FACULDADES INTEGRADAS DE TAQUARA

CURSO DE SISTEMAS DE INFORMAÇÃO

IMPLEMENTAÇÃO DE UM SOFTWARE USANDO ALGORITMO GENÉTICO PARA A SOLUÇÃO DO PROBLEMA DE CARREGAMENTO DE CONTENTOR DE CARGA E SEMELHANTES

JOSÉ ALENCAR PHILERENO

Taquara
2009

JOSÉ ALENCAR PHILERENO

IMPLEMENTAÇÃO DE UM SOFTWARE USANDO ALGORITMO GENÉTICO PARA A SOLUÇÃO DO PROBLEMA DE CARREGAMENTO DE CONTENTOR DE CARGA E SEMELHANTES

Trabalho de Conclusão apresentado ao Curso de Sistemas de Informação das Faculdades Integradas de Taquara, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação, sob orientação do Prof. Dr. Paulo Roberto Ferreira Junior.

Taquara

2009

Aos meus pais a dedicação e empenho para que eu estudasse, à minha família por me suportar nas horas de estresse, aos amigos e colegas a compreensão dispensada e acima de tudo à Deus por ter colocado todos vocês no meu caminho.

AGRADECIMENTOS

Agradeço ao Prof. Dr. Paulo Ferreira Junior, que mesmo não fazendo mais parte do corpo docente das Faculdades Integradas de Taquara, continuou acreditando e apoiando o meu trabalho.

Agradeço aos professores (as) da faculdade o apoio e votos de sucesso durante a jornada do curso, em especial ao Prof. Francisco Assis do Nascimento pela dedicação dispensada durante este trabalho.

Agradeço aos colegas de curso as incansáveis parcerias e ajudas, em especial ao Luciano Hoffmaister Ribeiro as inúmeras informações sobre formatação usando a ferramenta de edição.

RESUMO

A otimização do carregamento de volumes diversificados em ambientes fechados como os *containers*, visando à agilidade na tarefa sem perder o foco na ocupação perfeita do espaço, é motivo de estudo e preocupação a algum tempo.

Existem muitas soluções no mercado com excelentes resultados, porém todas proprietárias, de alto custo e fechadas.

Este trabalho tem como objetivo implementar o algoritmo proposto no artigo de Gehring e Bortfeldt (1997), somando ao resultado final algumas funções adicionais, como a utilização do *software* de visualização de carga de Uriartt (2007) além de permitir que os ERPs integrem com o sistema através de arquivos XML, facilitando a troca de informações de maneira ágil e aberta.

Outro foco importante deste trabalho é o fato da ferramenta ser desenvolvida em linguagem Java com código aberto e gratuito, criando a possibilidade de alterações e continuação do seu desenvolvimento.

Palavras-chave: Problema de Carregamento de *Container*. Algoritmo genético. *Container Loading Problem.*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

3D - Imagem em três dimensões

API - *Application Programming Interface*

ASA - *American Standards Association*

CDDL - *Common Development and Distribution License*

CLP - *Container Loading Problem*

DFD – Diagrama de Fluxo de Dados

DOM - *Document Object Model*

ERP - *Enterprise Resource Planning*

GA – *Genetic Algorithm*

IDE - *Integrated Development Environment*

ISO - *International Standards Organization*

OMG - *Object Management Group*

PDF - *Portable Document Format*

UML - *Unified Modeling Language*

XML - *Extensible Markup Language*

XSD - XML Schema Definition

SUMÁRIO

1 INTRODUÇÃO

Em grande parte da cadeia produtiva mundial, existe o problema de acondicionamento da carga para entrega em caminhões ou contentores de carga, de volumes diversos e assimétricos, com suas regras e diretrizes de ordem de carregamento e empilhamento (DUBKE *et al.*, 2004).

Este trabalho de lógica é executado no estágio após a produção, tanto no setor de expedição como no setor de empacotamento do produto, e feito de forma a comprometer a produtividade e sem vistas ao melhor resultado do uso do espaço.

Igualmente, não havendo uma sistemática de carregamento, pode-se perder, através da sobra ou falta de espaço, em valores expressos nos fretes por cubagem ou no consumo de combustível ao transportar espaços vazios.

O problema já bastante conhecido de carregamento de contentores de carga (*Container Loading Problem* – Problema de Carregamento de *Container*), mais normalmente conhecido por *containers*, foi tratado por vários autores de diferentes formas, com heurísticas e métodos os mais variados (MORABITO e ARENALES, 1994).

O CLP consiste basicamente em acomodar dentro de um ambiente tridimensional, uma dada carga de volumes de tamanhos e pesos variados, visando-se a otimização do espaço e, consequentemente, o carregamento da maior quantidade de volumes possíveis. Além disso, outras regras devem ser observadas, tais como a estabilidade da carga, o total de peso permitido, as orientações possíveis de cada volume (seus giros possíveis), o custo/valor da carga carregada, etc.

Conforme Klein (2006), o CLP é um problema que se encontra na categoria NP-difíceis, pois o mesmo não possui uma solução de ordem polinomial e geralmente é de ordem exponencial ou de ordem fatorial.

O escopo deste trabalho é implementar o CLP com o algoritmo genético proposto por Gehring e Bortfeldt (1997) que obteve bons resultados nos seus testes, visando oferecer uma forma amigável de acoplamento fraco com os ERPs através da troca de mensagens por arquivo no formato XML.

Além disso, a implementação oportuniza o envio das informações do carregamento ao *software* de visualização 3D, desenvolvido por Uriartt (2007), também de maneira independente e autônoma.

1.1 Por que mais uma implementação?

São várias as formas de se resolver o CLP, todas funcionais e bastante testadas.

As variantes deste problema, podendo ser numerosamente expressivas, trazem resultados os mais diversos possíveis, podendo alguns ter melhoras no aproveitamento do espaço, melhoras no desempenho computacional e/ou melhoras na aplicação das restrições e regras de carregamento usuais (KLEIN, 2006).

Contudo, o uso do algoritmo genético proposto por Gehring e Bortfeldt (1997) teve uma significativa melhora em todos os pontos descritos, conforme testes comparativos publicado no documento base deste trabalho.

Na bateria de testes, foram comparados seis problemas, denominados GB1 até GB6 caracterizados na Figura 1, diferentes entre três métodos, sendo CBGMM sugerido por Gehring *et al*. (1990) *apud* Gehring e Bortfeldt (1997), CL sugerido por Scheithauer (1992) *apud* Gehring e Bortfeldt (1997) e CBGAT por Gehring e Bortfeldt (1997), este último objeto deste estudo (GEHRING e BORTFELDT, 1997).

A comparação entre o método CBGAT e CL somente foi possível no teste GB6, visto o método utilizado no algoritmo CL somente prever carga em pequenos volumes. Os demais testes foram baseados em *containers* de 40’[1].

Algumas restrições foram consideradas neste teste, tais como a estabilidade da carga, a orientação dos objetos, aceitação de empilhamento, capacidade do ambiente de carga.

Deve-se observar também que nenhum dos métodos concorrentes utiliza a regra de estabilidade. Portanto, esta regra é ignorada pelos métodos CBGMM e CL.

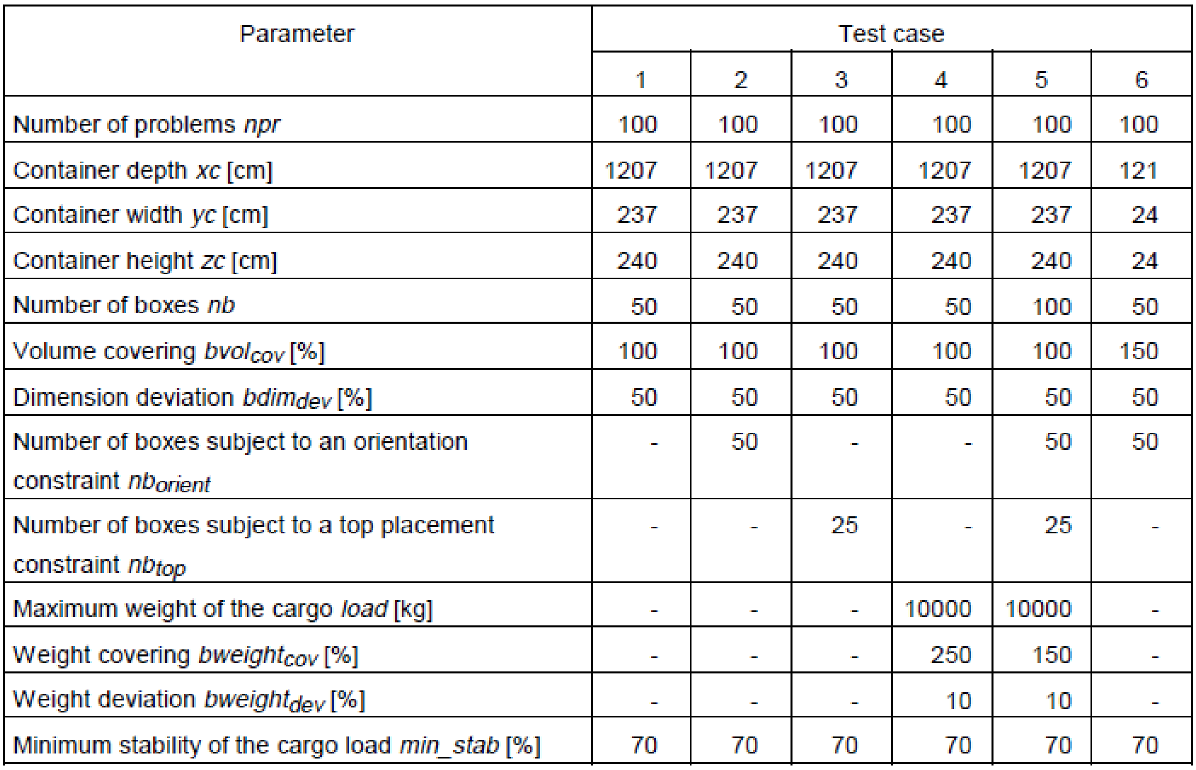
Nos testes GB1 até GB5, a melhora na utilização do volume do *container* ficou entre 1,19% e 5,33%, dependendo da regra utilizada. No teste GB6, excedeu 5,03% (GEHRING e BORTFELDT, 1997).

---

[1] 40’ (quarenta pés): tamanho padrão comercial de 11,56 m de comprimento, 2,29 m de largura, 2,23 m de altura e 2,13 m de altura útil. Fonte: http://www.greville.com.br/especificacoes.html

O tempo de computação do método CBGAT somou menos que 6 minutos para todos os testes.

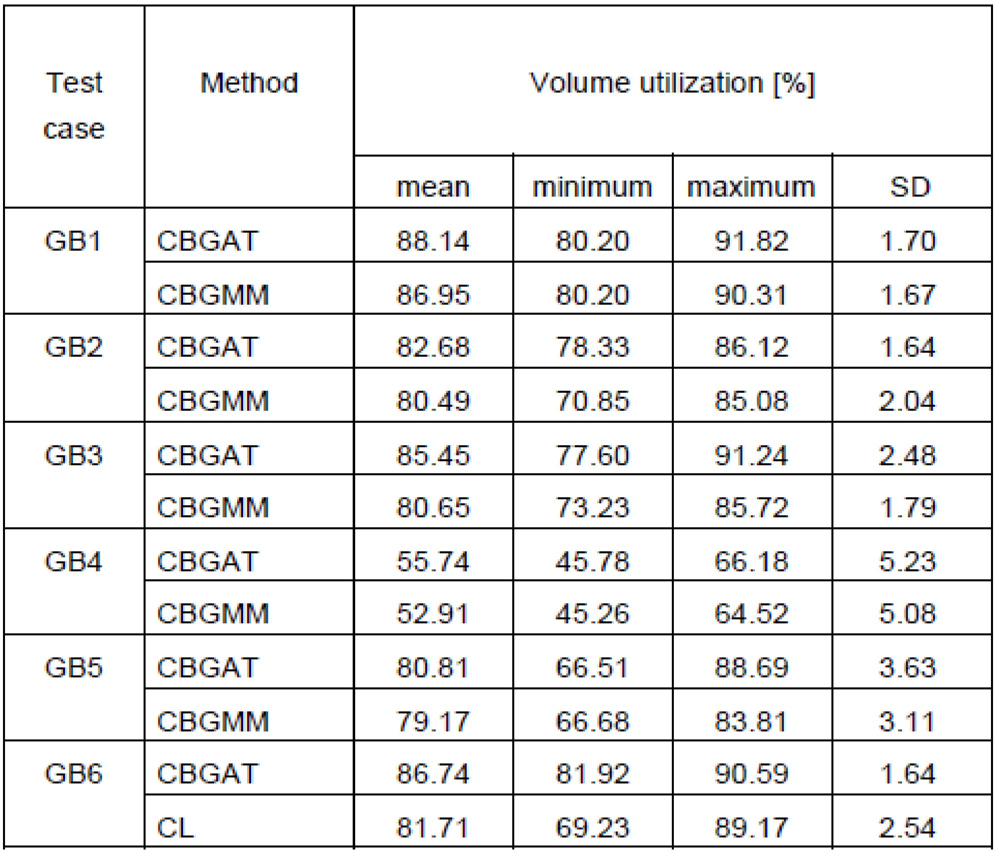
Na Figura 1, constam os valores de cada parâmetro utilizados nos testes.



Parameter	Test case					
	1	2	3	4	5	6
Number of problems npr	100	100	100	100	100	100
Container depth xc [cm]	1207	1207	1207	1207	1207	121
Container width yc [cm]	237	237	237	237	237	24
Container height zc [cm]	240	240	240	240	240	24
Number of boxes nb	50	50	50	50	100	50
Volume covering $bvol_{cov}$ [%]	100	100	100	100	100	150
Dimension deviation $bdim_{dev}$ [%]	50	50	50	50	50	50
Number of boxes subject to an orientation constraint nb_{orient}	-	50	-	-	50	50
Number of boxes subject to a top placement constraint nb_{top}	-	-	25	-	25	-
Maximum weight of the cargo $load$ [kg]	-	-	-	10000	10000	-
Weight covering $bweight_{cov}$ [%]	-	-	-	250	150	-
Weight deviation $bweight_{dev}$ [%]	-	-	-	10	10	-
Minimum stability of the cargo load min_stab [%]	70	70	70	70	70	70

Figura 1: Parâmetros para a montagem dos testes GB1 até GB6.
Fonte: International Transactions in Operational Research (1997, p. 415)

Na Figura 2, os resultados médio, mínimo e máximo de cada teste para os métodos correspondentes na sua linha de informação, referentes ao percentual de utilização do volume.



Test case	Method	Volume utilization [%]			
		mean	minimum	maximum	SD
GB1	CBGAT	88.14	80.20	91.82	1.70
	CBGMM	86.95	80.20	90.31	1.67
GB2	CBGAT	82.68	78.33	86.12	1.64
	CBGMM	80.49	70.85	85.08	2.04
GB3	CBGAT	85.45	77.60	91.24	2.48
	CBGMM	80.65	73.23	85.72	1.79
GB4	CBGAT	55.74	45.78	66.18	5.23
	CBGMM	52.91	45.26	64.52	5.08
GB5	CBGAT	80.81	66.51	88.69	3.63
	CBGMM	79.17	66.68	83.81	3.11
GB6	CBGAT	86.74	81.92	90.59	1.64
	CL	81.71	69.23	89.17	2.54

Figura 2: Resultados para os testes GB1 até GB6.
A coluna SD (Standard Deviation) é o desvio padrão referente à média.
Fonte: International Transactions in Operational Research (1997, p. 415)

No quesito estabilidade, o algoritmo CBGAT, único do teste que considera esta restrição, alcançou um percentual de afastamento do centro de gravidade do ambiente bem aquém do permitido na parametrização, o que realça a viabilidade do uso do mesmo.

Segundo Gehring e Bortfeldt (1997) *apud* Gehring e Bortfeldt (1997), os resultados alcançados através da aplicação do algoritmo genético para problemas de complexidade maior, tais como balanceamento de linha (GEHRING e SCHÜTZ, 1994) e outras restrições práticas, falam a favor da abordagem genética.

Com base nos resultados obtidos pelo algoritmo citado, o presente trabalho o implementou com a linguagem de programação Java visando a livre disponibilização do uso e seu código aberto para que outras pessoas interessadas possam dar continuação na sua melhoria.

A visualização oferecida pelo *software* de carregamento em 3D, permitirá ao usuário final dirimir dúvidas do carregamento, comparando as informações do relatório - resultado do processo deste trabalho - com a imagem dos volumes montados pelo visualizador.

Segundo Carlis (1999) e Gershon (1997) *apud* Kirner *et al* (2004, p. 1):

> A visualização de informação é responsável por mapear conjuntos de dados disponíveis, muitas vezes, em formato baseado em texto (incluindo descrições textuais, informação numérica mostrada em tabelas, etc.), representando-os em um formato visual, visando assistir os usuários na exploração e entendimento de tais conjuntos de dados.

A integração com outros *softwares* de gestão (ERPs) trará agilidade na comunicação, diminuindo possibilidades de erros em digitação. Segundo Martins (2002, p. 2), “[...] é impossível trabalhar sem a troca de dados e compartilhamento de processos entre outros setores”.

Com vistas a estas duas integrações, este trabalhou desenvolveu formatos de arquivos no padrão XML para fornecer informações ao visualizador e receber dados da carga diretamente dos programas de gestão.

Existem soluções prontas no mercado que contemplam o problema, como o CargoWiz[2], *software* americano, comercializado por licenças sem necessidade de renovações, mas que trabalha de forma isolada, *stand-alone*[3], sem possibilidade de integração com os ERPs da empresa.

O diferencial deste trabalho é a obtenção dos resultados semelhantes aos *softwares* comerciais, mas com custo zero e aberto à manutenção e/ou alteração pelo próprio usuário.

