António Filipe de Oliveira Paiva

Geração Automática de Modelos de Simulação de uma Linha de Produção na Indústria Têxtil

Dissertação submetida à Escola de Engenharia da Universidade do Minho
para a obtenção do grau de Mestre em Engenharia Industrial

Universidade do Minho
Escola de Engenharia
Departamento de Produção e Sistemas
Guimarães – 2005

António Filipe de Oliveira Paiva

Geração Automática de Modelos de Simulação de uma Linha de Produção na Indústria Têxtil

Dissertação submetida à Escola de Engenharia da Universidade do Minho
para a obtenção do grau de Mestre em Engenharia Industrial

Realizada sob a supervisão cientifica do
Prof.. Guilherme Augusto Borges Pereira,
Professor Auxiliar do Departamento de Produção e Sistemas da Escola de Engenharia da Universidade do Minho
e do
Prof.. Ricardo Jorge Silvério de Magalhães Machado,
Professor Auxiliar do Departamento de Sistemas de Informação da Escola de Engenharia da Universidade do Minho

Universidade do Minho
Escola de Engenharia
Departamento de Produção e Sistemas
Guimarães – 2005

Aos Meus Pais.

AGRADECIMENTOS

Prof. Guilherme Augusto Borges Pereira

Prof. Ricardo Jorge Silvério de Magalhães Machado

Dr. Carlos Manuel Fernandes da Silva

Eng. Pedro António Oliveira Vieira

RESUMO

Esta dissertação tem como objectivo a construção de um sistema de apoio à decisão para estudar o impacto de diversas alternativas de gestão da produção aplicadas a um sistema produtivo da indústria têxtil.

Desenvolveu-se um *Modelo Base*, em linguagem de simulação, que integra o processo de tricotar, no âmbito da produção para a indústria têxtil. As características específicas dos recursos utilizados nesta produção foram desenvolvidas de forma a permitir uma utilização iterativa e susceptível de serem parametrizáveis pelo utilizador.

É feita uma abordagem à simulação e às suas vantagens na aplicação à monitorização remota e a estudos de viabilidade. Apoiado na simulação, construiu-se uma ferramenta interactiva, permitindo a geração automática de modelos com diferentes estratégias de controlo que sustenta a viabilidade do sistema a propor.

Suportado pela simulação em ambiente *Arena*® elaborou-se uma aplicação que possibilita ao utilizador projectar e racionalizar os meios de produção disponíveis. Nesta ferramenta foram desenvolvidas diversas estratégias alternativas de controlo para cada área considerada relevante.

A ferramenta foi desenvolvida com carácter genérico, com uma enorme flexibilização (do ponto de vista da definição de politicas e estratégias na utilização dos recursos existentes), e centrada na implementação dos principais índices de desempenho. Está dotada de diversas estratégias de controlo, ao nível da gestão dos materiais, do controlo do operador de máquina, do controlo da equipa de manutenção e do escalonamento da produção

A parametrização é outra característica importante da ferramenta. O modelo é definido pelo do utilizador, aumentando a participação, influência e responsabilidade do utilizador no resultado final.

Esta aplicação foi elaborada com uma interface amigável e com o propósito de ser acessível a utilizadores não especialistas quer em simulação, quer em programação.

Todas as modificações ocorridas no estado do sistema durante a execução da simulação são registadas para possibilitar posteriores estudos e analises, bem como para permitir melhorias em simulações futuras.

ABSTRACT

The objective of this dissertation is the construction of a decision support system to study the impact of distinct production management alternatives applied to a textile industry productive system.

A *Modelo Base* was developed using a simulation language, which integrates the tricot process within textile industry production scope. The specific characteristics of the production used resources had been developed in a way to allow an iterative utilization and parameterization by the user.

An approach to the simulation and its advantages in remote monitoring and viability studies is followed. An interative tool was developed, using simulation, allowing the automatic generation of models with different control strategies of which supports the viability of the proposed system.

