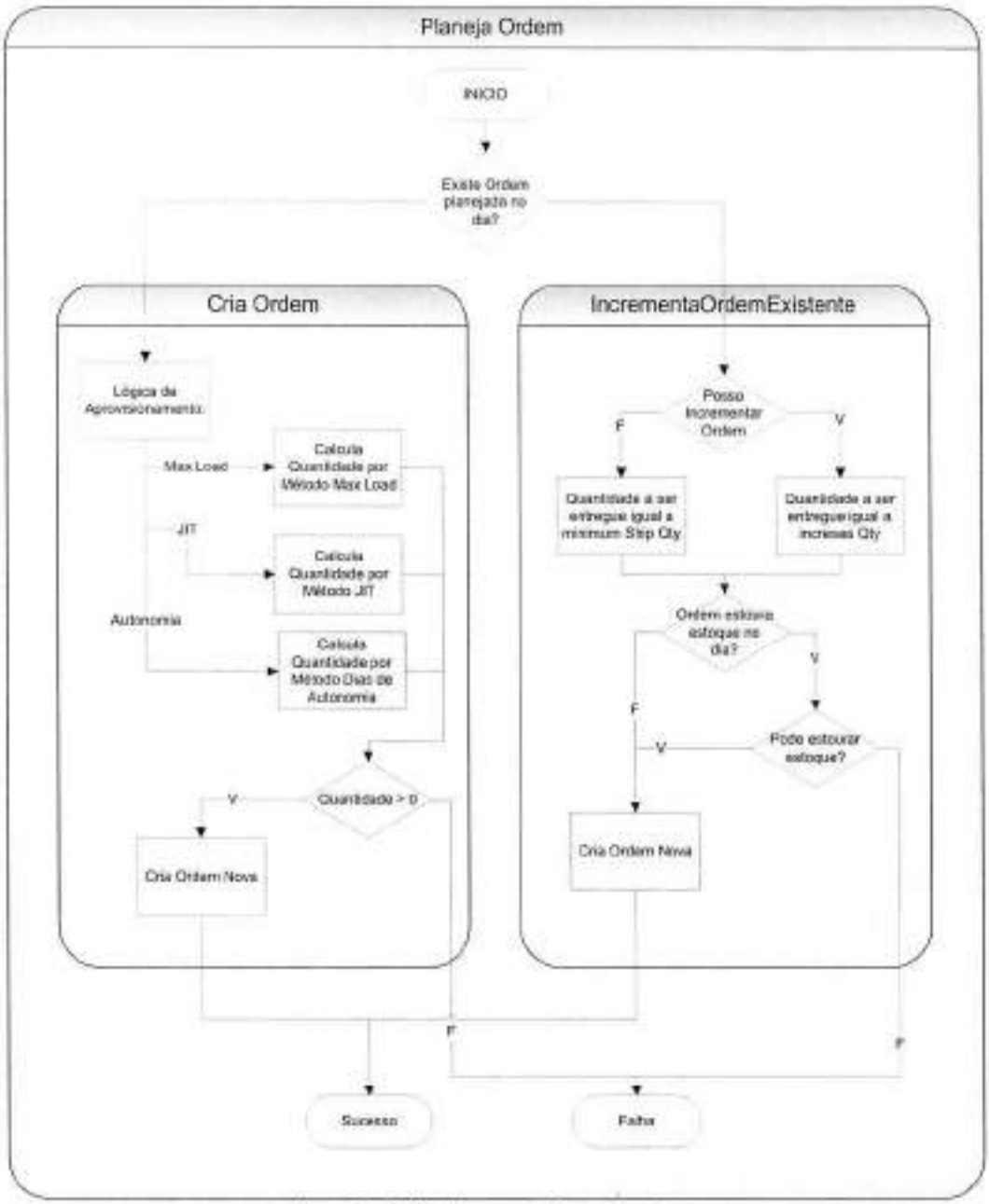


Figura G.3: Algoritmo para planejar ordens.