1.2 Objetivos

O principal objetivo do presente trabalho é desenvolver um *software* usando o algoritmo genético para a solução de carga em contentores e semelhantes, observando suas regras de carregamento e as restrições do ambiente a ser

---

[2] CargoWiz – *Software* de propriedade de Softtruck

[3] De maneira autônoma – Dicionário Michaelis

carregado, como medidas e capacidade de peso suportada.

As informações sobre a carga, o ambiente a ser carregado e as regras de carregamento são inseridas através de uma *interface*[4] do sistema.

O trabalho também desenvolveu um processo de entrada de dados através de arquivos no padrão XML, o qual será validado pelo *Schema* XML publicado no apêndice A, garantindo-se assim a integridade das informações necessárias ao funcionamento do CargaFacil. Os resultados obtidos no processamento da carga são externados no formato de relatório em PDF com a sequência dos volumes a serem carregados, na ordem do fundo para frente do ambiente, da esquerda para direita e em um arquivo no formato XML, com as mesmas informações da ordem de carregamento, porém com as coordenadas cartesianas de cada volume, sendo *x* a coordenada de profundidade, *y* a coordenada de largura e *z* a coordenada de altura do ambiente de carga. Com estas informações de posicionamento de cada volume, o programa opcionalmente poderá executar o visualizador referido já com as devidas informações que lhe são necessárias ao seu funcionamento.

1.3 Alguns conceitos necessários

Nesta seção são dados alguns esclarecimentos e conceitos sobre o que significa contentor, no que consiste um *container* e um caminhão furgão.

1.3.1 Contentor

O CLP refere-se exclusivamente a *containers* que são usados como ambientes de acomodação de carga para transporte, principalmente marítimo.

Mas o problema pode ser estendido para outros ambientes, tais como caminhão furgão, caminhonetas fechadas e até mesmo para ambientes estáticos de

---

[4] Tela de entrada de dados para o usuário final

armazenamento, por exemplo, uma sala de estocagem.

O que muda de um ambiente para outro são suas diretrizes de carregamento, regras de acomodação, limites de peso, *etc*.

1.3.2 Container

Conforme a Grande Enciclopédia Larousse Cultural (1988, p. 1604):

> Caixa volumosa de embalagem para transporte, a longa distância (princ. marítimo), de variadas mercadorias; cofre de carga. (Os *containers* são padronizados pela International Standards Organization, ISO.)

A definição dos padrões que norteiam a construção de *containers* está especificada pela norma ISO (*International Standards Organization*) embora também seja aceita em alguns países a norma ASA (*American Standards Association).*

O Brasil, por ter adotado a norma europeia, regulamentou a construção e uso destas embalagens baseando-se nas diretrizes da norma, controlada pelos órgãos Associação Brasileira de Normas Técnicas (ABNT) e Instituto de Metrologia, Normalização e Qualidade Industrial (Inmetro) (NOVO MILÊNIO, 2009).

No Artigo 4º do Decreto nº 80.145 de 15 de agosto de 1977, a definição brasileira:

> O *container* é um recipiente construído de material resistente, destinado a propiciar o transporte de mercadorias com segurança, inviolabilidade e rapidez, dotado de dispositivo de segurança aduaneira e devendo atender às condições técnicas e de segurança previstas pela legislação nacional e pelas convenções internacionais ratificadas pelo Brasil

Na Figura 3, um exemplo de ambiente de carga do tipo *container* sendo carregado.

Figura 3: *Container* de 9'6" de altura e 40 pés de comprimento
Fonte: Novo Milênio (2009)

1.3.3 Caminhão furgão

O transporte tipo furgão é usado para curtas e médias distâncias, onde a carga deva ser protegida de acesso indevido e contra as intempéries.

De modo semelhante ao *container*, o carregamento se faz do fundo do baú para o início, com a mesma dificuldade de ajuste da carga, independentemente do tamanho do ambiente.

Exemplificando o ambiente, tem-se na Figura 4, à esquerda da imagem, um furgão tipo carreta no modo estacionário (sob suporte próprio), sem o caminhão acoplado.

Figura 4: Furgão tipo carreta para tração por caminhão
Fonte: Comprecar (2009)

Na Figura 5, outro modelo de furgão fixo no caminhão.

Figura 5: Furgão acoplado ao caminhão
Fonte: Carrocerias Rocha (2009)

Independentemente do tipo ou tamanho do furgão, o ambiente assemelha-se ao *container* por ter tamanho útil definido e fixo.

Na próxima seção, serão apresentadas as fundamentações teóricas relativas ao CLP e algumas soluções possíveis.

2 UM GRANDE PROBLEMA COM VÁRIAS SUGESTÕES DE SOLUÇÃO

Nesta seção serão apresentadas as fundamentações teóricas relativas ao CLP e algumas soluções possíveis.

2.1 O problema de carregamento de *containers*

Segundo Gehring e Bortfeldt (1997), o problema de carregamento de *containers* pode ser agrupado em diferentes maneiras. Uma das distinções básicas existentes entre elas está no fato de que haver casos em que um dado conjunto de objetos deve ser carregado na sua totalidade e casos em que alguns destes objetos podem ser deixados de fora.

Outra importante distinção, conforme Bortfeldt (1994) consiste na forma dos objetos da carga, que podem ser:

a) de igual forma, sendo considerado um caso de carga homogênea;
b) de formas misturadas mas com poucas variações[5], sendo considerado um caso de carga fracamente heterogênea;
c) de formas misturadas, e com muitas variações, sendo considerado um caso de carga fortemente heterogênea;

O algoritmo proposto neste trabalho focará os casos considerados fortemente heterogêneos e em apenas um ambiente de carga.

O método heurístico de Portmann (1990) *apud* Gehring e Bortfeldt (1997), preenche um dado *container* de baixo para cima, considerando o proposto por Gehring *et al.* (1990), a montagem de camadas verticais em paralelo com a frente do ambiente e o preenchendo camada a camada. O algoritmo de Scheithauer (1992) *apud* Gehring e Bortfeldt (1997), é baseado na estratégia da necessidade da entrega, enquanto a abordagem feita por Morabito e Arenales (1994), trata o *container* em partes guilhotinadas em vários pequenos ambientes a serem carregados (GEHRING e BORTFELDT, 1997).

---

[5] Não existe uma definição de quantidade de variações nas literaturas

A Figura 6 mostra um exemplo de guilhotinamento proposto por Morabito e Arenales (1994). O volume A foi repartido em B e C. Ao lado, uma amostragem do particionamento.

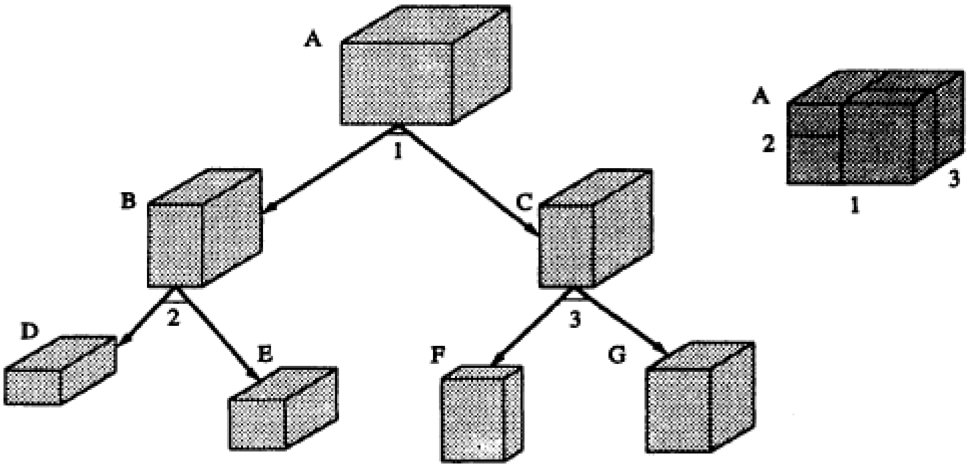


Figura 6: Exemplo da abordagem AND/OR
Fonte: Morabito e Arenales (1994, p. 62)

Finalizando, os algoritmos propostos por Loh e Nee (1992), Ngoi *et al.* (1995), Bischoff *et al.* (1995) e Bischoff e Ratcliff (1995) são adaptados para uma carga fracamente heterogênea, o que foge do escopo deste trabalho.

A escolha pelo algoritmo genético de Gehring e Bortfeldt (1997) baseia-se nos resultados obtidos nos testes de comparação e pelo fato do mesmo abordar a restrição de estabilidade da carga.

Com base nos testes referidos na subseção 1.1, Figuras 1 e 2, ratificando ainda a escolha do CBGAT, algumas considerações devem ser observadas:

a) todos os testes contêm a regra de estabilidade mínima, denotada como C4;
b) os testes GB2 e GB6 contêm a regra de orientação de objetos C1;
c) o teste GB3 contém a regra de máximo de volumes empilhados C2;
d) o teste GB4 contém a regra de capacidade máxima de carga C3;
e) o teste GB5 contém todas as regras C1, C2 e C3 juntas.

Fora o teste GB4, a distância média entre o centro de gravidade da carga e a metade da profundidade do *container* e a metade da largura ficou menos que 5% das medidas do mesmo. Considerando o tamanho do *container* de 40’, este percentual corresponde a 60 cm de distância em relação à profundidade e 12 cm em relação à largura, considerados estes valores bastante aceitáveis, conforme se observa na última linha de parâmetro da Figura 1 (GEHRING e BORTFELDT, 1997).

A média de utilização do espaço do *container* também se destaca nos resultados obtidos.

2.2 Formulação do problema

Com base na prática de carregamento de volumes, para o qual aqui se entende por caixas, caixotes ou semelhantes, apenas arranjos estáveis são aceitos, ou seja, empilhados de maneira que não caiam por si só. Com isso, o uso de material adicional, por exemplo, espumas de preenchimento, não se faz necessário.

Para esta condição, Gehring e Bortfeldt (1997) definem três regras básicas de viabilidade:

a) cada volume deve estar apoiado no chão do ambiente ou sobre outro volume do qual seu centro de gravidade suporte o volume de cima. Esta regra é representada pela sigla F1;
b) cada volume deve estar posicionado paralelamente aos lados do ambiente. Esta regra é representada pela sigla F2;
c) cada volume deve estar completamente dentro do ambiente. Esta regra é representada pela sigla F3.

Para a primeira regra de viabilidade (F1), assume-se que o centro de gravidade do volume é sempre seu centro geométrico.

O ambiente referido, podendo este ser um *container*, um furgão de caminhão ou outro semelhante, deve ser um espaço tridimensional, definido pelas coordenadas cartesianas x, y e z, representando a profundidade, largura e altura respectivamente.

Formulando-se o problema do CLP tem-se: dado um ambiente de carga com dimensões de *xc* de profundidade, *yc* de largura e *zc* de altura e um conjunto *nb* de volumes retangulares *b*, onde *b = 1...nb*, com dimensões *bdim1(b), bdim2(b) e bdim3(b)*, peso *bweight(b)* e valor *bvalue(b)* > 0, determine um arranjo viável de subconjuntos de volumes dentro do ambiente tanto que o valor total dos volumes carregados seja maximizado assim como as restrições relevantes sejam respeitadas.

O valor de cada volume pode ser o espaço por ele ocupado ou o valor agregado, como por exemplo, o valor do frete ou do produto constante nele. A primazia nesta regra é sempre a maximização do valor carregado, seja o total do seu volume ou seu custo agregado.

Gehring e Bortfeldt (1997) continuam com a definição de cinco restrições básicas, mas facultativas:

a) orientação: um dado volume deve ser condicionado a uma posição restrita. Uma ou duas dimensões do volume não poderão ser orientadas na direção vertical. Esta restrição é representada pela sigla C1;
b) empilhamento: um dado volume não poderá suportar peso sobre si. Nenhum outro volume poderá estar sobre este. Esta restrição é representada pela sigla C2;
c) capacidade de carga do ambiente: o total de peso da carga não poderá exceder o total permitido para aquele ambiente. Esta restrição é representada pela sigla C3;
d) estabilidade do volume: a estabilidade do volume é calculado em razão à base em que ele está apoiado. Esta restrição exige que a pilha não caia após seu total empilhamento. É representada pela sigla C4;
e) balanceamento da carga: a distância entre a coordenada x do centro da carga em relação ao centro do ambiente não poderá exceder um dado valor. A coordenada y deverá ser análoga a anterior. Esta restrição é representada pela sigla C5.

Esta última restrição, não implementada nesta ferramenta, poderia ser estendida ao balanceamento de peso nas extremidades do ambiente. O resultado seria uma carga centralizada (caso não completasse o ambiente) e com peso distribuído.

Uma das abordagens possíveis para a solução deste problema é o uso de algoritmos genéticos.

Na próxima subseção, serão introduzidos alguns conceitos sobre algoritmo genético, suas operações e aplicações, assim como otimização.

2.3 Algoritmo genético aplicado à otimização de processo

Nesta subseção, aborda-se alguns conceitos sobre algoritmo genético, inteligência artificial e seu uso em otimização de processo.

2.3.1 Algoritmo genético (GA)

Segundo Lopes (1999) *apud* Berz (2008, p. 22), “[...] são métodos de busca e otimização baseados no conceito Darwiniano da evolução dos seres vivos e em fundamentos da genética”.

Russel e Norvig (1995, p. 619), “[...] o que é bom para a natureza, é bom também para sistemas artificiais”.

De uma outra forma, Rebello e Hamacher (2000, p. 4) conceituam:

> Algoritmos Genéticos (AG) são métodos de busca e otimização baseados nos princípios de Seleção Natural e Reprodução Genética. Eles empregam um processo adaptativo e paralelo de busca de soluções em problemas complexos (Pacheco, 1999). Os AG se enquadram na classe de métodos heurísticos inteligentes ou Metaheurísticas.

Leclerc e Potvin (1997) *apud* Rebello e Hamacher (2000, p. 4), “[...] dizem que AG são procedimentos de busca aleatória que imitam (em alto nível de abstração) o processo de Evolução Natural.”

Selow, Lopes e Neves (2001, p. 1) afirmam que “algoritmos genéticos têm sido utilizados satisfatoriamente nas últimas décadas para resolver problemas combinatoriais de alto grau de complexidade, em várias áreas da Engenharia (LOPES, 1999) e da Computação”.

Segundo Rebello (2000), o algoritmo genético consiste em uma sequência de passos como segue:

a) criar uma população ou geração inicial de soluções. Cada item desta população é um indivíduo;
b) analisar se a regra de finalização das evoluções está satisfeita. Se sim, selecionar a melhor população (mais evoluída);
c) avaliar cada indivíduo de acordo com o problema;

d) atribuir a cada indivíduo um valor, conhecido como *fitness* (LOPES, 1999), com o qual este terá a sua probabilidade de cruzamento calculada. Este valor expressa sua qualidade com relação ao problema;
e) escolher a operação a ser realizada: cruzamento ou mutação;
f) se cruzamento, escolher dois indivíduos da população os quais gerarão dois novos indivíduos. Se mutação, escolher um indivíduo que será alterado alguma característica gerando um novo indivíduo;
g) voltar ao passo b e repetir os passos agora com a nova população.

O resultado final pode ser entendido pela Figura 7.

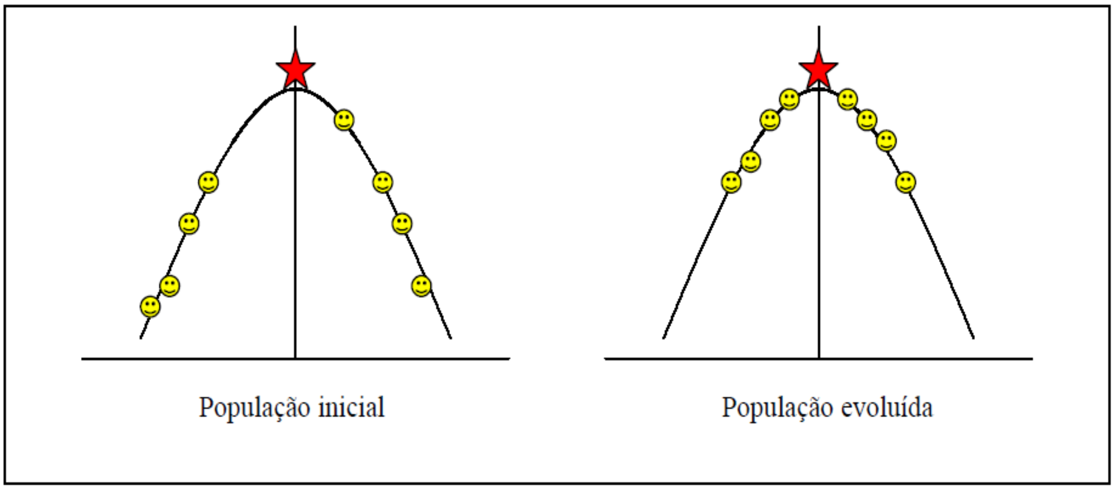


Figura 7: Evolução de uma população de soluções
Fonte: Rebello e Hamacher (2000, p. 4)

Como pode se observar na Figura 7, a população evoluída está mais próxima do ponto ótimo da solução.

2.3.2 Operações em GA

As modificações probabilísticas aplicadas na população visando melhorar a qualidade dos indivíduos resultantes são chamadas de operadores genéticos (LOPES, 1999).

São vários os operadores existentes, que vão favorecer a evolução a uma determinada característica dos indivíduos, dependendo do problema e da representação (REBELLO E HAMACHER, 2000).

Porém, os mais usados são o cruzamento (*crossover*) e a mutação (*mutation*) conforme afirma Lopes (1999, p. 2):

> A recombinação toma dois indivíduos e combina partes de ambos para formar dois novos descendentes. A mutação, por outro lado, atua em um indivíduo em particular, muda aleatoriamente um bit de sua composição. A recombinação atua como busca local, enquanto que a mutação realiza uma busca global do espaço de busca.