An application in *Arena*® was developed to allow the user to project and rationalize the available means of production. Different alternative control strategies for each relevant area had been developed in this tool.

A generic tool was developed, with enormous flexibility (on the point of view of the definition of politics and strategies in the use of the existing resources) and focussed on the implementation of the main performance indexes. It is endowed with different control strategies, to the level of materials management, machine operator control, maintenance teams control and production scheduling.

The parameterization is another important characteristic of the tool. The model is defined by the user, increasing the user participation, influence and responsibility in the final result.

This application shows a friendly interface with the purpose of accessibility to users without any kind of simulation and programming specialized skills.

Every change in the state of the system, occurred during the execution of the simulation, are registered to make possible future studies and analyses as well as allowing improvements in future simulations.

ÍNDICE GERAL

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE NOMENCLATURAS

ASPOL	A Simulation Process-Oriented Language (Linguagens de Simulação)
ActiveX	Livraria da Microsoft para ligar objectos
Arena	Ferramenta de Simulação
BEST/1	Ferramenta de Simulação
BETHSIM	Ferramenta de Simulação
BOM	Bill of Materials
C	Linguagem de Programação
C++	Linguagem de Programação
CMF	Ferramenta de Simulação
CSIM	Biblioteca de Simulação
CSL	Control and Simulation Language (Linguagens de Simulação)
DAO	Data Access Objects
Delphi	Linguagem de Programação
ECSL	Extended Control and Simulation Language (Linguagens de Simulação)
EFC	Biblioteca de Simulação
EFM	Biblioteca de Simulação
EOI	Economic Order Interval
EOQ	Economic Order Quantity
Excel	Aplicação da Microsoft Windows
FIFO	First In First Out
FIVE	Ferramenta de Simulação
FORTRAN	Linguagem de Programação
GAMSTAF	Gerador Automático de Modelos de Simulação para a Têxtil António Falcão
GASP	Graph Algorithm and Software Package (Linguagens de Simulação)
GPSS	General Purpose Simulation System (Linguagens de Simulação)
HOS	Ferramenta de Simulação
HPSIM	Biblioteca de Simulação
HVF	High Value First
IDSS	Ferramenta de Simulação
Informix	Base de Dados
Java	Linguagem de Programação
LIFO	Last In Last Out
LUC	Least Unit Cost
LVF	Lower Value First
Macintosh	Empresa de Software
MedModel	Ferramenta de Simulação
MFC	Microsoft Foundation Classes
Microsoft Office	Aplicação da Microsoft Windows
Microsoft Windows	Empresa de Software
MOPADS	Ferramenta de Simulação
MRP	Material Requirement Planning, ou Planificação das Necessidades de Materiais
NETWORLK	Ferramenta de Simulação
OLE	Object Linking and Embedding
Oracle	Empresa de Software, Base de Dados
PERFORMS	Ferramenta de Simulação
PET	Ferramenta de Simulação
PL	Programação Linear
POQ	Periodic Order Quantity
SAD	Sistema de Apoio à Decisão
SAINT	Ferramenta de Simulação
ServiceModel	Ferramenta de Simulação

SIMAN	Simulation Modeling And Analysis (Linguagens de Programação)
SIMPACK	Biblioteca de Simulação
SIMPL/IX	Linguagens de Simulação
SIMULA	Linguagens de Simulação
SLAM II	Linguagens de Simulação
SMPL	Biblioteca de Simulação
SOL Server	Base de Dados
SQL	Structured Query Language (Linguagem de Programação)
TAF	Textil António Falcão
UMPredictor	Ferramenta de Simulação
UML	Unified Modeling Language
VBA	Visual Basic for Applications
VIS	Visual Interactive Simulation
Visual Basic	Linguagem de Programação
Windows	Sistema Operativo da Microsoft Windows
XL	Ferramenta de Simulação

Capítulo 1

Introdução

Este trabalho tem como objectivo principal desenvolver um sistema de apoio à decisão (SAD), baseado na simulação, para servir como ferramenta de suporte ao estudo do impacto de diferentes estratégias de implementação da estrutura produtiva.