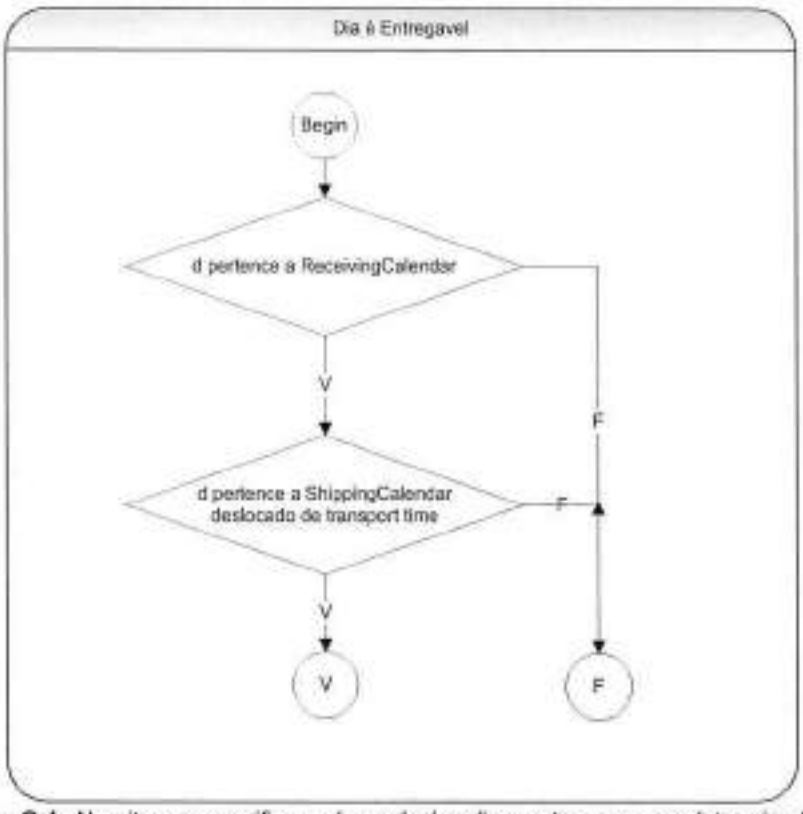


Figura G.4: Algoritmo que verifica se é possível realizar entrega em um determinado dia.

APÊNDICE H – ESPECIFICAÇÃO DE MENSAGENS

O padrão desenvolvido neste projeto utiliza mensagens XML baseadas na norma *Chem eStandards* da CIDX.

Todas as mensagens possuem a seguinte estrutura padrão:

```
<Tipo da mensagem>
      <Header>
      </Header>
      <Body>
            <Properties></Properties>
            <Partners></Partners>
            <Details></Details>
      </Body>
</Tipo da mensagem>
```

H.1 Especificação dos campos:

- *Tipo da mensagem:* especifica o tipo da mensagem, esta pode ser:
  - Order Create;
  - Order Response;
  - Ship Notice;
  - Inventory Actual Usage.

- *Header:* contém as seguintes informações:
  - *Código identificador da mensagem:* Código único que identifica a mensagem eletrônica na rede EDI pela qual a mesma esta trafegando;
  - *Sistema emissor da mensagem:* Informações sobre o sistema emissor da mensagem incluindo seu código de identificação;
  - *Sistema destinatário da mensagem:* Informações sobre o sistema destinatário da mensagem incluindo seu código de identificação.

- *Body:* Corpo da mensagem contêm os seguintes itens:
  - *Properties:* Contêm informações fiscais sobre o documento, informações sobre local de origem e destino e data do documento;

- *Partners:* Contêm informações sobre todos os parceiros envolvidos na relação VMI: BUYER, SELLER, SHIPTO, BILLTO, PAYER
- *Details:* Contêm o conteúdo da mensagem.

H.2 Conteúdo dos campos:

- Mensagem Order Create:
  Contém um ou mais pedidos de compra, cada pedido contém as seguintes informações:
  - Produto desejado;
  - Data de entrega;
  - Quantidade desejada.

- Mensagem Order Response:
  Contém um ou mais confirmações de pedidos de compra, cada confirmação contém as seguintes informações:
  - Produto desejado;
  - Data de entrega;
  - Quantidade desejada;
  - Status: Confirmado, Pendente ou Apagado.

- Mensagem Ship Notice:
  Contém um ou mais avisos de embarque, cada aviso de embarque contém as seguintes informações:
  - Produto;
  - Quantidade enviada;
  - Data de saída de mercadoria;
  - Data prevista de recebimento.

- Inventory Actual Usage
  Indica o nível de estoque no cliente para um determinado produto:
  - Produto;
  - Estoque atual;
  - Hora da medição.

- Demand Forecast

  Contém uma ou mais linhas de previsões de estoque. Todas as linhas referenciam previsões de consumo de um determinado produto indicado no início da mensagem, Cada linha de previsão contém as seguintes informações:

  - Data futura;
  - Previsão de consumo da data.