2.3.3 Função de adequação do indivíduo

Segundo Lopes (1999), cada indivíduo da população pode ser uma solução do problema e que a seleção dos melhores é baseada em uma medida qualitativa, a qual é calculada pela função de *fitness*.

Portanto, esta função torna-se o ponto crítico do algoritmo, uma vez que é ela que evolui um indivíduo em detrimento de outro.

Este procedimento imita a evolução natural que guia a evolução das espécies.

2.3.4 Algumas aplicações usando a abordagem de GA

São inúmeras as aplicações já desenvolvidas usando a abordagem do algoritmo genético. Dentre elas pode-se destacar:

a) um sistema para o problema do caixeiro viajante utilizando algoritmos genéticos, Berz (2008);
b) uma proposta de algoritmo genético de duas fases para roteamento de veículos, Rebello e Hamacher (2000);
c) programação genética na detecção de eventos epilépticos: Resultados preliminares, Marchesi, Stelle e Lopes (1997);
d) algoritmo genético aplicado à predição da estrutura de proteínas utilizando o modelo 3D-HP *side chain*, Benítez e Lopes (2007).

Na subseção seguinte, será abordado o algoritmo específico deste trabalho.

2.4 Abordagem básica do algoritmo

Para formular a abordagem básica, alguns conceitos devem ser conhecidos.

Em primeira instância, assume-se que as regras de viabilidade F1, F2 e F3 são respeitadas.

Apenas um volume do arranjo (pilha), a qual será tratada como volume base, estará apoiado sobre o chão do ambiente. Todos os outros volumes colocados sobre outro volume deverá estar totalmente apoiado sobre o mesmo, não podendo restar pedaços do volume superior para fora da base. Este tipo de arranjo é chamado de torre. O volume base da torre é chamado de base da torre.

Na Figura 8, o primeiro empilhamento respeita estas normas, tornando-se uma torre viável. As duas outras não.

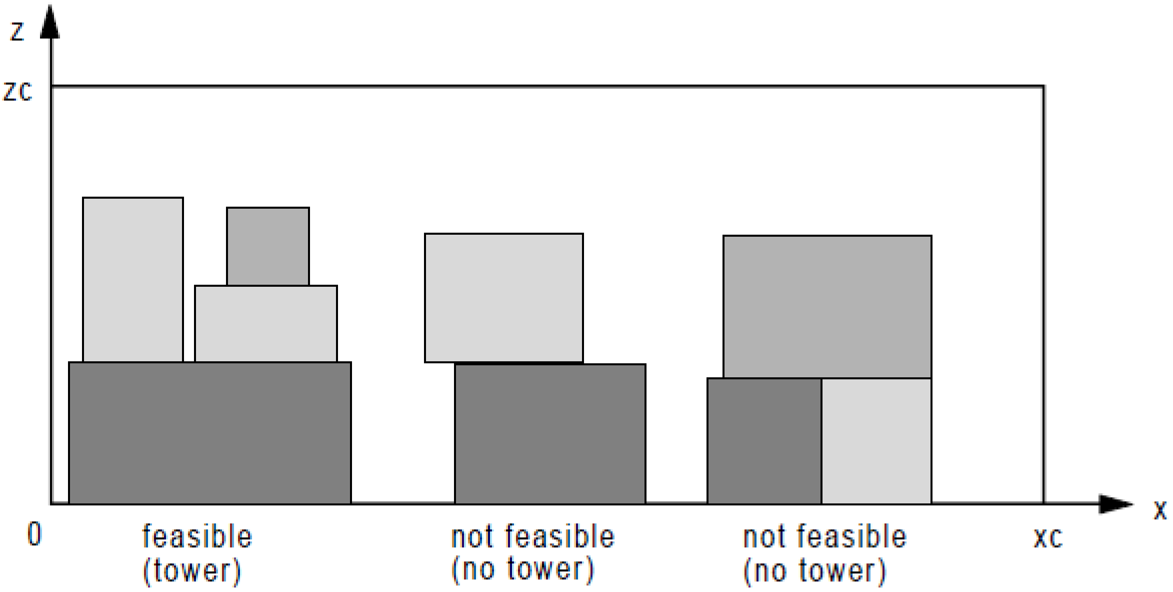


Figura 8: Exemplo de montagem de torre
Fonte: Gehring e Bortfeldt (1997, p. 3)

Usando estes conceitos básicos, Gehring e Bortfeldt (1997) formulam como segue:

a) no primeiro passo, os volumes *nb* são arranjados em conjunto de torres.

   Para isso é usado um teste com algoritmo guloso. Segundo Szwarcfiter (1984) *apud* Sorroche e Lopes (2003, p. 3):

> a denominação algoritmo guloso provém do fato de que a cada passo procura-se incorporar à solução até então construída, a melhor porção possível, compatível com algum critério especificado.

b) no segundo passo, o chão do ambiente é coberto pelas bases das torres geradas. O resultado do problema bidimensional é resolvido com um algoritmo genético, o qual maximiza o valor total carregado. Este passo é finalizado combinando-se a geração do conjunto de torres e a melhor solução para a maior cobertura do chão do ambiente;
c) ambos os passos anteriores são repetidos várias vezes. Cada repetição inclui a geração de novas variações de conjuntos de torres e cobertura do chão com as torres novas. No fim, a melhor geração é usada como solução do problema.

Gehring e Bortfeldt (1997) afirmam que esta abordagem e especialmente a escolha do conceito de torres assegura a viabilidade da solução gerada.

Na próxima subseção, alguns conceitos sobre as tecnologias aplicadas neste trabalho serão esclarecidas.

2.5 Tecnologias aplicadas

Nesta subseção serão abordados alguns conceitos sobre cada tecnologia usada.

2.5.1 Processo de *software*

Segundo Pressman (2002), um *software* não é um produto manufaturado no sentido clássico da palavra e sim é desenvolvido ou passa por um processo de engenharia. Apesar da semelhança entre a fabricação de *hardware* e *software*, as duas atividades são fundamentalmente diferentes. Um projeto de software não pode ser gerido como se fossem projetos de fabricação.

O modelo de processo escolhido para este projeto é o sequencial linear, também conhecido como ciclo de vida clássico ou modelo em cascata.

Pressman (2002) o define como um modelo de abordagem sistemática sequencial para o desenvolvimento de *software* que começa no nível de sistema e progride através da análise, projeto, codificação, teste e manutenção.

Em nota na página, Pressman (2002, p. 26) observa:

> Apesar do modelo original em cascata proposto por Winston Royce prever "ciclos de realimentação", a grande maioria das organizações que aplica esse modelo de processo o trata como se fosse estritamente linear.

A Figura 9 exemplifica o processo.

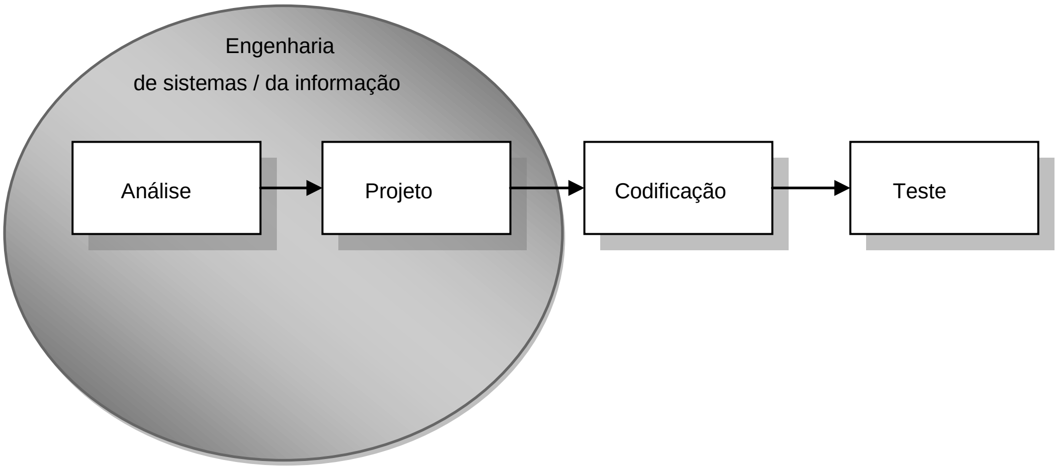


Figura 9: Representação do modelo linear sequencial
Fonte: Adaptado de Pressman (2002)

2.5.2 Linguagem de modelagem

Em uma breve descrição, Larman (2004, p. 34) traduz: "A Linguagem de Modelagem Unificada (UML) é uma linguagem para especificar, visualizar, construir e documentar os artefatos de sistemas de *software*, bem como para modelar negócios e outros sistemas que não sejam de *software*."

Segundo Medeiros (2004), a *Unified Modeling Language*, idealizada em 1995 por Ivar Jacobson, Grady Booch e Jim Rumbaugh auxilia na modelagem de sistemas, mostrando casos e processos de maneira visual e interativa. A partir da versão 1.3, a UML passou a ser mantida pelo OMG estando hoje na versão 2.2 (OMG).

Medeiros (2004, p. 9) afirma:

> A finalidade da UML é proporcionar um padrão para a preparação de planos de arquitetura de projetos de sistemas, incluindo aspectos conceituais, como processos de negócios e funções de sistema, além de itens concretos, como as classes escritas em determinada linguagem de programação, esquemas de banco de dados e componentes de software reutilizáveis.

A partir do resultado da análise, foi feito os casos de uso e diagramações pertinentes. Não foram usados todos os treze diagramas previstos na UML, pois como disse Medeiros (2004, p. 10), "[...] como foi dito (e bem dito) por Ivar Jacobson, 20% da UML resolve cerca de 80% dos problemas do dia a dia".

2.5.3 Plataforma

Neste trabalho foi escolhida a linguagem Java, versão 1.6.0_13 para a implementação do algoritmo. Esta escolha se dá ao fato da linguagem ser, além de gratuita, multiplataforma (DEITEL, 2001, p. 67).

Com isso, fica garantido que o *software* resultante deste trabalho possa ser utilizado em qualquer ambiente computacional livre de qualquer licença de uso, no qual possua um *runtime*[6] Java instalado.

Como ambiente de programação optou-se pela IDE Netbeans, versão 6.7.1 duplamente licenciado sobre a *Common Development and Distribution License* (CDDL) e a GNU *General Public License* versão 2, conforme figura abaixo.

A Figura 10 ratifica estas informações contidas no item "Sobre" do menu "Ajuda" da IDE.

---

[6] http://java.sun.com/javase/6/webnotes/install/jre/README

O IDE NetBeans e a plataforma NetBeans são baseados no software do netbeans.org, o qual foi licenciado duplamente sobre a Common Development and Distribution License (CDDL) e a GNU General Public License versão 2 com exceção do Classpath. Para maiores informações, por favor visite www.netbeans.org.

Versão do produto: NetBeans IDE 6.7.1 (Build 200907230233)
Java: 1.6.0_13; Java HotSpot(TM) Client VM 11.3-b02
Sistema: Windows XP versão 5.1 executando em x86; Cp1252; pt_BR (nb)
Diretório do usuário: C:\Documents and Settings\alencar\.netbeans\6.7

Figura 10: Tela Sobre do Netbeans
Fonte: Autor

Para leitura e interpretação do arquivo XML, utilizou-se a API[7] DOM (*Document Object Model*)[8] que carrega o arquivo inteiro na memória e monta suas partes em formato de árvore, facilitando a localização do seu conteúdo.

O modelo conceitual do uso do DOM está exemplificado na figura seguinte.

---

[7] Conjunto de rotinas e padrões prontas e sua documentação
[8] http://java.sun.com/j2se/1.4.2/docs/api/org/w3c/dom/package-summary.html

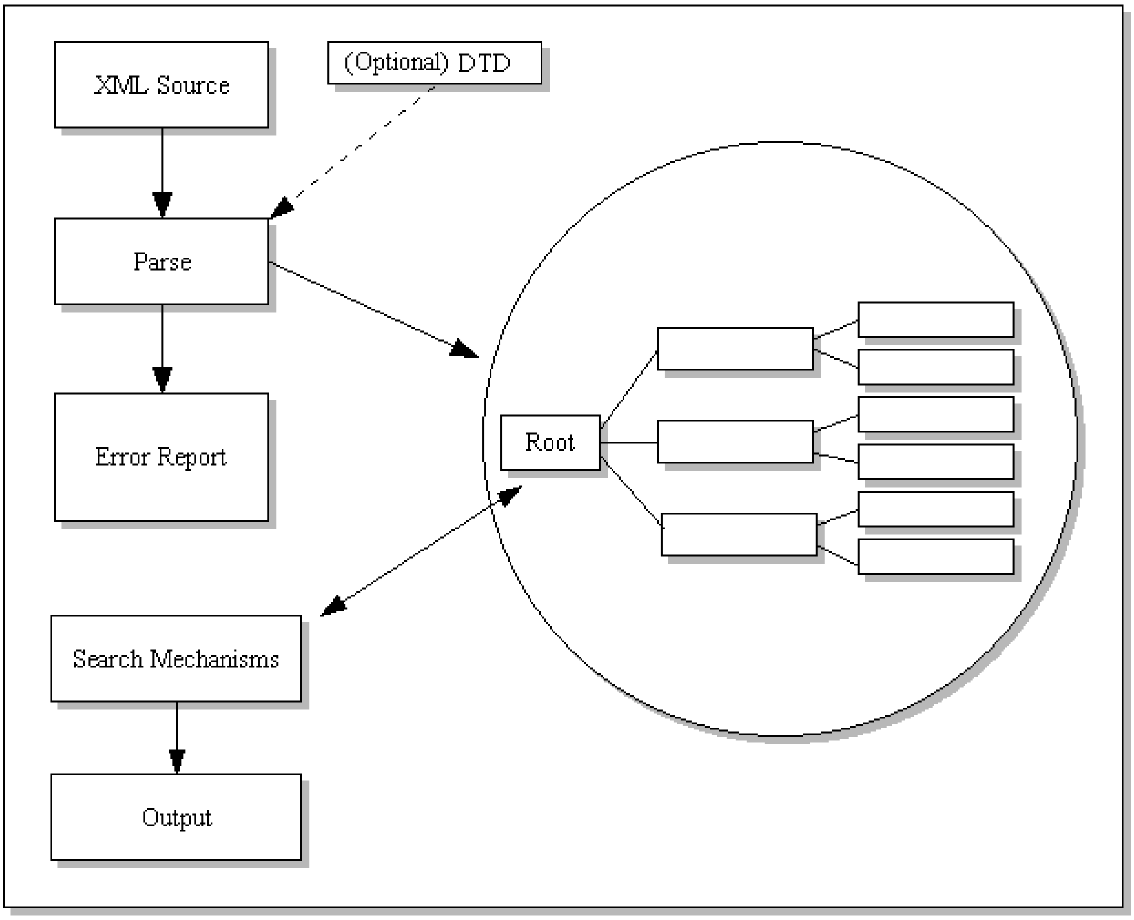


Figura 11: Modelo conceitual do uso de DOM
Fonte: Federizzi (2009)

Conforme pode se ver na Figura 11, o conteúdo do XML é lido e transformado em uma árvore de objetos com base na *tag* root (raiz) do arquivo.

2.5.4 Linguagem para representação estruturada de dados

Para a troca de informações entre o programa resultante deste trabalho e os demais (ERPs, visualizador) foram definidos leiautes[9] usando o padrão XML 1.0, que é “uma linguagem baseada em marcadores, como o HTML, que serve para descrever estruturas de dados” (SAMPAIO, 2006, p. 46).

---

[9] Esboço mostrando a distribuição física e tamanhos de elementos

Cantù (2003, p. 676) descreve o XML como

> [...] uma linguagem de marcação, o que significa que ela usa símbolos para descrever seu próprio conteúdo – neste caso, tags consistindo em texto definido de maneira especial, incluído entre os sinais de menor e maior. Ela se chama extensível porque permite tags (marcas) livres – em contraste, por exemplo, com HTML, que tem tags predefinidas.

Com a finalidade de criticar o arquivo de entrada de dados, para que este contenha a estrutura definida, desenvolveu-se um XML Schema (extensão xsd) que, segundo Sampaio (2006, p. 52), “[...]. É a maneira mais moderna de se definir estruturas de documentos XML”.

Com este XML Schema o programa valida o arquivo de entrada antes de importá-lo, evitando o processamento inócuo de dados inconsistentes.

2.5.5 Testes

Conforme Gladcheff, Zuffi e Silva (2001, p. 4):

> Um Produto de Software é definido pela norma ISO/IEC 9126-1 [ISO9126-1 1997] como "uma entidade de software disponível para liberação a um usuário" e, Qualidade de Software é definida como "a totalidade das características de um produto de software que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas". Em geral, as necessidades explícitas são expressas na definição de requisitos propostos pelo produtor e as necessidades implícitas são aquelas que podem não estar expressas nos documentos do produtor, mas que são necessárias ao usuário.

Segundo Bartié (2002) *apud* Benitti e Zimmermann (2005), os testes “têm por objetivo identificar o maior número possível de erros tanto nos componentes isolados quanto na solução tecnológica como um todo”.

Uma das abordagens de teste é o teste caixa-branca, onde Pressman (2002, p. 435) diz:

> Um teste *caixa-branca* de software é baseado num exame rigoroso do detalhe procedimental. Caminhos lógicos internos ao software são testados, definindo casos de testes que exercitam conjuntos específicos de condições e/ou ciclos. O “estado do programa” pode ser examinado em vários pontos para determinar se o estado esperado ou enunciado corresponde ao estado real.

Segundo Pressman (2002, p. 436), com este teste, também conhecido por *teste caixa de vidro*, o engenheiro de *software* pode derivar casos de teste que

a) garantam que todos os caminhos independentes de um módulo tenham sido exercitados pelo menos uma vez;

b) exercitam todas as decisões lógicas em seus lados verdadeiros e falsos;

c) executam todos os ciclos nos seus limites e dentro de seus intervalos operacionais;

d) exercitam as estruturas de dados internas para garantir sua validade.