APÊNDICE I – INTERFACES

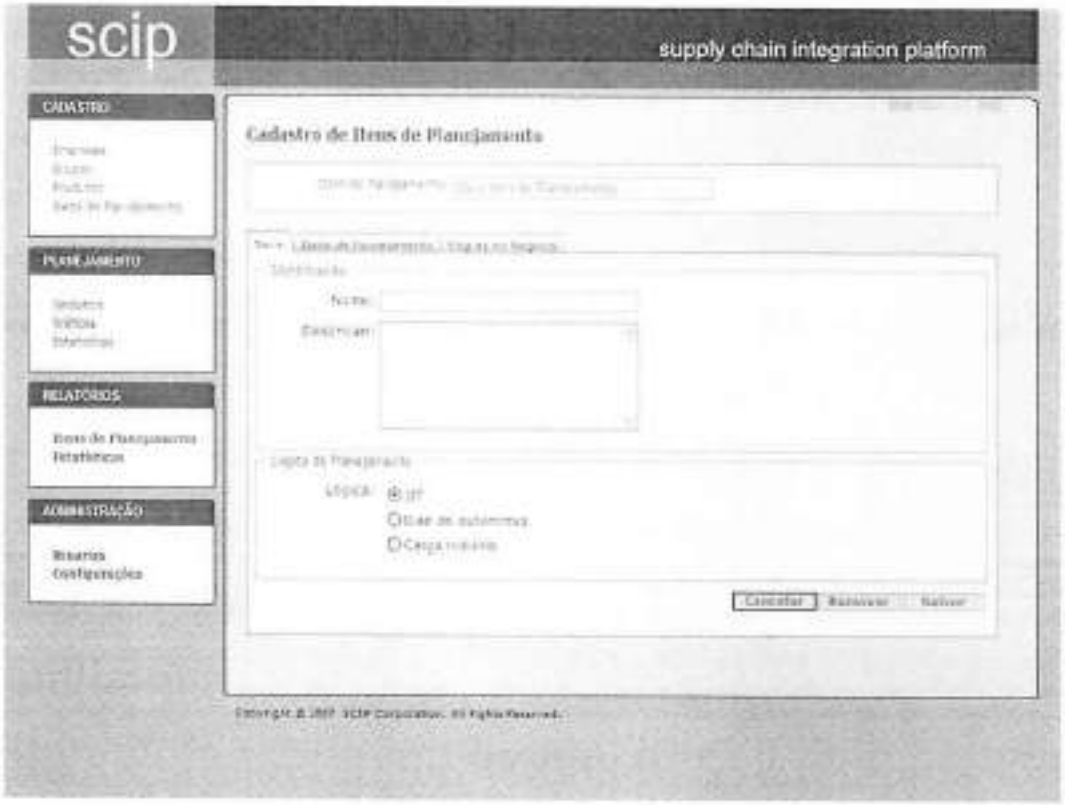


Figura I.1: Cadastro de Item de Planejamento.

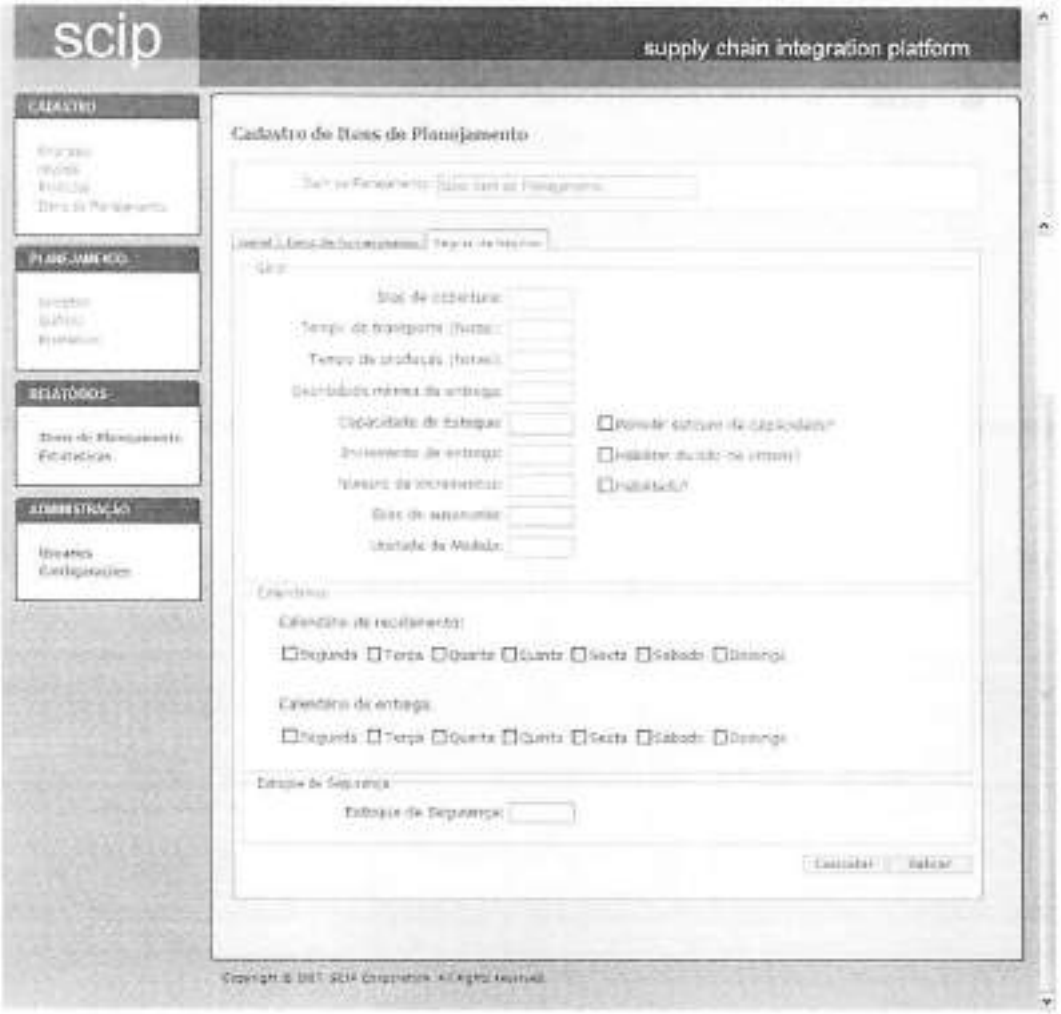


Figura I.2: Cadastro de item de planejamento.

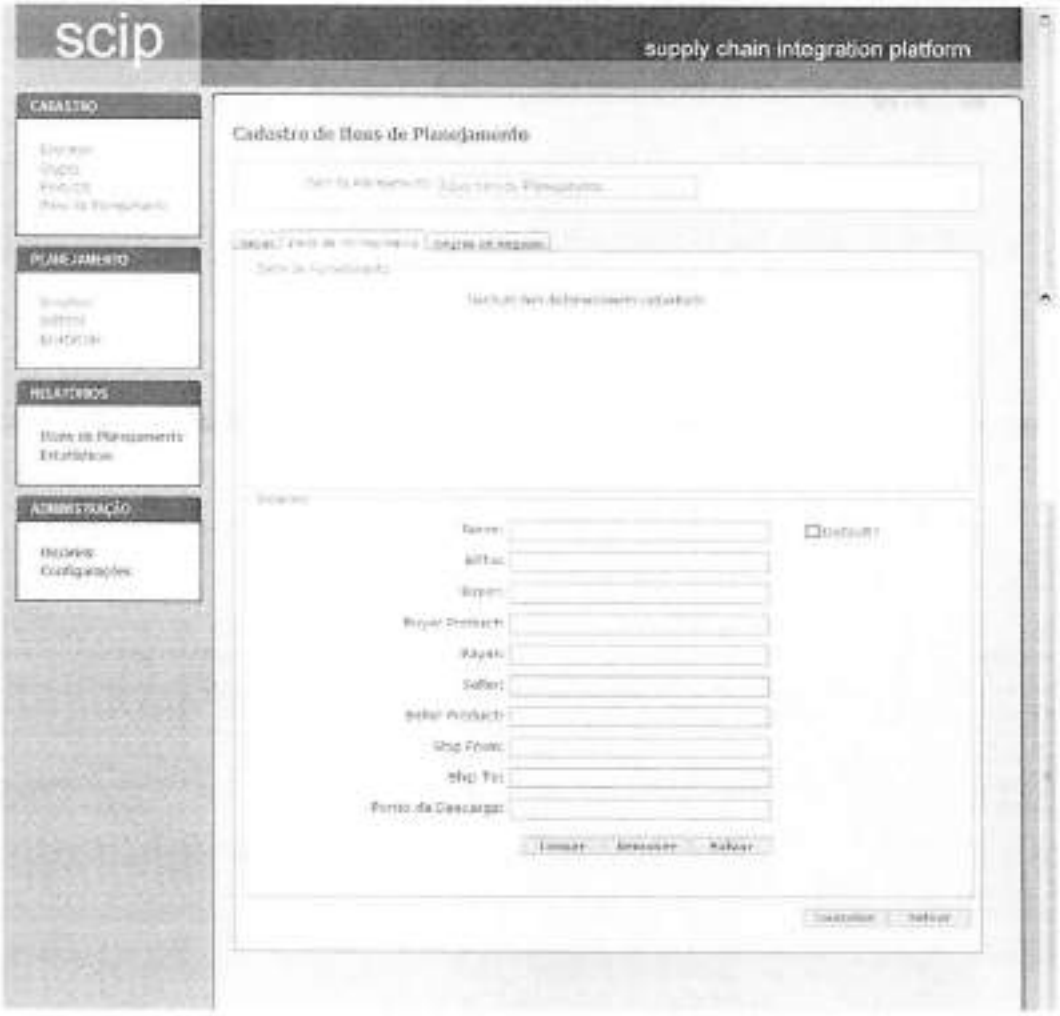


Figura I.3: Cadastro de Item de Fornecimento.