Como ferramenta para execução dos testes necessários, foi usada a biblioteca JUnit[10] versão 4.5 a qual vem junto na instalação do Netbeans 6.7.1.

Em seu trabalho, Silva e Teixeira (2008) demonstram a arquitetura completa da biblioteca conforme ilustrado na figura seguinte.

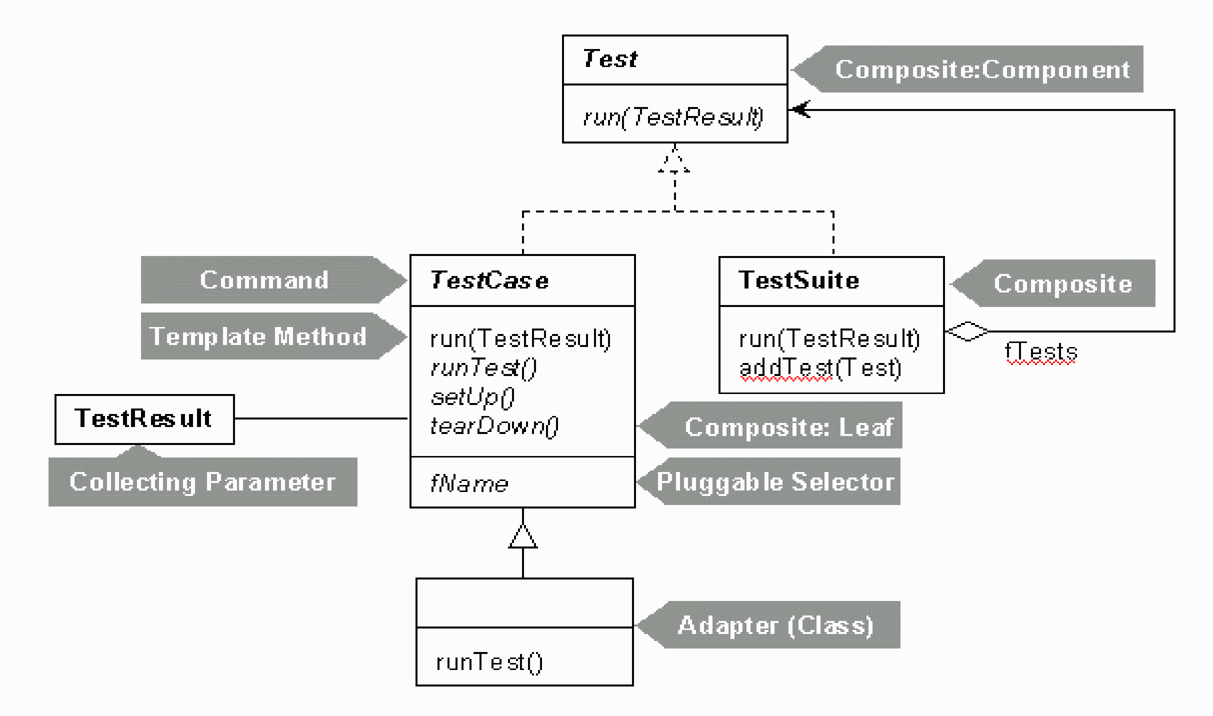


Figura 12: Arquitetura completa do JUnit
Fonte: Silva e Teixeira (2008, p. 4)

Conforme Biasi e Becker (2006, p. 3),

> Um *driver* JUnit é constituído por uma classe principal, cujos métodos representam os casos de teste. Adicionalmente, os métodos *setup* e *teardown* permitem especificar pré e pós condições comuns a todos os casos de teste. A execução de cada caso de teste no JUnit resulta na ativação dos seguintes métodos: (1) *setup*; (2) método correspondente ao caso de teste; e (3) *teardown*. O JUnit permite o uso de diferentes assertivas para comparar os testes, tais como *assertTrue*, *assertEquals e assertFalse.*Um teste no JUnit pode apresentar como resultado três valores: *pass*, *fail* ou *error*.

---

[10] http://www.junit.org/

Com o uso deste *driver*, é possível de ser testado cada método de uma classe, em separado.

Os testes consistem em executar o método definindo-se o resultado esperado. Em qualquer parte do teste pode-se incluir assertivas de comparação que deveriam ou não acontecer.

2.5.6 Avaliação de desempenho

Conforme Jain (1991), avaliação de desempenho é necessária para cada etapa no ciclo de vida de um sistema de computador. Como exemplo, cita que para um administrador de sistema, esta avaliação se faz necessária para decidir por qual sistema optar .

Para que comparações sejam feitas, define-se uma carga de trabalho igual aos *softwares* comparados, os quais são submetidos a um teste de carga.

Todas as execuções são feitas sob as mesmas condições de ambiente, tais como processador, memória, sistema operacional, etc.

Jain (1991) conceitua que para um teste de carga pode ser usadas uma carga real ou sintética. Uma carga real geralmente não é apropriada para teste, pois não pode ser repetida, pois é parte de uma operação normal. Ao invés disso, uma carga sintética, a qual tem características similares às reais, pode ser aplicada repetidamente de maneira controlada.

Na próxima seção, serão abordados os detalhes do desenvolvimento da ferramenta CargaFacil, *software* resultado deste trabalho.

3 CARGAFACIL – OTIMIZANDO CARREGAMENTOS

Nesta seção, serão descritos todos os passos desde a análise até a fase de testes da ferramenta.

3.1 Descrição da ferramenta

O *software* de otimização de carregamento de contentores e semelhantes propõe-se a auxiliar na montagem de uma dada carga de volumes quadrados ou retangulares, dos mais variados tamanhos e pesos, dentro de um ambiente também quadrado ou retangular, com dimensões definidas, capacidade de peso suportado especificado ou não.

A entrada de dados poderá ser feita através de digitação na tela do programa ou através de importação de arquivo XML com a estrutura aceita pelo XML Schema.

Os dados necessários para a operação são:

a) Identificação
  - Número do lote para identificação do relatório.
  - Data do embarque para controle da expedição.
  - Observações se forem necessário.

b) Ambiente de carga
  - Código do ambiente.
  - Descrição do ambiente.
  - Comprimento útil.
  - Largura útil.
  - Altura útil.

c) Pedido
  - Número do pedido.
  - Nome do cliente.

d) Itens do pedido
    - Referência do produto.
    - Descrição do produto.
    - Quantidade de caixas.
    - Peso unitário.
    - Valor total dos produtos na caixa.
    - Comprimento da caixa.
    - Largura da caixa.
    - Altura da caixa.
    - Código da orientação.
    - Quantidade máxima da pilha.
    - Peso máximo permitido sobre a caixa.
    - Se este produto pode ser empilhado.

A definição do leiaute do arquivo de entrada assim como as regras para validação dos dados está definida no XML Schema do apêndice A.

No apêndice B, exemplifica-se um lote de carga formatado no padrão definido neste trabalho.

O resultado final do processo será um relatório de carregamento em PDF, com a sequência do carregamento, iniciado do fundo para frente e da esquerda para direita.

Para visualização deste relatório, o usuário deverá ter em seu computador um leitor de PDF a seu critério.

Também será gerado um arquivo no formato XML com as mesmas informações de posicionamento do carregamento, como a entrada do *software* de visualização.

Não se faz necessário a execução do programa visualizador. Portanto, este trabalho limita-se somente a gerar as informações para o mesmo.

3.2 Desenvolvimento

Como mencionado anteriormente, a ferramenta CargaFacil foi desenvolvida usando um processo sequencial.

As seções seguintes irão apresentar as tarefas e os artefatos produzidos em cada etapa.

3.2.1 Análise

Segundo Pressman (2002, p. 266), “[...] resultam na especificação das características operacionais do software (função, dados e comportamento), indicam a *interface* do software com outros elementos do sistema e estabelecem restrições que o software deve satisfazer”.

Como ferramentas de análise, foram utilizados o Caso de Uso Descritivo, e o Diagrama de Caso de Uso.

A seguir será detalhado cada um destes artefatos.

3.2.1.1 Caso de Uso Descritivo

De acordo com Larman (2004), casos de uso são artefatos textuais redigidos e não desenhados. Contudo, a UML define um diagrama de caso de uso para ilustrar os nomes dos casos de uso e seus atores.

Larman (2004, p. 71) ainda afirma:

> [...] eles não descrevem o funcionamento interno de um sistema [...] é descrito como tendo responsabilidades, [...]. Ao definir responsabilidades do sistema com casos de uso caixa-preta, é possível especificar o que o sistema deve fazer (os requisitos funcionais) sem decidir como ele o fará (o projeto).

Para Medeiros (2004, p. 36), fica clara a importância do uso deste artefato quando afirma:

> O Caso de Uso é a parte mais importante da construção de software orientado a objetos utilizando a UML. Os Casos de Uso são, talvez, o único instrumento que acompanha um software do seu início até a sua conclusão.

UC001 - Caso de uso Entrada de dados

Descrição: Permite a entrada de dados pelo usuário, através de tela ou importação.

Atores: Usuário, Sistema, Validador.

Escopo: Receber, validar e armazenar dados.

Cenários principais:

1. O usuário escolhe se vai digitar ou importar os dados. Usuário
   Caso escolha importar, remete para o cenário 7. Sistema
2. Informa os dados do lote. Usuário
3. Informa os dados do ambiente. Usuário
4. Informa os dados dos pedidos e seus itens. Usuário
5. Escolhe a opção de salvar os dados digitados. Usuário
6. Os dados são salvos no formato XML.
   Remete ao cenário 10. Sistema
7. Solicita o nome e o caminho do arquivo. Sistema
8. Informa onde e qual arquivo importar. Usuário
9. Importa os dados do arquivo. Sistema
10. Escolhe a opção de processar o carregamento. Usuário
11. Processa os dados. Sistema
12. Executa o leitor de PDF definido pelo usuário. Sistema
13. Encerra o processo. Remete ao cenário 1. Sistema

Cenários alternativos:

2.1 Não informa o número do lote.
- 2.1.1 Alerta o usuário da falta da informação. Sistema
- 2.1.2 Remete ao cenário 2. Sistema

2.2 Não informa a data de embarque.
- 2.2.1 Alerta o usuário da importância da informação. Sistema

3.1 Não informa o código do contentor.
- 3.1.1 Alerta o usuário da importância da informação. Sistema

3.2 Não informa a descrição do contentor ou informa com menos de 5 caracteres.
- 3.2.1 Alerta o usuário da falta ou tamanho inválido da informação. Sistema

3.2.2 Remete ao cenário 3. Sistema

3.3 Não informa o comprimento do contentor.
3.3.1 Alerta o usuário da falta da informação. Sistema
3.3.2 Remete ao cenário 3. Sistema

3.4 Não informa a largura do contentor.
3.4.1 Alerta o usuário da falta da informação. Sistema
3.4.2 Remete ao cenário 3. Sistema

3.5 Não informa a altura do contentor.
3.5.1 Alerta o usuário da falta da informação. Sistema
3.5.2 Remete ao cenário 3. Sistema

3.6 Não informa a capacidade de peso suportada pelo contentor.
3.6.1 Alerta o usuário da importância da informação. Sistema

4.1 Não informa o número do pedido.
4.1.1 Alerta o usuário da falta da informação. Sistema
4.1.2 Remete ao cenário 4. Sistema

4.2 Não informa o nome do cliente.
4.2.1 Alerta o usuário da falta da informação. Sistema
4.2.2 Remete ao cenário 4. Sistema

4.3 Não informa a referência do produto.
4.3.1 Alerta o usuário da falta da informação. Sistema
4.3.2 Remete ao cenário 4. Sistema

4.4 Não informa a quantidade de caixas desta referência.
4.4.1 Alerta o usuário da falta da informação. Sistema
4.4.2 Remete ao cenário 4. Sistema
4.5 Não informa o comprimento da caixa.
4.5.1 Alerta o usuário da falta da informação. Sistema
4.5.2 Remete ao cenário 4. Sistema
4.6 Não informa a largura da caixa.
4.6.1 Alerta o usuário da falta da informação. Sistema
4.6.2 Remete ao cenário 4. Sistema

4.7 Não informa a altura da caixa.
4.7.1 Alerta o usuário da falta da informação. Sistema
4.7.2 Remete ao cenário 4. Sistema

4.8 Não informa o código do tipo de orientação da caixa, ou informa fora da faixa permitida [1, 2, 4, 6].
4.8.1 Alerta o usuário da falta ou validade da informação. Sistema
4.8.2 Remete ao cenário 4. Sistema

5.1 Não informa o caminho e o nome do arquivo.
5.1.1 Sugere um caminho e um nome. Sistema

6.1 Não consegue gravar o arquivo.
6.1.1 Alerta o usuário do erro encontrado. Sistema
6.1.2 Solicita um novo caminho e nome para o arquivo. Sistema
6.1.3 Remete ao cenário 5. Sistema

8.1 Não informa um nome de arquivo existente.
8.1.1 Alerta o usuário do erro encontrado. Sistema
8.1.2 Remete ao cenário 7. Sistema

9.1 Não consegue ler ou abrir o arquivo.
9.1.1 Alerta o usuário do erro encontrado. Sistema
9.1.2 Remete ao cenário 7. Sistema

9.2 Não valida o leiaute ou as informações do arquivo. Validador
9.2.1 Alerta o usuário do erro encontrado. Sistema
9.2.2 Remete ao cenário 7. Sistema

11.1 Não consegue processar os dados.
11.1.1 Alerta o usuário do erro encontrado. Sistema
11.1.2 Remete ao cenário 7. Sistema

12.1 Não consegue executar o leitor de PDF.
12.1.1 Alerta o usuário do erro encontrado. Sistema
12.1.2 Remete ao cenário 1. Sistema

Quadro 1: Caso de uso descritivo do processo principal
Fonte: autor

3.2.1.2 Diagrama de caso de uso

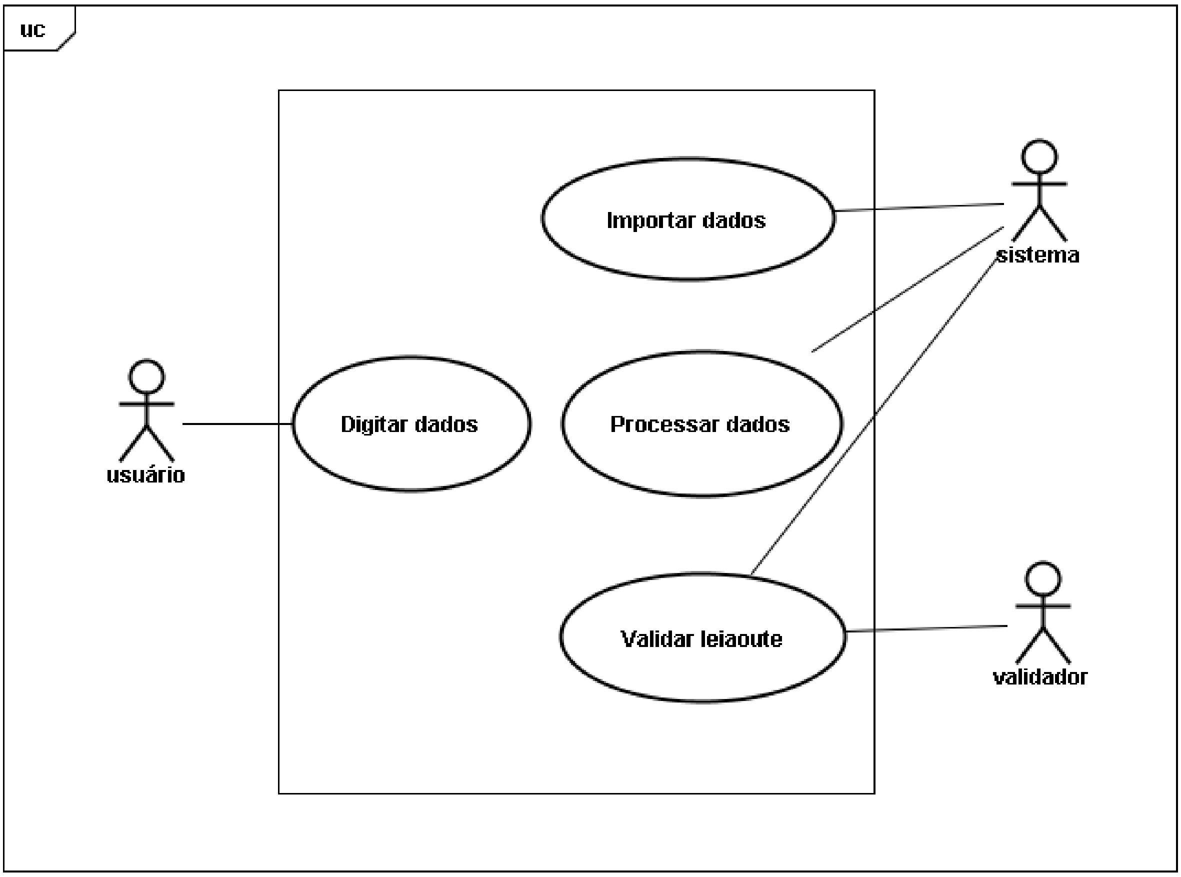
Segundo Larman (2004, p. 92), a UML fornece a notação de diagramas de casos de uso para ilustrar os nomes dos casos e seus atores, bem como os relacionamentos entre si.

Mas, logo em seguida, o autor alerta que estes diagramas não se fazem necessários, pois os casos de uso são documentos de texto.

Afirma Larman (2004, p. 92):

> [...]. Especialistas em modelagem de casos de uso mundialmente reconhecidos, como Anderson, Fowler, Cockburn, entre outros, não se preocupam com os diagramas de casos de usos e seus relacionamentos, concentrando-se, em vez disso, em sua redação. Sem desconsiderar este aviso, um simples diagrama de caso de uso oferece um visual sucinto que ilustra os atores externos e como eles usam o sistema.

Com isso, desenha-se um simples e único diagrama, com vistas a ilustrar sucintamente o processo principal da ferramenta.