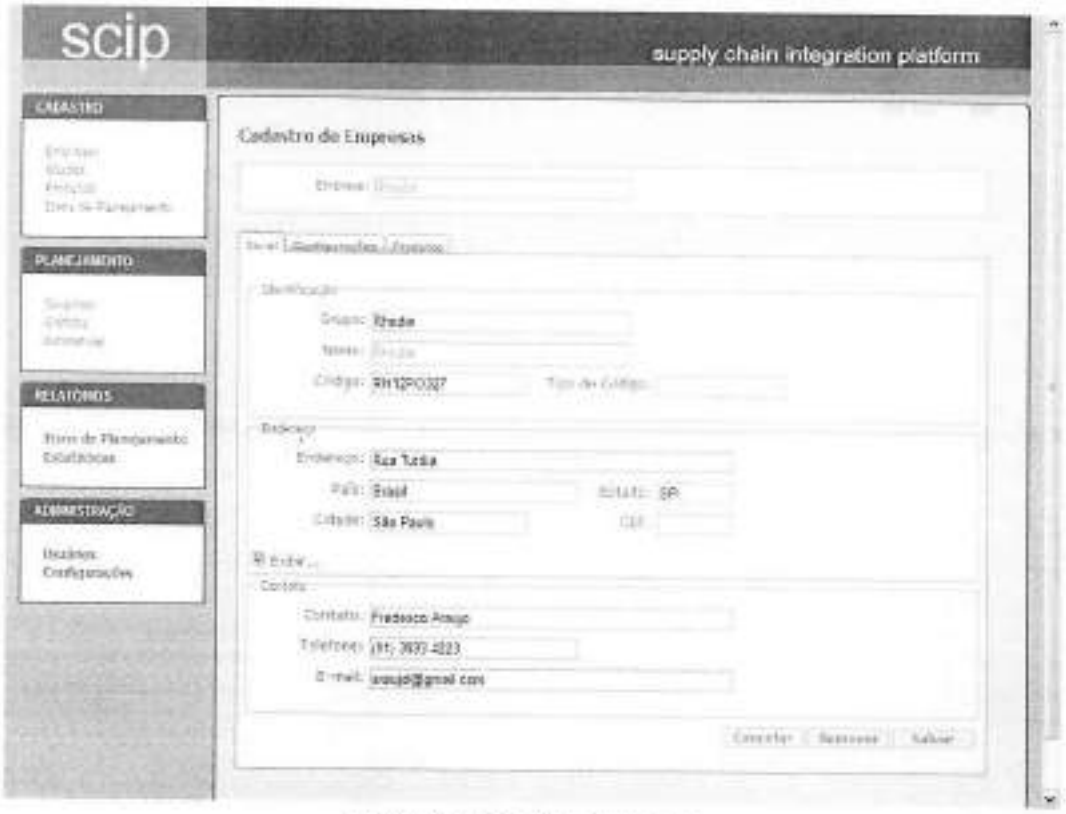


Figura I.4: Cadastro de Empresas.

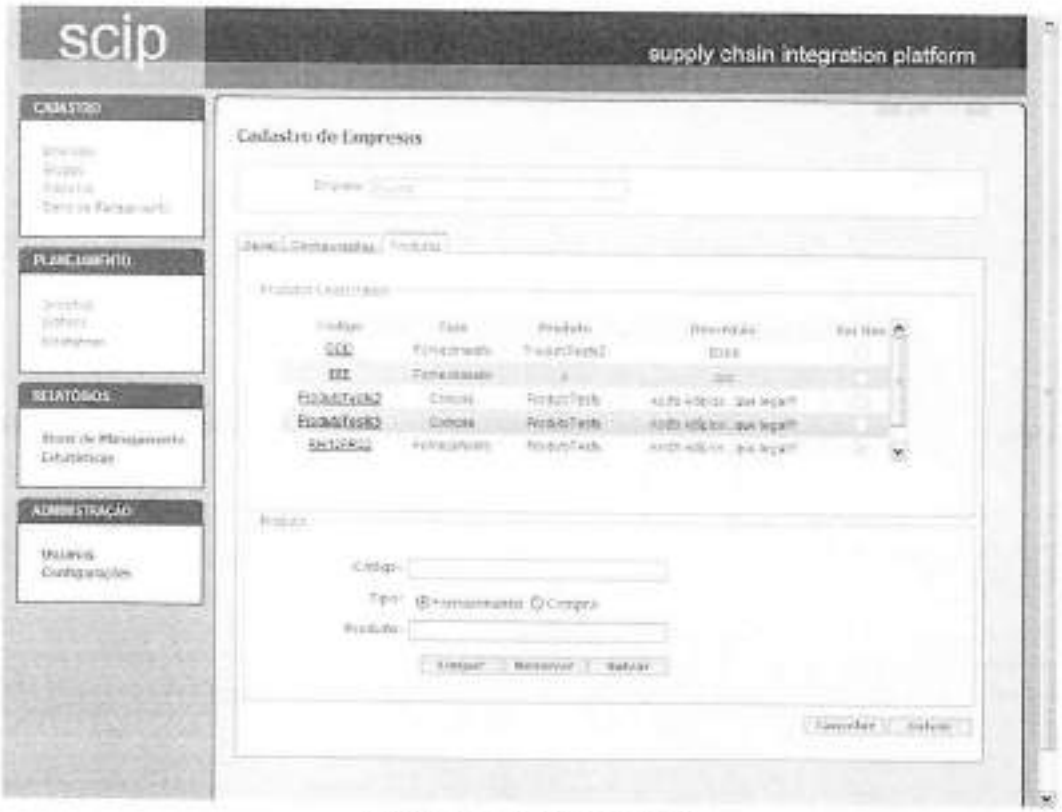
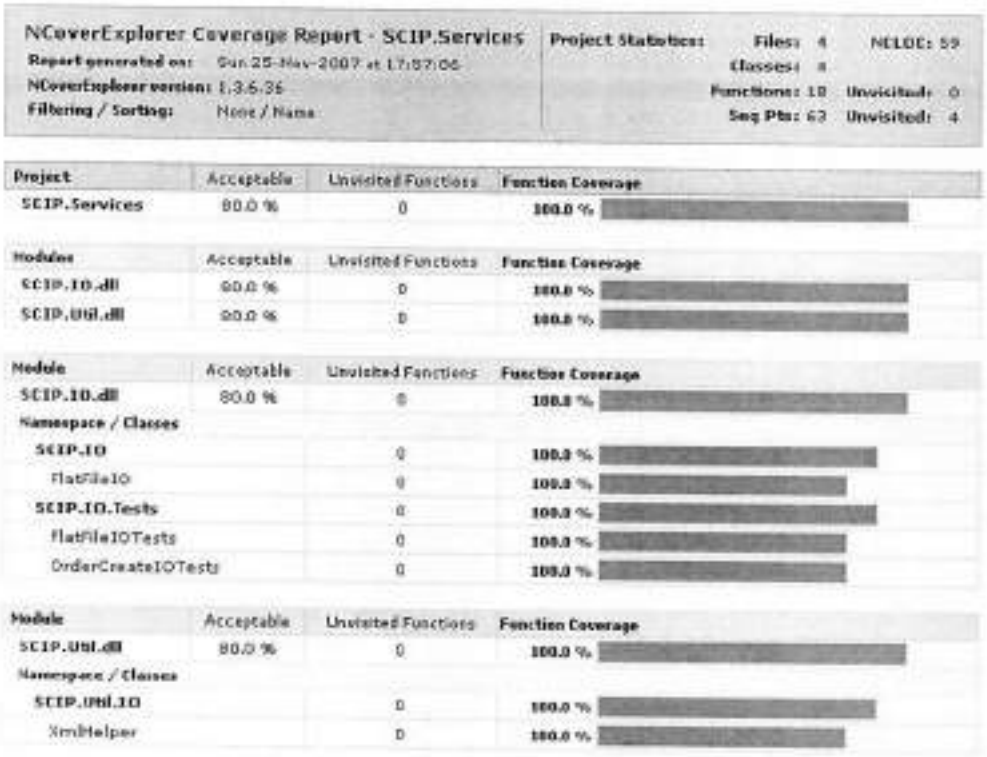


Figura I.5: Cadstro de Empresa.

APÊNDICE J – TESTES

J.1 Testes Unitários

A seguir são apresentados resultados dos testes unitários testados com a ferramenta NCoverExplorer que verifica a cobertura dos testes.



NCoverExplorer Coverage Report - SCIP.Services

Report generated on: Sun 25-Nov-2007 at 17:07:06
NCoverExplorer version: 1.3.6.36
Filtering / Sorting: None / Name

Project Statistics: Files: 4 NCLOC: 59
Classes: 4
Functions: 10 Unvisited: 0
Seq Pts: 62 Unvisited: 4

Project	Acceptable	Unvisited Functions	Function Coverage
SCIP.Services	80.0 %	0	100.0 %

Modules	Acceptable	Unvisited Functions	Function Coverage
SCIP.IO.dll	80.0 %	0	100.0 %
SCIP.Util.dll	80.0 %	0	100.0 %

Module	Acceptable	Unvisited Functions	Function Coverage
SCIP.IO.dll	80.0 %	0	100.0 %
Namespace / Classes			
SCIP.IO		0	100.0 %
FlatFileIO		0	100.0 %
SCIP.IO.Tests		0	100.0 %
FlatFileIOTests		0	100.0 %
OrderCreateIOTests		0	100.0 %

Module	Acceptable	Unvisited Functions	Function Coverage
SCIP.Util.dll	80.0 %	0	100.0 %
Namespace / Classes			
SCIP.Util.IO		0	100.0 %
XmlHelper		0	100.0 %

Figura J.1: Cobertura dos testes unitários – parte 1.