Quadro 2: Diagrama do caso de uso principal
Fonte: Autor

3.2.2 Projeto

Conceitua Pressman (2004, p. 329):

> Projeto é uma representação de engenharia de algo que vai ser construído. Ele pode ser delineado para os requisitos do cliente e ao mesmo tempo avaliado quanto à qualidade, com base num conjunto de critérios predefinidos para o "bom" projeto. No contexto da engenharia de software, o projeto focaliza quatro áreas principais de preocupação: dados, arquitetura, interfaces e componentes.

Para especificar estas quatro fases, este trabalho baseou-se na notação da linguagem UML, criando o diagrama de classe, diagrama de sequência e diagrama de atividade.

Medeiros (2004) explica que a UML nos faz pensar em um *software* em um local e codificá-lo em outro, mas deixa claro que esta linguagem não nos indica como deve ser feito.

3.2.2.1 Diagrama de classe

Diagrama de classe representa a estrutura de cada classe do sistema assim como seus relacionamentos.

Conforme Medeiros (2004, p. 117), “um diagrama de classe pode atender a diversos Casos de Uso, e não a um único. Se você tem um diagrama de classes inteiro, atendendo apenas a um Caso de Uso, provavelmente este Caso de Uso está enorme e seus conceitos estão errados.”

Neste trabalho fugiu-se à regra no que tange ao caso de uso mais amplo, pois não envolveu equipe de trabalho e, consequentemente, não se fez necessário quebrar o mesmo em partes menores.

Nas Figuras 13 e 14, o desenho do diagrama de classe está repartido em duas partes para melhor entendimento do texto.

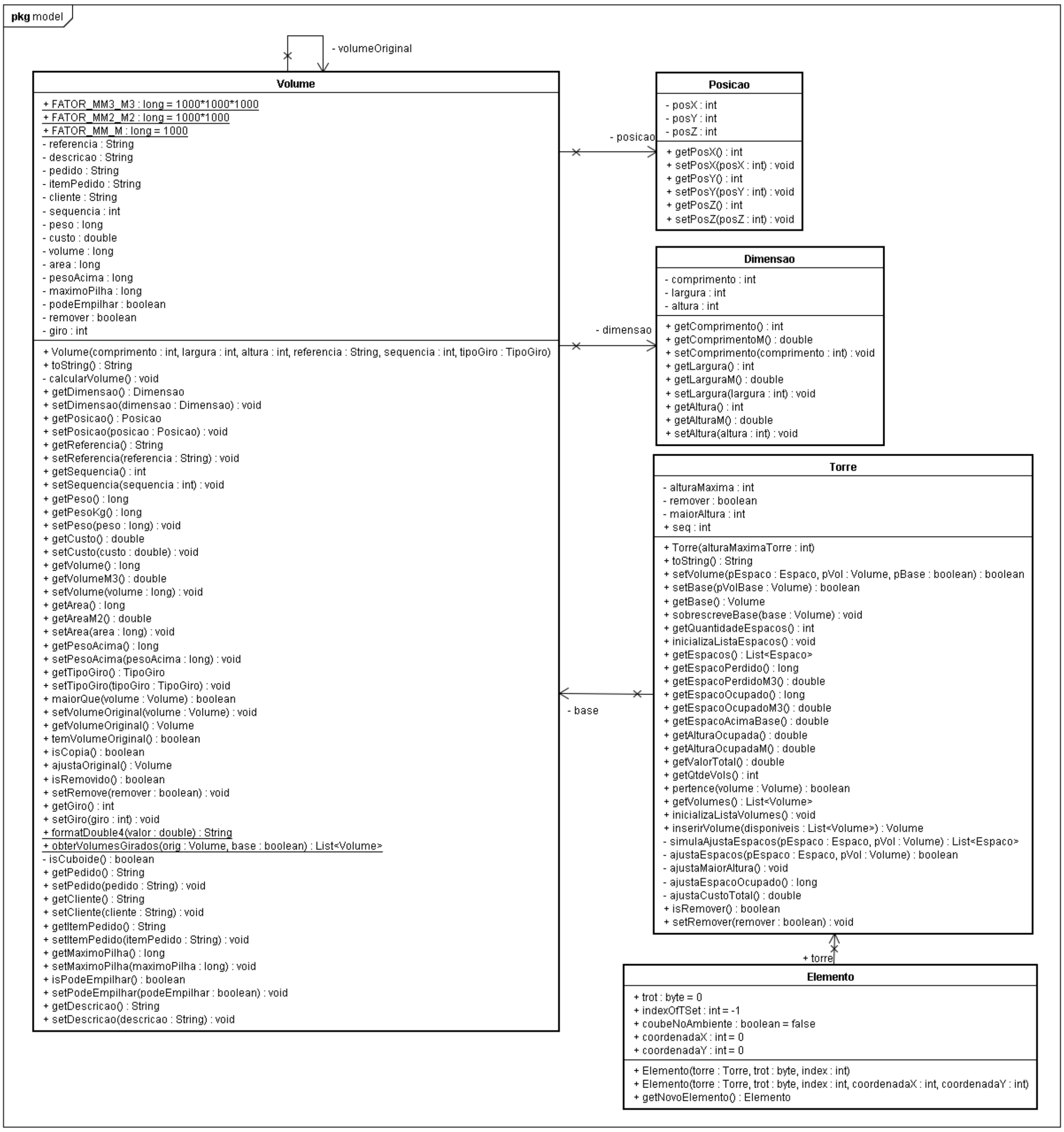


Figura 13: Diagrama de classe parte 1
Fonte: Autor

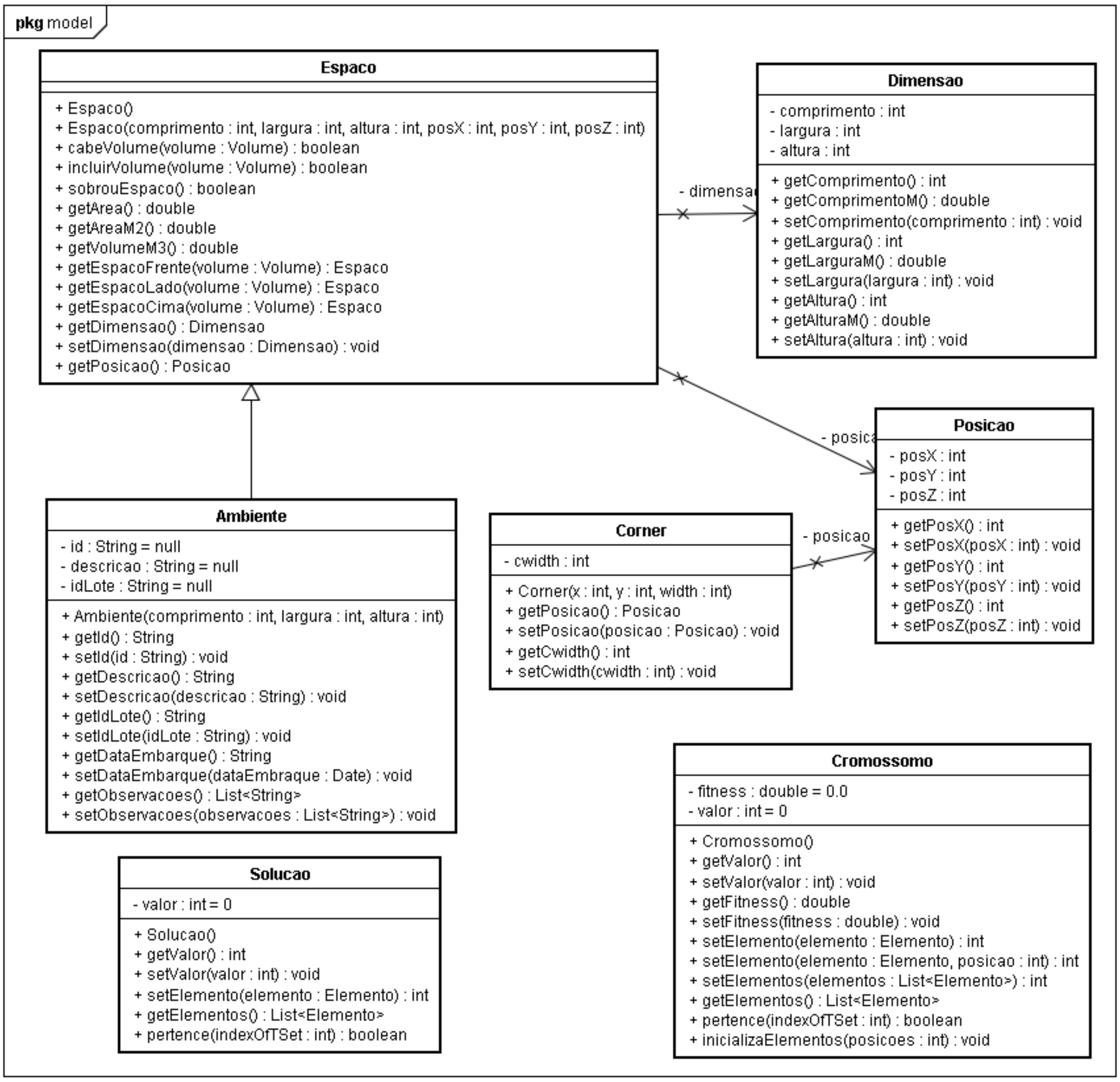


Figura 14: Diagrama de classe parte 2
Fonte: Autor

3.2.2.2 Diagrama de sequência

Dentre os diagramas de interação existentes na UML 2.2, o diagrama de sequência é o mais comumente usado, em detrimento ao diagrama de comunicação e o diagrama de interação – Visão Geral e o *Timing Diagram* (MEDEIROS, 2004).

A razão disso, Medeiros (2004, p.147) explica:

> Este diagrama pode ser usado para mostrar a evolução de uma dada situação em determinado momento do software, mostrar uma dada colaboração entre duas ou mais classes e pode, também, ser usado para mostrar a tradução de um Caso de Uso desde a interação com o usuário até a finalização daquele dado processo.
> Pessoalmente, é neste sentido que vejo um bom significado ao diagrama de seqüência. [...]

Como pode ser visto na Figura 15, o desenho do diagrama de sequência do caso de uso principal.

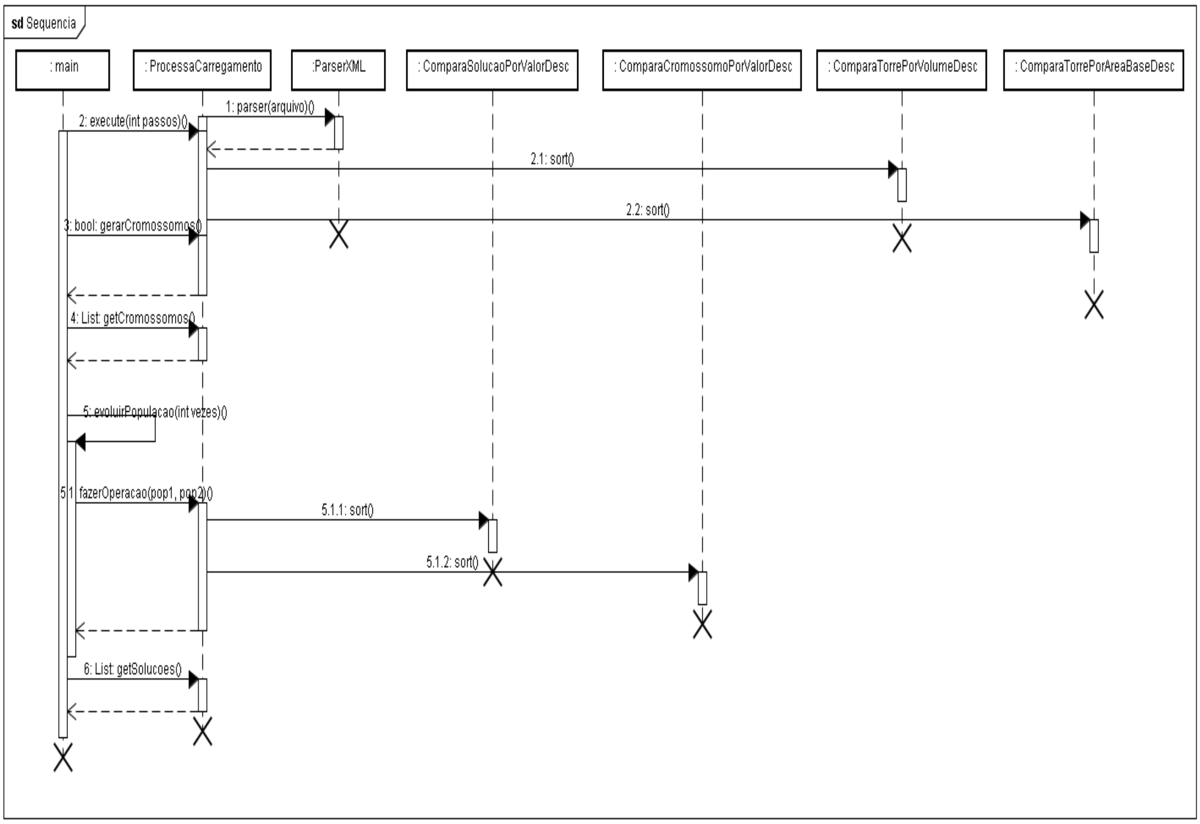


Figura 15: Diagrama de sequência
Fonte: Autor

3.2.2.3 Diagrama de atividade

Medeiros (2004, p. 62) explica que

> o diagrama de Atividades existe para ajudá-lo a criar boas descrições de Casos de Uso, mostrar uma situação por meio de vários Casos de Uso ou ajustar dúvidas surgidas no diagrama de classes ou objetos [...].

Pode-se adotar a abordagem de escrever casos de uso sem a necessidade de desenhar diagrama de atividade, o que é mais comumente usado no mundo real. Porém, torna-se interessante para pessoas que estão acostumadas ao uso do diagrama de fluxo de dados (DFD) da análise estruturada (MEDEIROS, 2004).

A exemplo do diagrama de sequência, abaixo o desenho do diagrama de atividade do cenário principal.

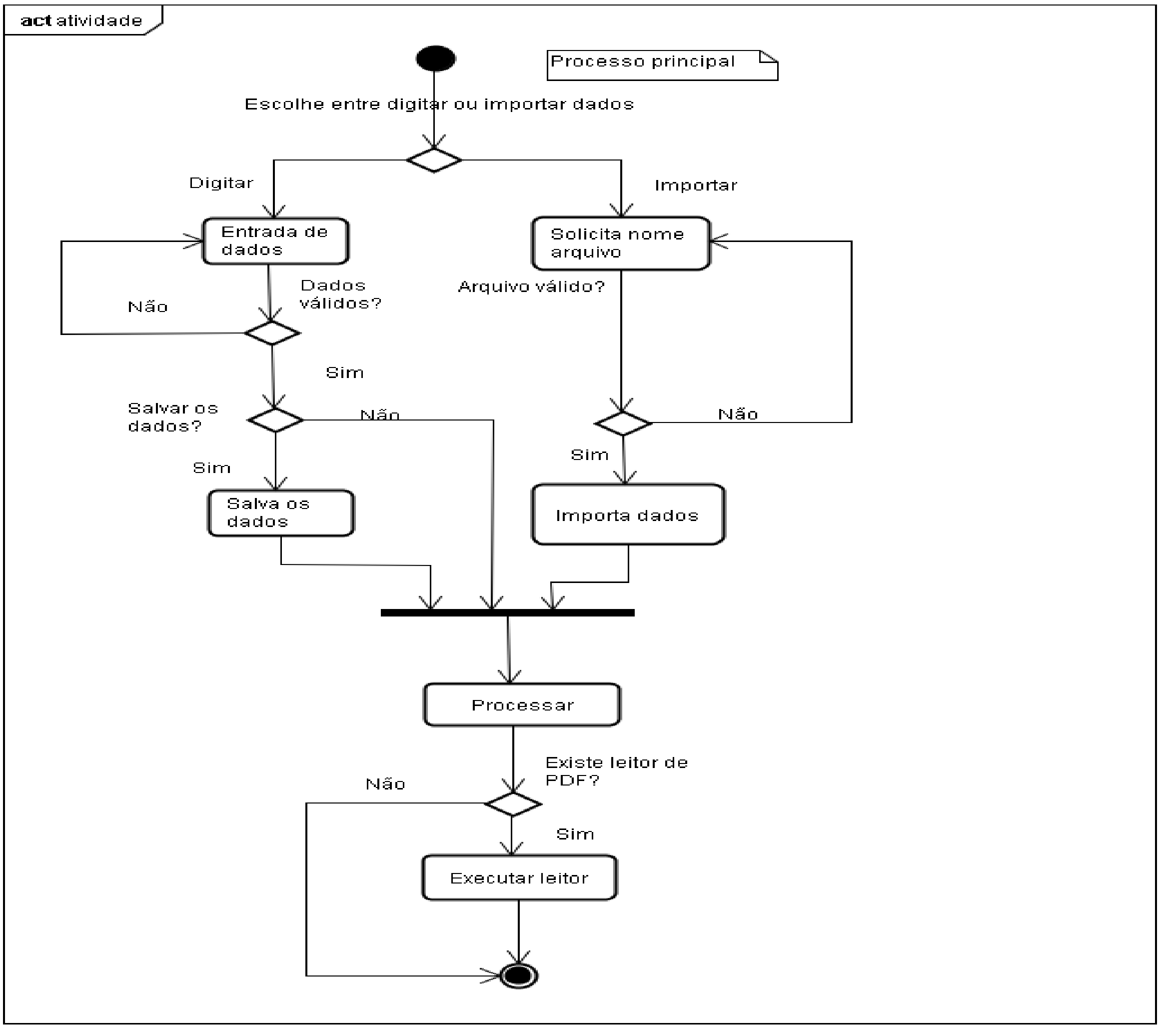


Figura 16: Diagrama de atividade do processo principal
Fonte: Autor

3.2.3 Codificação

Para o desenvolvimento do software, produto fim deste projeto, foi utilizada a linguagem de programação Java, sobre a IDE Netbeans e sem a necessidade de repositório de dados do tipo banco de dados e afins.