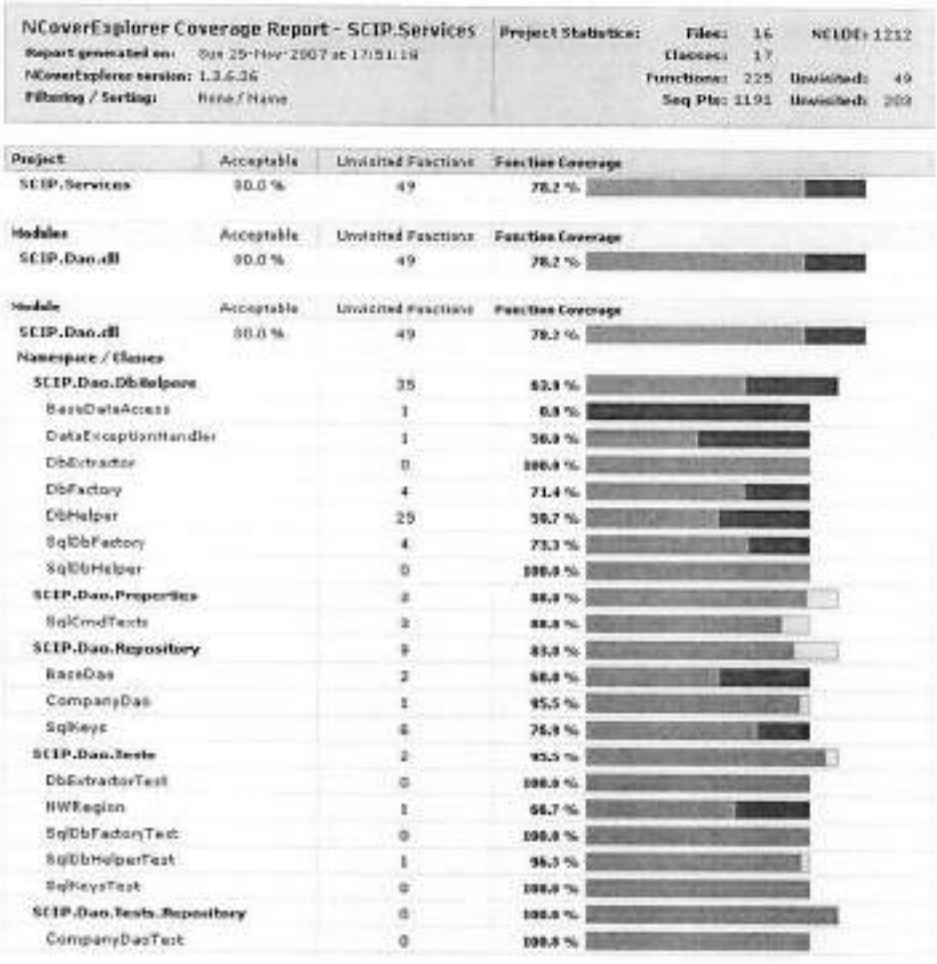


Figura J.2: Cobertura dos testes unitários – parte 2.

Figura J.3: Cobertura dos testes unitários – parte 3.

J.2 Plano de Testes Integrados

Função
Cadastrar uma nova empresa no sistema SCIP.
Procedimento de Teste
1. **Ação:** Abrir tela de cadastro de empresas 2. **Ação:** Entrar com os dados de empresa • Grupo: • nome • Descrição: • Endereço: • Contato: 3. **Ação:** Acionar botão Salvar; 4. **Verificar:** Observar que foi incluída na listagem de empresas a empresa previamente inserida; 5. **Ação:** Selecionar a empresa na listagem; 6. **Verificar:** Observar que os dados da empresa foram exibidos corretamente; 7. **Ação:** Selecionar a tab Configurações 8. **Ação:** Adicionar um ponto de descarga: • Nome: • Descrição: 9. **Ação:** Acionar o botão inserir 10. **Verificar:** Inserção do ponto na tabela 11. **Ação:** Acionar o botão editar para o ponto inderido e alterar descrição 12. **Ação:** Acionar o botão confirmar 13. **Verificar:** Alteração realizada, atualização da tabela com as modificações
☐ Teste Aprovado ☐ Teste aprovado com comentários ☐ Teste Reprovado
Comentários ou Motivo de Reprovação