Os dados digitados, necessitando salvá-los, serão armazenados no formato XML, nomeados a critério do usuário.

As configurações necessárias para a execução do leitor de PDF e para execução do visualizador do carregamento serão armazenadas em arquivos texto, no formato utilizado pela classe java.util.Properties[11].

Os arquivos de entrada e saída foram projetados no padrão XML e o relatório de carregamento gerado em PDF.

A organização das pastas que compõem a ferramenta é meramente uma sugestão, pois todas as subpastas são configuráveis através do arquivo de configuração cargafacil.properties que se encontra junto ao executável CargaFacil.jar.

A única pasta que deve ser mantida junto ao executável é a lib, onde se encontram as bibliotecas da ferramenta.

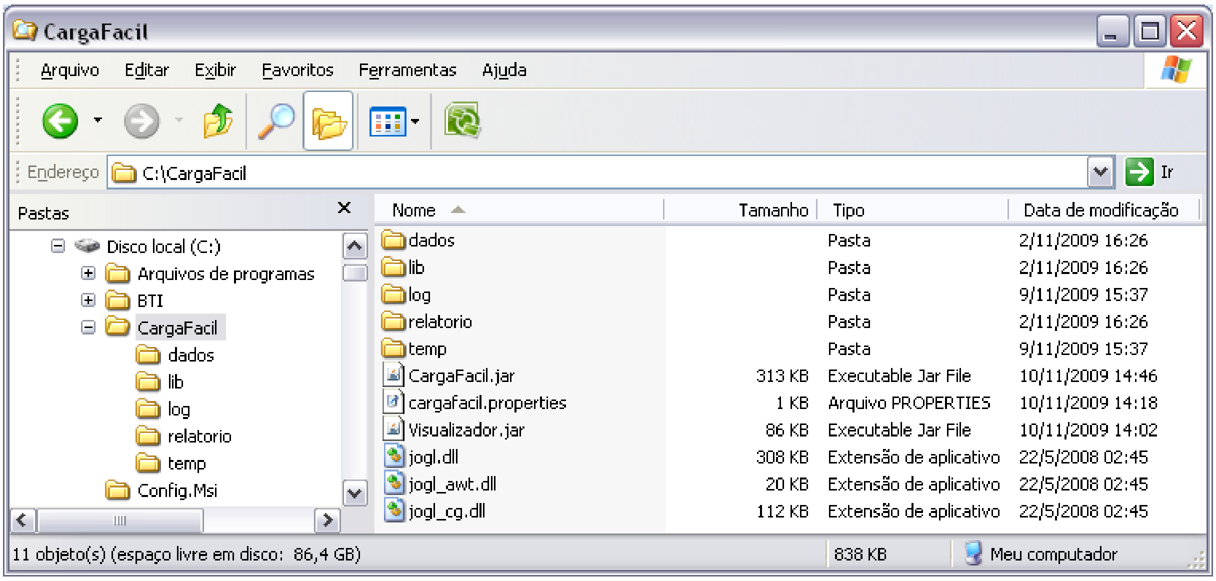
```
pathDados=.\\dados\\
pathLog=.\\log\\
arquivoLog=cargafacil.log
cmdLeitorPDF=C\:\\Arquiv~1\\Adobe\\Reader 9.0\\Reader\\AcroRd32.exe
tamMaximoLogEmMB=10
pathTemp=.\\temp\\
pathRelatorio=.\\relatorio\\
```

Quadro 3: Exemplo de conteúdo do arquivo cargafacil.properties
Fonte: Autor

---

[11] http://java.sun.com/j2se/1.4.2/docs/api/java/util/Properties.html

Na Figura 17, a demonstração da estrutura sugerida conforme quadro anterior.



Endereço: C:\CargaFacil

Nome	Tamanho	Tipo	Data de modificação
dados		Pasta	2/11/2009 16:26
lib		Pasta	2/11/2009 16:26
log		Pasta	9/11/2009 15:37
relatorio		Pasta	2/11/2009 16:26
temp		Pasta	9/11/2009 15:37
CargaFacil.jar	313 KB	Executable Jar File	10/11/2009 14:46
cargafacil.properties	1 KB	Arquivo PROPERTIES	10/11/2009 14:18
Visualizador.jar	86 KB	Executable Jar File	10/11/2009 14:02
jogl.dll	308 KB	Extensão de aplicativo	22/5/2008 02:45
jogl_awt.dll	20 KB	Extensão de aplicativo	22/5/2008 02:45
jogl_cg.dll	112 KB	Extensão de aplicativo	22/5/2008 02:45

11 objeto(s) (espaço livre em disco: 86,4 GB)   838 KB

Figura 17: Estrutura de pastas sugeridas para a ferramenta
Fonte: Autor

3.2.4 Testes de unidade

Os testes usados durante o desenvolvimento foram todos baseados na biblioteca JUnit, integrada a IDE Netbeans.

A cada nova classe e/ou método do sistema, foram executados testes de resultado do tipo verdadeiro ou falso e, em outros casos, testes de valor nulo.

O uso da biblioteca é bastante simples e eficaz. Com a facilitadora tecla de atalho da IDE – Ctrl+Shift+U – uma classe de teste é criada com base nos métodos da classe em que está o foco. Nesta fase, somente o esboço dos métodos de testes são criados.

Feito isso, implementa-se o código de teste assim como define-se o resultado esperado para aquele método.

No Quadro 4 tem-se um fragmento de código de uma unidade de teste.

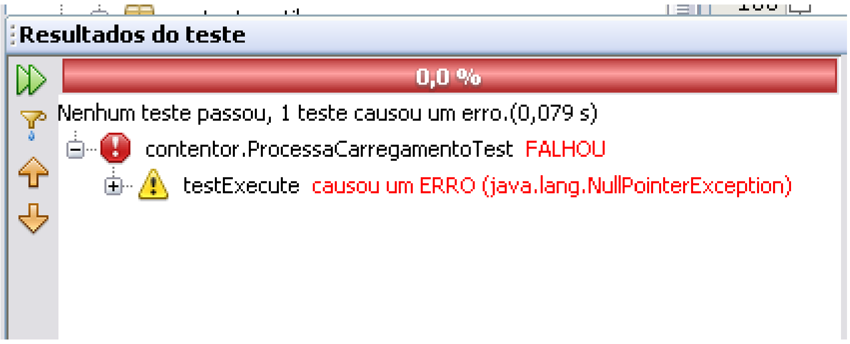
Para executar os testes, usa-se a opção de menu, Executar/Executar arquivo ou a tecla de atalho Shift+F6.

```
/**
* @author alencar
 */
public class ProcessaCarregamentoTest_salvo {
    String xml = "C:/CargaFacil/xmlEntrada/carga.xml";
    String xml2 = "C:/CargaFacil/xmlEntrada/magiclogic.xml";

    public ProcessaCarregamentoTest_salvo() {
    }

    @BeforeClass
    public static void setUpClass() throws Exception {
    }

    @AfterClass
    public static void tearDownClass() throws Exception {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    @Test
    public void geraTorres() {
        ProcessaCarregamento pc = new ProcessaCarregamento(xml2);
        int pops = 50;
        int evol = 500;
        int vols = 163;
        pc.nPop = pops;

        pc.msgs.append("\nGerando torres");
        System.out.println("Gerando torres");
```

Quadro 4: Fragmento de código utilizando a biblioteca JUnit
Fonte: Autor

Como resultado da execução, tem-se uma área especifica na IDE com as informações pertinentes.

A saída do resultado vê-se na Figura 18.



Resultados do teste

0,0 %

Nenhum teste passou, 1 teste causou um erro.(0,079 s)

contentor.ProcessaCarregamentoTest FALHOU

testExecute causou um ERRO (java.lang.NullPointerException)

Figura 18: Resultado da execução do JUnit
Fonte: Autor

3.3 Como usar a ferramenta CargaFacil

Nesta seção será esclarecido como utilizar a ferramenta CargaFacil e as definições dos arquivos de entrada e saída, assim como o relatório de carregamento.

3.3.1 *Interface* com o usuário

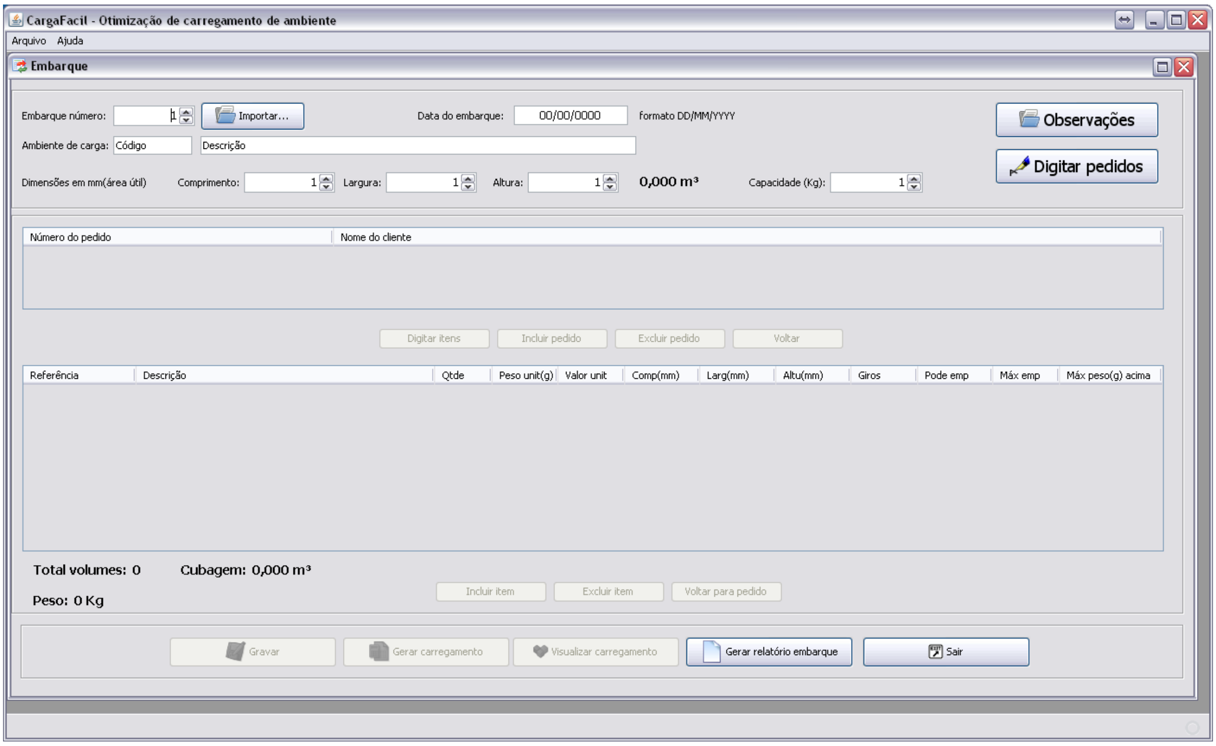
A tela principal do sistema, conforme se observa na Figura 19, será onde as informações são carregadas através do botão “Importar...” ou então digitadas.

As observações de embarque poderão ser incluídas através do botão “Observações”.

Após a carga das informações do embarque, também tratado como lote de carga, o usuário poderá salvar clicando em “Gravar”, calcular o carregamento clicando em “Gerar carregamento” ou ainda visualizar o resultado do item anterior clicando em “Visualizar carregamento”.

Este último botão executará o visualizador do Uriartt (2007) passando o arquivo XML como parâmetro. Neste caso, o programa visualizador, assim como toda a sua configuração necessária, deverá estar presente na mesma pasta do

CargaFacil.



Embarque número: 1
Data do embarque: 00/00/0000
Ambiente de carga: Código Descrição
Comprimento: 1
Largura: 1
Altura: 1
0,000 m³
Capacidade (Kg): 1
Total volumes: 0
Cubagem: 0,000 m³
Peso: 0 Kg

Figura 19: Tela principal do CargaFacil, acessível através do menu Arquivo
Fonte: Autor

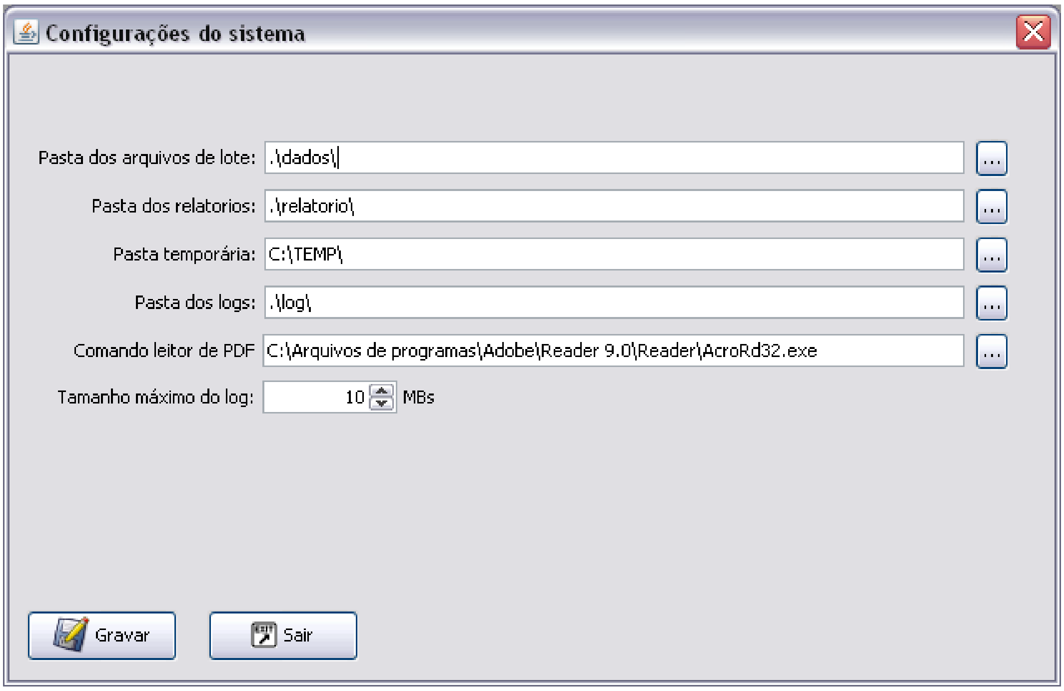
As configurações necessárias ao funcionamento do CargaFacil são informadas através do item de menu “Arquivo/Configurações”.

Esta mesma tela abrirá automaticamente na primeira execução do *software*, garantindo que o usuário informe, ou no mínimo olhe, as observações padrões sugeridas pelo sistema.

Toda a estrutura de pastas que constam nas configurações será criada automaticamente pelo sistema.

A única informação que não será sugerida pelo sistema, é o nome do programa leitor de PDF preferencial do usuário.

Na Figura 20, tem-se a imagem da tela de configuração.



Pasta dos arquivos de lote: .\dados\

Pasta dos relatorios: .\relatorio\

Pasta temporária: C:\TEMP\

Pasta dos logs: .\log\

Comando leitor de PDF C:\Arquivos de programas\Adobe\Reader 9.0\Reader\AcroRd32.exe

Tamanho máximo do log: 10 MBs

Figura 20: Tela de configuração do CargaFail
Fonte: Autor

3.3.2 Arquivo de entrada por importação de dados

Todas as informações pertinentes a um carregamento poderão ser importadas através de arquivo no padrão XML, com base na estrutura definida pelo XML Schema, doravante tratado simplesmente como esquema, desenvolvido neste trabalho.

Para entender o que o esquema exige de estrutura e dados, observa-se os elementos que contêm alguma restrição. Por exemplo:

```
<xs:element name="identificacao">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="idLote" minOccurs="1" maxOccurs="1"/>
      <xs:element ref="dataEmbarque" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="observacoes" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Quadro 5: Fragmento do arquivo carga.xsd
Fonte: Autor

Neste trecho de código apresentado no Quadro 5, tem-se a parte da identificação do lote, onde a *tag* <idLote> tem a restrição minOccurs=”1” e maxOccurs=”1”. Isso nos diz que esta *tag* deve estar presente uma e apenas uma vez no arquivo, dentro do trecho da *tag* <identificacao>.

Estas regras também ficam claras no caso de uso descritivo, onde se pode ler:

```
   2.1 Não informa o número do lote.
      2.1.1 Alerta o usuário da falta da informação. Sistema
      2.1.2 Remete ao cenário 2. Sistema

   2.2 Não informa a data de embarque.
      2.2.1 Alerta o usuário da importância da informação. Sistema
```

Quadro 6: Fragmento do caso de uso descritivo
Fonte: Autor

Nota-se que, no cenário alternativo 2.1, a divergência com o valor esperado é alertado ao usuário e o sistema força o retorno ao mesmo campo da entrada do dado errado. Já no cenário alternativo 2.2, onde a data de embarque tem as regras definidas pelo esquema de no mínimo zero e no máximo uma vez, a divergência somente é alertada, não forçando o usuário a informar uma data válida.

Estas regras definidas no esquema são aplicadas tanto na validação do arquivo importado quanto na entrada de dados digitado, visto que a ferramenta dispõe a opção de salvar estas informações em arquivos do mesmo padrão XML.

3.3.3 Arquivo de saída para integração com o visualizador

As informações geradas pelo processamento da carga, gerando uma otimização do carregamento, são gravadas em arquivo no formato XML proposto pelo visualizador desenvolvido por Uriartt (2007).

A ferramenta disponibiliza uma opção de executar o visualizador, passando como parâmetro o arquivo gerado.

Um pequeno trecho do arquivo de parâmetro passado para o visualizador está representado no Quadro 7.

```
<?xml version="1.0" encoding="UTF-8"?>
<container
  width = "2349"
  height = "2388"
  depth = "5893">
  <box class = "1" id = "1">
    <definition
      width = "470.0"
      height = "290.0"
      depth = "790.0"/>
    <position
      x = "0.0"
      y = "0.0"
      z = "0.0"/>
  </box>
```

Quadro 7: Fragmento do XML passado como parâmetro ao visualizador
Fonte: Autor

3.3.4 Relatório com as diretrizes para o carregamento

De forma simultânea à geração do arquivo acima citado, a ferramenta disponibiliza um relatório em PDF, com as diretrizes do carregamento, volume a volume, do fundo para frente do ambiente, da esquerda para a direita.

Os volumes estarão identificados pelas referências do produto nele contido, conforme informações do lote.

De igual forma, os dados referentes ao lote, ao ambiente e aos pedidos estarão presentes adequadamente.

Na próxima seção, será explicado como foram feitas as avaliações de desempenho e seus respectivos resultados.

4 EXPERIMENTOS

Nesta seção, as técnicas de avaliação de desempenho utilizadas no trabalho serão descritas.

Será descrita a geração das cargas, os resultados, comparativos com *softwares* comerciais e as justificativas necessárias aos resultados.

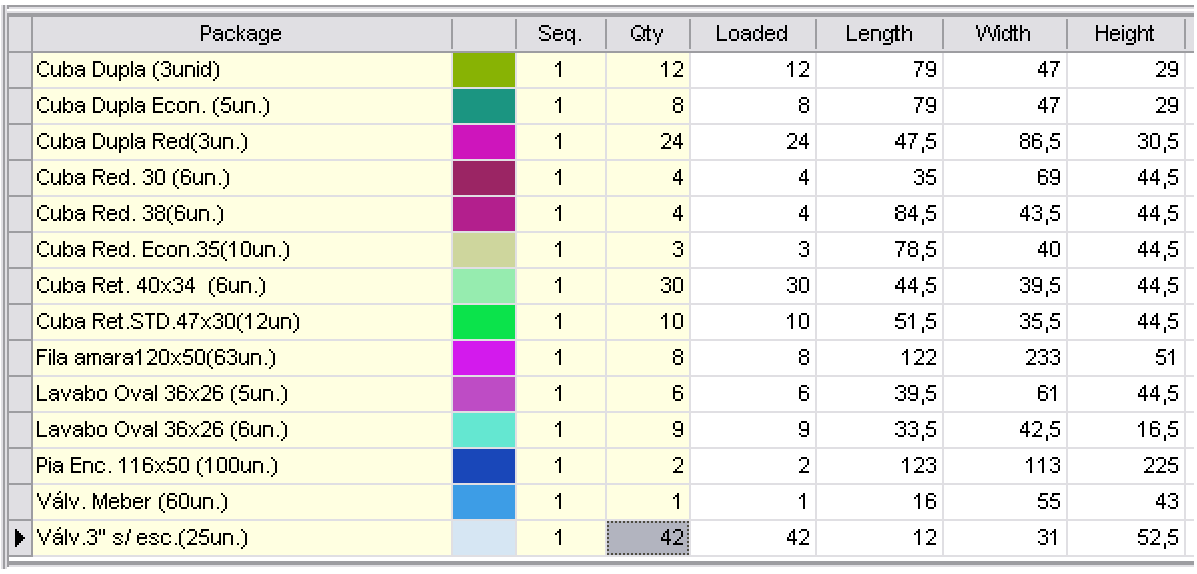
4.1 Simulação de carga

Para os experimentos executados, foram utilizadas duas cargas sintéticas definidas pelos *softwares* Cube-IQ da MagicLogic e CargoWiz, respectivamente para os ambientes (*containers*) de 20’ (vinte pés) e 40’ (quarenta pés).

A configuração de tamanho do *container* de 20’, denominado “20’ (6.1 meters) Dry Container”, usado neste experimento, também definido pelo *software* acima citado, é de 589,3 cm de comprimento, 234,9 cm de largura e 238,8 cm de altura, todas referenciando o espaço útil do ambiente.

Neste ambiente foram simulados 163 volumes a serem carregados.

Na Figura 21, tela do *software* Cube-IQ demonstrando o tamanho da carga.



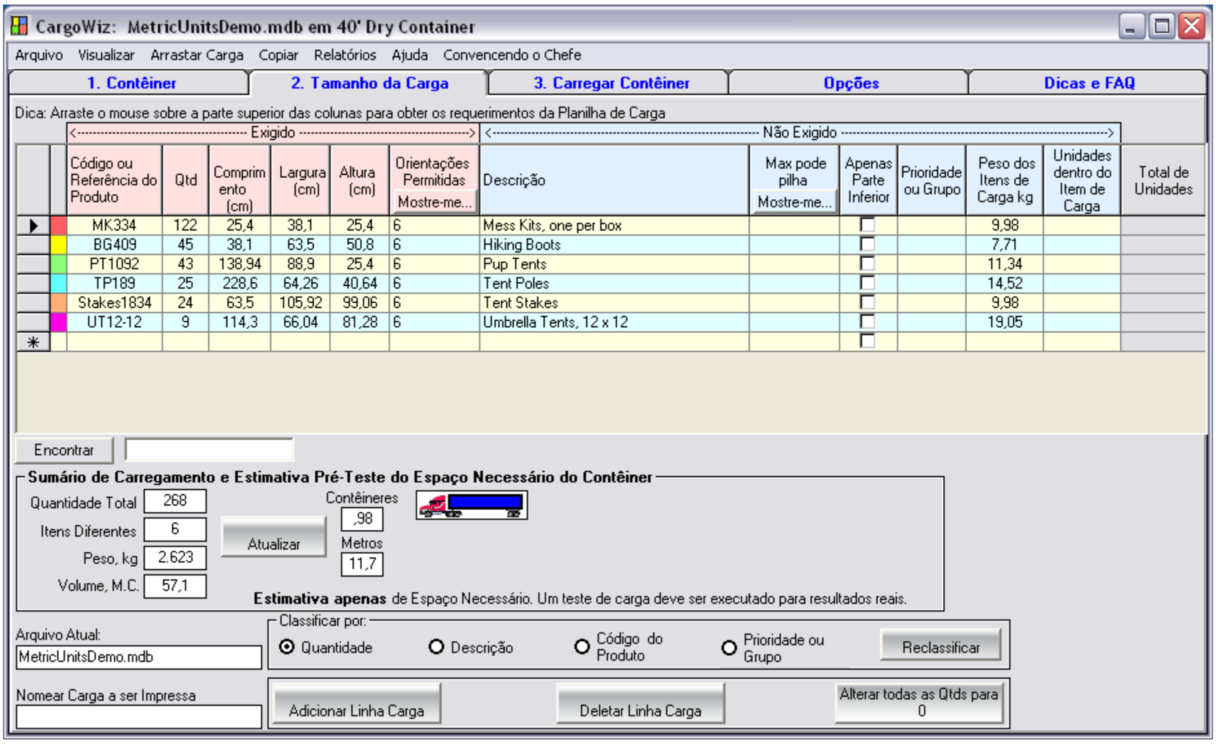
Package		Seq.	Qty	Loaded	Length	Width	Height
Cuba Dupla (3unid)	🟩	1	12	12	79	47	29
Cuba Dupla Econ. (5un.)	🟦	1	8	8	79	47	29
Cuba Dupla Red(3un.)	🟪	1	24	24	47,5	86,5	30,5
Cuba Red. 30 (6un.)	🟪	1	4	4	35	69	44,5
Cuba Red. 38(6un.)	🟪	1	4	4	84,5	43,5	44,5
Cuba Red. Econ.35(10un.)	🟩	1	3	3	78,5	40	44,5
Cuba Ret. 40x34 (6un.)	🟩	1	30	30	44,5	39,5	44,5
Cuba Ret.STD.47x30(12un)	🟩	1	10	10	51,5	35,5	44,5
Fila amara120x50(63un.)	🟪	1	8	8	122	233	51
Lavabo Oval 36x26 (5un.)	🟪	1	6	6	39,5	61	44,5
Lavabo Oval 36x26 (6un.)	🟦	1	9	9	33,5	42,5	16,5
Pia Enc. 116x50 (100un.)	🟦	1	2	2	123	113	225
Válv. Meber (60un.)	🟦	1	1	1	16	55	43
▶ Válv.3" s/ esc.(25un.)	🟦	1	42	42	12	31	52,5

Figura 21: Carga sintética para o container de 20’
Fonte: MagicLogic

Para o *container* de 40’, denominado “40’ (12.2 meters) Dry Container”, o tamanho definido é de 1194 cm de comprimento, 234 cm de largura e 239 cm de altura.

Neste ambiente foram simulados 268 volumes a serem carregados.

Na Figura 22, tela do *software* CargoWiz demonstrando o tamanho da carga.



Dica: Arraste o mouse sobre a parte superior das colunas para obter os requerimentos da Planilha de Carga

Código ou Referência do Produto	Qtd	Comprimento (cm)	Largura (cm)	Altura (cm)	Orientações Permitidas Mostre-me...	Descrição	Max pode pilha Mostre-me...	Apenas Parte Inferior	Prioridade ou Grupo	Peso dos Itens de Carga kg	Unidades dentro do Item de Carga	Total de Unidades
MK334	122	25,4	38,1	25,4	6	Mess Kits, one per box		☐		9,98		
BG409	45	38,1	63,5	50,8	6	Hiking Boots		☐		7,71		
PT1092	43	138,94	88,9	25,4	6	Pup Tents		☐		11,34		
TP189	25	228,6	64,26	40,64	6	Tent Poles		☐		14,52		
Stakes1834	24	63,5	105,92	99,06	6	Tent Stakes		☐		9,98		
UT12-12	9	114,3	66,04	81,28	6	Umbrella Tents, 12 x 12		☐		19,05		

Sumário de Carregamento e Estimativa Pré-Teste do Espaço Necessário do Contêiner

Quantidade Total: 268
Itens Diferentes: 6
Peso, kg: 2.623
Volume, M.C.: 57,1
Contêineres: ,98
Metros: 11,7

Estimativa apenas de Espaço Necessário. Um teste de carga deve ser executado para resultados reais.

Arquivo Atual: MetricUnitsDemo.mdb

Classificar por: ● Quantidade ○ Descrição ○ Código do Produto ○ Prioridade ou Grupo

Nomear Carga a ser Impressa

Figura 22: Carga sintética para o container de 40’
Fonte: CargoWiz

4.2 Resultados das experiências

Nesta subseção, será tratado sobre o ambiente onde foram produzidos os experimentos e os resultados obtidos.

4.2.1 Ambiente de teste

Todas as execuções foram na mesma máquina com as seguintes configurações de *hardware*:

- Processador Intel® Core®2 Duo E7400 @2.80GHz, 3,24GB de RAM, *Motherboard* PCWARE, modelo PW-945GCX, sistema operacional Microsoft Windows XP Professional, versão 5.1.2600 Service Pack 3 Compilação 2600.

A versão instalada do *runtime* Java era a 1.6.0_15-b03.

4.2.2 Resultados obtidos

Por se tratar de algoritmo heurístico, os resultados obtidos pelo presente trabalho foram variados e determinados pelos fatores aplicados.

Os três fatores usados nos experimentos do *software* CargaFacil foram:

a) quantidade da população inicial;
b) quantidade de evoluções da população;
c) ambiente de carga e a quantidade de volumes a ser carregado.

As variações escolhidas para os fatores baseiam-se nas afirmações de Gehring e Bortfeldt (1997), onde afirma que uma população inicial menor que 50 conduz a um resultado pobre e incrementos acima de 300 evoluções não agregam melhores resultados. Com isso foram escolhidos valores menores e maiores a estes dois fatores.

Na Tabela 1, são relacionados os experimentos em ambiente de 20 pés com 163 volumes e seus fatores.

Tabela 1: Resultado da simulação em *container* de 20 pés e 163 volumes

Experimento	População inicial	Evoluções	Tempo do processamento em s	Quantidade de volumes carregados
1	10	20	8	117
2	20	200	53	139
3	50	100	33	132
4	50	200	56	139
5	50	300	86	134
6	50	500	134	140
7	50	1000	264	149
8	100	100	39	132
9	100	200	60	133
10	100	300	85	137
11	100	500	138	139
12	100	1000	265	146

Fonte: Autor

Na Tabela 2, são relacionados os experimentos em ambiente de 40 pés com 268 volumes e seus fatores.

Tabela 2: Resultado da simulação em *container* de 40 pés e 268 volumes

Experimento	População inicial	Evoluções	Tempo do processamento em s	Quantidade de volumes carregados
1	10	20	22	197
2	20	200	120	219
3	50	100	72	210
4	50	200	115	206
5	50	300	156	208
6	50	500	255	213
7	50	1000	474	217
8	100	100	85	216
9	100	200	130	214
10	100	300	177	210
11	100	500	269	215
12	100	1000	507	221

Fonte: Autor

O *software* comercial CargoWiz submetido ao teste com o ambiente de 40’ e carga de 268 volumes, levou aproximadamente 14 segundos para simular o carregamento e conseguiu embarcar todos os volumes.

O *software* comercial Cube-IQ submetido ao teste com o ambiente de 20’ e carga de 163 volumes, levou aproximadamente 17 segundos para simular o carregamento e conseguiu embarcar todos os volumes.

4.3 Análises dos resultados

4.3.1 Eficiência comparada aos *softwares* comerciais Cube-IQ e CargoWiz

Os resultados obtidos nas experiências foram desastrosos se comparados à eficiência e eficácia dos *softwares* comerciais.

Como se observa, o algoritmo implementado não foi eficaz em nenhum dos testes, pois o máximo de volumes carregados no ambiente de 20’ foram 149 unidades, o que corresponde a 91,41% do total carregado pelo Cube-IQ e no ambiente de 40’ foram carregados 221 volumes, o que corresponde a 82,46% do que conseguiu o CargoWiz.

No quesito eficiência, mesmo se desconsiderando que o algoritmo não conseguiu completar a carga, os resultados ficaram bem aquém do esperado. O CargaFacil levou 247 segundos a mais em comparação ao Cube-IQ e 493 segundos a mais com relação ao CargoWiz.

4.3.2 Análise final do *software* CargaFacil

Os resultados finais da ferramenta ficaram aquém da expectativa introduzida pelo artigo base deste trabalho. Comparados com a eficácia esperada de 95%, os resultados ficaram 3,59% abaixo no ambiente de 20’ e 12,54% abaixo no ambiente de 40’.

O fato do baixo desempenho da codificação em Java já havia sido citado por Klein (2006) no seu trabalho semelhante a este.

Mesmo que a implementação em Java não visasse desempenho comparável com outros programas desenvolvidos em linguagem mais apropriada ao caso, e tendo-se em consideração que o algoritmo é heurístico, o qual não necessariamente sempre obterá o melhor resultado, esperava-se a eficácia do carregamento total.

Um dos pontos a serem analisados, visto os resultados inadequados, é a parte do algoritmo de construção de torres o qual dá ênfase ao menor desperdício de espaço possível acima do volume base, mas não trata da adequação do volume base na área a ser preenchida.

A estratégia de montagem de torres mostrou-se limitada, pois não garante que as mesmas terão uma boa distribuição na base do ambiente a ser carregado. Com isso, caso a base da torre não possa ser devidamente colocada em algum espaço disponível, todos os volumes desta montagem ficam de fora, pois a estratégia não permite que o algoritmo genético as coloque de forma desmembrada.

Apesar do mal resultado, o objetivo deste trabalho foi alcançado na sua totalidade com a implementação do algoritmo do artigo de Gehring e Bortfeldt (1997) e as adequações propostas.

5 CONCLUSÕES

5.1 Trabalhos futuros

Como trabalhos futuros, pode-se sugerir um estudo para a substituição do algoritmo usado neste trabalho, por outro(s) algoritmo(s) que se propõe(m) a resolver o CLP ou, com menor impacto, o aprimoramento do aqui utilizado.

Sugere-se também uma análise mais criteriosa da primeira parte do presente algoritmo que gera as torres com o menor espaço desperdiçado, ou até mesmo a substituição do algoritmo do tipo guloso por outro mais eficiente.

Igualmente, a melhoria na *interface* com o usuário e alguns acréscimos na digitação também seriam bem apropriados.

Em complemento a esta trabalho, sugere-se implementar a regra de balanceamento de carga citado como restrição C5 além do balanceamento do peso, regra esta não citada no artigo base deste trabalho.

5.2 Considerações finais

Este trabalho implementou o algoritmo proposto no artigo base com ênfase ao uso de algoritmo genético com a proposta de se obter um *software* capaz de competir com os similares comerciais somando-se ao fato de ser de código aberto e de uso gratuito, assim como possivelmente ajustável pela empresa que o adotar.

Seguiu-se a risca o artigo base de Gehring e Bortfeldt (1997) independentemente de expectativas de resultados positivos.

Tem-se como resultado deste trabalho, um embrião para a continuação de um competidor a altura com os já existentes, mantendo-se a premissa de código aberto e gratuito. O resultado deste trabalho está disponibilizado no Portal da Inovação do Vale do Paranhana (http://portaldainovacao.faccat.br).

Apesar dos resultados pouco satisfatórios de eficiência e eficácia, conclui-se que, com a união de trabalhos já desenvolvidos até aqui e somando outros que virão, é bastante factível a arte final desta ferramenta.

6 REFERÊNCIAS

BENÍTEZ, César Manuel Vargas; LOPES, Heitor S. **Algoritmo genetico aplicado a predicao da estrutura de proteinas utilizando o modelo 3D-HP Side Chain**. Universidade Tecnóloga Federal do Paraná. 2007.

BENITTI, Fabiane Barreto; ZIMMERMANN, Ana Paula. **Testware:** ferramenta de planejamento e execução de casos de teste. UNIVALI, FURB. 2005.

BERZ, Everton Luís. **Um sistema para o problema do caixeiro viajante utilizando algoritmos genéticos**. Monografia (Graduação em Sistemas de Informação) - Faculdades Integradas de Taquara. 2008.

BIASI, Luciano B.; BECKER, Karin. Geração automatizada de drivers e stubs de teste para JUnit a partir de especificações U2TP. **Anais**. XX Simpósio Brasileiro de Engenharia de Software. Porto Alegre, 2006.

BISCHOFF, E. E.; JANETZ, F.; RATCLIFF, M. S. W. Loading pallets with non-identical items. **European Journal of Operational Research**, Vol. 84, pp. 681 - 692. 1995

BISCHOFF, E. E.; RATCLIFF, M. S. W. **Issues in the development of approaches to container loading.** Omega, Vol. 23, pp. 377-390. 1995.

BITTENCOURT, J. R.; OSÓRIO, F. ANNeF – Artificial Neural Networks Framework: Uma Solução Software Livre para o Desenvolvimento, Ensino e Pesquisa de Aplicações de Inteligência Artificial Multiplataforma. **Anais.** II Workshop sobre Software Livre. Porto Alegre: SBC, 2001.

BORTFELDT, A. A genetic algorithm for the container loading problem. **Anais**. Proceedings of the Conference on Adaptive Computing and Information Processing, Londres, Vol. 2, pp. 25-32. 1994.

CANTÙ, Marco. **Dominando o Delphi 7 – “A Bíblia”**. São Paulo. Pearson Education, 2003.

CARGOWIZ. **Optimize Cargo Load Planning - Truck and Container Loading Software**. Disponível em: <http://www.softtruck.com/>. Acesso em: 06/08/2008.

COMPRECAR. Disponível em: <http://www.comprecar.com.br/caminhoes/campinas/carreta/semi-reboque-furgao/0/75758>. Acesso em: 25 out. 2009.

CARROCERIAS ROCHA. Disponível em: <http://www.carroceriasrocha.com.br/carga-seca.htm>. Acesso em: 25 out. 2009.

DEITEL, H. M.; DEITEL, P. J. **Java, como programar**. 3ª ed. Porto Alegre: Bookman, 2001.

DISPOSITIVO. Disponível em: <http://www.dpi.ufv.br/disciplinas/inf390/files/paginas/gbdi.icmc.sc.usp.br/documentacao/apostilas/cg/ap02.html>. Acesso em: 04 dez. 2008.

DUBKE , Alessandra Fraga; FERREIRA, Fábio. R. N.; RODRIGUES, Gisela G.; PIZZOLATO, Nélio D. O processo de consolidação de cargas por operadores logísticos internacionais: Reflexões com vistas ao comércio exterior brasileiro. **Anais.** XXIV ENEGEP - Encontro Nacional de Engenharia de Produção. Florianópolis, 2004.

FEDERIZZI, Gustavo Link. Disponível em <http://www.inf.ufrgs.br/gppd/disc/inf01008/trabalhos/sem01-1/t2/apis_xml_java/#Intro>. Acesso em: 25 out. 2009.

FORMATO PDF. Disponível em: <http://pt.wikipedia.org/wiki/Portable_Document_Format>. Acesso em: 04 dez. 2008.

GEHRING, H.; BORTFELDT, A. A Genetic Algorithm for Solving the Container Loading Problem. **International Transactions in Operational Research**, Vol. 4, Nº 5/6, pp. 401-418. 1997.

GLADCHEFF, Ana P.; ZUFFI, Edna M.; SILVA, Dilma M. da. Um Instrumento para Avaliação da Qualidade de Softwares Educacionais de Matemática para o Ensino Fundamental. **Anais.** VII WORKSHOP DE INFORMÁTICA NA ESCOLA, Fortaleza, CE, Brasil, 2001.

GRANDE ENCICLOPÉDIA LAROUSSE CULTURAL. São Paulo: Librairie Larousse, 1988. 30 volumes.

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH. **Elsevier Science Ltd**. 1997.

JAIN, Raj. **The art of computer systems performance analysis**. USA: John Wiley & Sons, Inc. 1991.

KIRNER, C.; KIRNER, T.G.; JÚNIOR, N.C.; BUK, C.V. Título do artigo. **Anais.** VII Symposium on Virtual Reality. UFPA. SP, 2004.

KLEIN, Rodrigo. **Biblioteca java para solucionar o problema de carregamento de contêineres.** Feevale. 2006.

LARMAN, Craig. **Utilizando UML e padrões**. 2. Ed. Porto Alegre: Bookman, 2004.

LOH, T. H.; NEE, A. Y. C. A packing algorithm for hexahedral boxes. **Anais**. Proceedings of the Conference of Industrial Automation, Singapura, pp. 115 - 126. 1992

LOPES, Heitor S. Algoritmos genéticos em projetos de engenharia: aplicações e perspectivas futuras. In: SBAI, 1999, São Paulo. **Anais.** 4o. Simpósio Brasileiro de Automação Inteligente**.** São Paulo: Tec Art Editora, 1999.

MAGICLOGIC Optimization Inc. **Cube-IQ Load Planning & Optimization Software**. Disponível em <http://www.magiclogic.com>. Acesso em: 06 ago. 2008.

MARCHESI, Bruno; STELLE, Álvaro Luiz; LOPES, Silvério. Programação genética na detecção de eventos epilépticos resultados preliminares. **Anais**. III Congresso Brasileiro de Redes Neurais, p. 374-378. 1997.

MARTINS, Vinicius. Proposta de uma ferramenta de integração entre sistemas ERP-SCADA: caso prático. **Anais**. XXII Encontro Nacional de Engenharia de Produção. Curitiba, PR. 2002.

MEDEIROS, ERNANI SALES DE. **Desenvolvendo software com UML 2.0: definitivo.** São Paulo: Pearson Makron Books, 2004.

MELLO, R. S. Gerenciamento de Dados XML **Anais.** V Escola de Informática Norte da SBC, 2003, Palmas. ENCOINFO/EIN 2003. Palmas: Centro Universitário Luterano de Palmas - ULBRA, 2003.

MORABITO, Reinaldo; ARENALES, Marcos. **International Transactions in Operational Research**, Vol. 1, Nº 1, pp. 59-73. 1994.

NGOI, B. K. A.; TAY, M. L.; CHUA, E. S. Applying spatial representation techniques to the container packing problem. **International Journal of Production Research**, Vol. 32, pp. 111 - 123. 1994

NOVO MILÊNIO. Disponível em :<http://www.novomilenio.inf.br/porto/conteinm.htm> Acesso em: 25 out. 2009.

OMG. In: Documents associated with UML Version 2.2. Disponível em: <http://www.omg.org/spec/UML/2.2>. Acesso em 31 out. 2009.

PRESSMAN, ROGER S. **Engenharia de software.** 5ª ed. Rio de Janeiro: McGraw-Hill, 2002.

REBELLO, Fabrício Rocha; HAMACHER , Silvio . Uma proposta de algoritmo genético de duas fases para roteamento de veículos. In: XXXII SBPO, 2000, Viçosa. **Anais.** XXXII Simpósio Brasileiro de Pesquisa Operacional. Viçosa: 2000.

RESOLUCAO. Disponível em: <http://www.dpi.ufv.br/disciplinas/inf390/files/paginas/gbdi.icmc.sc.usp.br/documentacao/apostilas/cg/ap01.html>. Acesso em: 04 dez. 2008.

RUSSEL, Stuart; NORVIG, Peter. **Artificial Intelligence - A Modern Approach**. Upper Saddle River, New Jersey. Prentice-Hall, Inc. 1995.

SAMPAIO, Cleuton. **SOA e Web Service em Java.** Rio de Janeiro: Brasport, 2006.

SELOW, Roberto; LOPES, Heitor S.; NEVES JR, Flávio. Algoritmos genéticos para o problema de aninhamento na indústria de embalagens: um estudo com modelos reduzidos. **Anais.** V Simpósio Brasileiro de Automação Inteligente - SBAI, Canela, RS, 7-9/novembro. 2001.

SILVA, Fábio Tadeu Paiva da; TEIXEIRA, André. Teste de Unidade – Junit em Java. UCSAL. 2008.

SISLOG. **Significado de Open Source.** Disponível em: <http://www.sislog.com/article. php3?id_article=10>. Acesso em: 04 dez. 2008.

SORROCHE, Rogério; LOPES, Mauricio. C. Análise comparativa de algoritmos de grafos para um sistema de auxílio à matrícula de alunos. **Anais**. XII Seminário de Computação, 2003, Blumenau (SC). Anais do XII SEMINCO, 2003.

URIARTT, TIAGO MASOTTI. **Simulador para empacotamento 3D.** Monografia (Graduação em Ciência da Computação) - Feevale, Novo Hamburgo, 2007

APÊNDICES

APÊNDICE A – XML Schema para validação do lote de carga

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns="http://philler.com.br/cargafacil"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://philler.com.br/cargafacil"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="cargafacil" type="TCargafacil">
     <xs:annotation>
           <xs:documentation>Schema XML de validacao do
carregamento</xs:documentation>
     </xs:annotation>
  </xs:element>

  <xs:complexType name="TCargafacil">
    <xs:sequence>
      <xs:element ref="identificacao"/>
      <xs:element ref="ambienteCarga"/>
      <xs:element ref="pedidos"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="identificacao">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="idLote" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="dataEmbarque" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="observacoes" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="idLote">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="dataEmbarque" type="xs:date"/>

  <xs:element name="observacoes">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="obs"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="obs" type="xs:string"/>
  <xs:element name="ambienteCarga">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element ref="idAmbiente" minOccurs="0"/>
        <xs:element ref="descricaoAmbiente"/>
        <xs:element ref="comprimentoUtil"/>
        <xs:element ref="larguraUtil"/>
        <xs:element ref="alturaUtil"/>
        <xs:element ref="capacidadeCargaKg" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="idAmbiente" type="xs:string"/>

  <xs:element name="descricaoAmbiente">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="5"/>
     </xs:restriction>
  </xs:simpleType>
  </xs:element>

  <xs:element name="comprimentoUtil">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
     </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="larguraUtil">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
     </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="alturaUtil">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
     </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="capacidadeCargaKg">
    <xs:simpleType>
      <xs:restriction base="xs:decimal">
        <xs:minInclusive value="1"/>
     </xs:restriction>
    </xs:simpleType>
  </xs:element>
```

```
  <xs:element name="pedidos">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="pedido" minOccurs="1"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="pedido">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="detalhes"/>
      </xs:sequence>
      <xs:attribute name="cliente" use="required" type="xs:string"/>
      <xs:attribute name="numero" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="detalhes">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="item"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="item">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="referencia"/>
        <xs:element ref="descricao" minOccurs="0"/>
        <xs:element ref="quantidade"/>
        <xs:element ref="pesoUnit" minOccurs="0"/>
        <xs:element ref="valor" minOccurs="0"/>
        <xs:element ref="comprimento"/>
        <xs:element ref="largura"/>
        <xs:element ref="altura"/>
        <xs:element ref="orientacao"/>
        <xs:element ref="maximoPilha" minOccurs="0"/>
        <xs:element ref="maximoPesoSobre" minOccurs="0"/>
        <xs:element ref="podeSerEmpilhado" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="sequencia" use="required"
type="xs:positiveInteger"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="referencia" type="xs:string"/>
  <xs:element name="descricao" type="xs:string"/>
  <xs:element name="quantidade">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
```

```
     </xs:restriction>
  </xs:simpleType>
  </xs:element>

  <xs:element name="pesoUnit" type="xs:decimal"/>
  <xs:element name="valor" type="xs:decimal"/>

  <xs:element name="comprimento">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
     </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="largura">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
     </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="altura">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
     </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="orientacao">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="[1246]"/>
     </xs:restriction>
  </xs:simpleType>
  </xs:element>

  <xs:element name="maximoPilha" type="xs:positiveInteger"/>
  <xs:element name="maximoPesoSobre" type="xs:decimal"/>
  <xs:element name="podeSerEmpilhado" type="xs:boolean"
default="true"/>
</xs:schema>
```