Função
Cadastrar uma novo produto no SCIP.
Procedimento de Teste

1. **Ação:** Abrir a tela de cadastro de produtos;
2. **Ação:** selecionar a empresa a qual o produto pertence:
  • Empresa
3. **Ação:** Entrar com os dados do produto:
  • Nome
  • Código
  • Descrição
4. **Ação:** Acionar botão Salvar;
5. **Verificar:** Observar que foi incluída na listagem da o produto correspondente
6. **Ação:** Selecionar o produto da listagem de produtos da empresa;
7. **Verificar:** Observar que os dados do produto foram exibidos corretamente;

Resultado
☐ Teste Aprovado ☐ Teste aprovado com comentários ☐ Teste Reprovado
Comentários ou Motivo de Reprovação

Função
Cadastrar um novo Item de Planejamento no sistema SCIP.
Procedimento de Teste
1. **Ação:** Abrir a tela de cadastro de Item de Planejamento 2. **Ação:** Entrar com os dados do item • Descrição: • Descrição: 3. **Ação:** Acionar botão para adicionar um item de planejamento; 4. **Verificar:** Abertura do formulário para cadastro de um item de fornecimento; 5. **Ação:** Selecionar as empresas participantes: • Fornecedor: • Comprador: • Payer: • BillTo: 6. **Verificar:** Observar que campo de produtos se restringem aos disponíveis para o cliente e fornecedor, 7. **Ação:** Selecionar os Produtos: • sellerProduct: • buyerProduct: 8. **Ação:** Salvar modificações 9. **Verificar:** Verificar validação e adição do item de fornecimento ao item de planejamento. Volta para tela de cadastro do item de planejamento. 10. **Ação:** Cadastrar regras de negócio: 11. **Ação:** Salvar modificações; 12. **Verificar:** Validação e armazenamento do item de planejamento na base.
Resultado
☐ Teste Aprovado ☐ Teste aprovado com comentários ☐ Teste Reprovado
Comentários ou Motivo de Reprovação

Função
Visualização do sinóptico e um item de planejamento e aprovação de ordem
Procedimento de Teste
1. **Ação:** Abrir a tela de sinóptico 2. **Ação:** selecionar o item de planejamento pelo critério: • Fornecedor: 3. **Verificar:** Lista de resultados envolvendo a empresa selecionada; 4. **Ação:** Selecionar o item de planejamento com a seguinte Descrição: • Descrição: 5. **Verificar:** Exibição dos dados do item de planejamento e exposição do sinóptico do item, contendo estoque projetado e ordens agendadas. 6. **Ação:** Selecionar uma ordem planejada. 7. **Verificar:** Abertura da tela de detalhes da ordem selecionada indicando quantidade e estado atual da ordem. 8. **Ação:** Aprovar a ordem planejada acionando o botão aprovar. 9. **Verificar:** Verificar a alteração do estado da ordem para aprovado. 10. **Verificar:** Verificar a criação de um arquivo XML na pasta de saída contendo uma mensagem Order Create com os dados correspondentes da ordem aprovada.
Resultado
☐ Teste Aprovado ☐ Teste aprovado com comentários ☐ Teste Reprovado
Comentários ou Motivo de Reprovação

Função
Fornecer um XML de *Order Response* ao Sistema.
Procedimento de Teste
1. **Ação**: Fornecer um arquivo XML *Order Response* para uma ordem existente no sistema com status esperando por resposta na pasta de entrada; 2. **Ação**: Abrir tela de sinóptico correspondente; 3. **Verificar**: Observar que o estado da ordem foi mudado para confirmado; Verificar que a quantidade e data corresponde aos valores da mensagem fornecida;
Resultado
☐ Teste Aprovado ☐ Teste aprovado com comentários ☐ Teste Reprovado
Comentários ou Motivo de Reprovação

Função
Fornecer um arquivo de *Ship Notice* ao sistema.
Procedimento de Teste
1. **Ação**: Inserir um arquivo XML Ship Notice na pasta de entrada, referente a uma ordem confirmada do sistema; 2. **Ação**: abrir tela de visualização do sinóptico do item de planejamento correspondente; 3. **Verificar:** Verificar que ordem correspondente está em estado enviado;
Resultado
☐ Teste Aprovado ☐ Teste aprovado com comentários ☐ Teste Reprovado
Comentários ou Motivo de Reprovação

MULTIOFÍCIO
inovação em serviços gráficos
www.multioficio.com.br
multioficio@multioficio.com.br
(11) 3034.6085 - 3032.1001