APÊNDICE B – Exemplo de lote de carga no padrão XML

```
<?xml version="1.0" encoding="UTF-8"?>
<cargafacil xmlns="http://philler.com.br/cargafacil"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://philler.com.br/cargafacil
carga.xsd">
  <identificacao>
    <idLote>1</idLote>
    <dataEmbarque>2009-09-30</dataEmbarque>
    <observacoes>
      <obs>linha com observacao 1</obs>
      <obs>linha com observacao 2</obs>
    </observacoes>
  </identificacao>
  <ambienteCarga>
    <idAmbiente>DryC20</idAmbiente>
    <descricaoAmbiente>Dry Container 20' (5,82
metros)</descricaoAmbiente>
    <comprimentoUtil>5893</comprimentoUtil>
    <larguraUtil>2349</larguraUtil>
    <alturaUtil>2388</alturaUtil>
  </ambienteCarga>
  <pedidos>
    <pedido numero="123" cliente="ABC Comercio">
      <detalhes>
        <item sequencia="1">
          <referencia>CDupla</referencia>
          <descricao>Cuba dupla (3 unid)</descricao>
          <quantidade>12</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
          <comprimento>790</comprimento>
          <largura>470</largura>
          <altura>290</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
        <item sequencia="2">
          <referencia>CDuplaEco</referencia>
          <descricao>Cuba dupla Econ.(5 unid)</descricao>
          <quantidade>8</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
          <comprimento>790</comprimento>
          <largura>470</largura>
          <altura>290</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
```

```
    </item>
    <item sequencia="3">
      <referencia>CDuplaRed</referencia>
      <descricao>Cuba Dupla Red. (3 unid)</descricao>
      <quantidade>24</quantidade>
      <pesoUnit>0</pesoUnit>
      <valor>0.00</valor>
      <comprimento>475</comprimento>
      <largura>865</largura>
      <altura>305</altura>
      <orientacao>6</orientacao>
      <maximoPilha>9999</maximoPilha>
      <maximoPesoSobre>9999999.99</maximoPesoSobre>
      <podeSerEmpilhado>true</podeSerEmpilhado>
    </item>
    <item sequencia="4">
      <referencia>CRed 30</referencia>
      <descricao>Cuba Red. 30 (6 unid)</descricao>
      <quantidade>4</quantidade>
      <pesoUnit>0</pesoUnit>
      <valor>0.00</valor>
      <comprimento>350</comprimento>
      <largura>690</largura>
      <altura>445</altura>
      <orientacao>6</orientacao>
      <maximoPilha>9999</maximoPilha>
      <maximoPesoSobre>9999999.99</maximoPesoSobre>
      <podeSerEmpilhado>true</podeSerEmpilhado>
    </item>
    <item sequencia="5">
      <referencia>CRed 38</referencia>
      <descricao>Cuba Red. 38 (6 unid)</descricao>
      <quantidade>4</quantidade>
      <pesoUnit>0</pesoUnit>
      <valor>0.00</valor>
      <comprimento>845</comprimento>
      <largura>435</largura>
      <altura>445</altura>
      <orientacao>6</orientacao>
      <maximoPilha>9999</maximoPilha>
      <maximoPesoSobre>9999999.99</maximoPesoSobre>
      <podeSerEmpilhado>true</podeSerEmpilhado>
    </item>
    <item sequencia="6">
      <referencia>CRedEco 35</referencia>
      <descricao>Cuba Red. Econ.35 (10unid)</descricao>
      <quantidade>3</quantidade>
      <pesoUnit>0</pesoUnit>
      <valor>0.00</valor>
      <comprimento>785</comprimento>
      <largura>400</largura>
      <altura>445</altura>
      <orientacao>6</orientacao>
      <maximoPilha>9999</maximoPilha>
```

```
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
        <item sequencia="7">
          <referencia>CRet 40X43</referencia>
          <descricao>Cuba Ret. 40x34 (6un)</descricao>
          <quantidade>30</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
          <comprimento>445</comprimento>
          <largura>395</largura>
          <altura>445</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
        <item sequencia="8">
          <referencia>CRet 47X30</referencia>
          <descricao>Cuba Ret.STD.47x30(12un)</descricao>
          <quantidade>10</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
          <comprimento>515</comprimento>
          <largura>355</largura>
          <altura>445</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
        <item sequencia="9">
          <referencia>Fila amara 120X50</referencia>
          <descricao>Fila amara120x50(63un.)</descricao>
          <quantidade>8</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
          <comprimento>1220</comprimento>
          <largura>2330</largura>
          <altura>510</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
        <item sequencia="10">
          <referencia>LOval 36X26</referencia>
          <descricao>Lavabo Oval 36x26 (5un.)</descricao>
          <quantidade>6</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
          <comprimento>395</comprimento>
          <largura>610</largura>
```

```
          <altura>445</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
        <item sequencia="11">
          <referencia>LOval 36X26</referencia>
          <descricao>Lavabo Oval 36x26 (6un.)</descricao>
          <quantidade>9</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
          <comprimento>335</comprimento>
          <largura>425</largura>
          <altura>165</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
        <item sequencia="12">
          <referencia>PiaEnc 116X50</referencia>
          <descricao>Pia Enc. 116x50 (100un.)</descricao>
          <quantidade>2</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
          <comprimento>1230</comprimento>
          <largura>1130</largura>
          <altura>2250</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
        <item sequencia="13">
          <referencia>ValvMeber</referencia>
          <descricao>Valv. Meber (60un.)</descricao>
          <quantidade>1</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
          <comprimento>160</comprimento>
          <largura>550</largura>
          <altura>430</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
        <item sequencia="14">
          <referencia>Valv3"</referencia>
          <descricao>Valv.3" s/ esc.(25un.)</descricao>
          <quantidade>42</quantidade>
          <pesoUnit>0</pesoUnit>
          <valor>0.00</valor>
```

```
          <comprimento>120</comprimento>
          <largura>310</largura>
          <altura>525</altura>
          <orientacao>6</orientacao>
          <maximoPilha>9999</maximoPilha>
          <maximoPesoSobre>9999999.99</maximoPesoSobre>
          <podeSerEmpilhado>true</podeSerEmpilhado>
        </item>
      </detalhes>
    </pedido>
  </pedidos>
</cargafacil>
```