A viabilidade de um sistema deste tipo pressupõe a existência de uma ferramenta informática e de um modelo de simulação.

A ferramenta informática garante o interface entre o utilizador e o simulador. Nesta ferramenta o utilizador define o sistema que pretende criar, parametriza-o e selecciona as estratégias de controlo a aplicar ás entidades e recursos que o compõem.

O Modelo, representativo do sistema real, controla e garante a implementação das estratégias seleccionadas pelo utilizador. Ao modelo acoplou-se uma *template* com a definição dos módulos, em linguagem de simulação, que representam o funcionamento dos recursos e entidades do sistema real.

Estas duas ferramentas, denominadas por SAD, têm capacidade para gerar automaticamente modelos de simulação com diversas estratégias de controlo, ao nível da gestão dos materiais, do controlo do operador de máquina, do controlo da equipa de manutenção e do escalonamento da produção.

Esta SAD foi implementada de forma a serem flexível, genérica, parametrizável, e que proporcione uma visão prévia do comportamento do sistema e possibilite prever a necessidade de disponibilização de máquinas, operadores e materiais para a realização das encomendas no prazo definido.

Outra característica importante, é a possibilidade de estudar de alternativas ao nível dos factores intervenientes do ambiente produtivo bem como das estratégias que o controlarão sem ter que redesenhar todo o sistema de cada vez que se pretende alguma alteração.

1.1 Enquadramento

Nesta secção, descreve-se a empresa, o ambiente produtivo, o sistema de produção, os intervenientes e suas principais funções do sistema que serviu de base ao estudo tratado nesta dissertação.

Trata-se de uma empresa têxtil, aqui denominada de TAF, que produz meias e collants para senhoras.

As principais áreas de produção de meias/collants da TAF estão fisicamente separadas, todas elas funcionam com entrada e saída de matéria-prima.

Apesar de tentar demonstrar o funcionamento de todas as áreas necessárias à produção das meias/collants vai ser dada uma atenção especial à tricotagem, zona onde se situam os teares, que vão ser objecto de estudo mais detalhado. Vai-se considerar a área de tricotagem como uma empresa que apenas se dedica a essa operação, isto é, transforma matéria-prima em meia tubular.

Sistema Produtivo

Existem cinco áreas de produção: Tricotar, Confeccionar, Revistar, Tingir e Embalar. A sequência de produção no caso das:

1) meias é: Tricotar → Confeccionar → Tingir → Revistar → Embalar

2) collants é: Tricotar → Confeccionar → Revistar → Tingir → Embalar

Tricotar meias/collants**:** Produzir a meia tubular, com um tamanho e uma espessura, de entre várias alternativas.

Confeccionar meias/collants: cortar e coser a meia tubular, de forma a obter o collant. No caso das meias, apenas existe a operação de coser.

Revistar meias/collants: Verificar a qualidade dos produtos, segundo parâmetros e regras definidas. No caso das meias, esta funcionalidade é executada entre as operações de tingir e embalar, sendo os parâmetros a verificar o peso, as medidas e outros defeitos de processo (roto, aberto, etc.). No caso dos collants, esta funcionalidade é executada entre a operação confeccionar e tingir, sendo os parâmetros a verificar o peso, as medidas, e outros defeitos de processo (roto, aberto, painel mau, etc.).

Tingir meias/collants: atribuir uma cor, de várias possíveis, aos produtos.

Embalar meias/collants: acondicionar o produto acabado, em três níveis de embalagem. O primeiro nível, tipicamente uma embalagem (caixas, sacos), contendo um número reduzido de produtos, o segundo nível tipicamente uma embalagem maior (caixotes) que definem um lote, e o terceiro nível uma palete com várias embalagens (caixotes) empilhadas, e devidamente cintadas.

Intervenientes no Sistema Produtivo

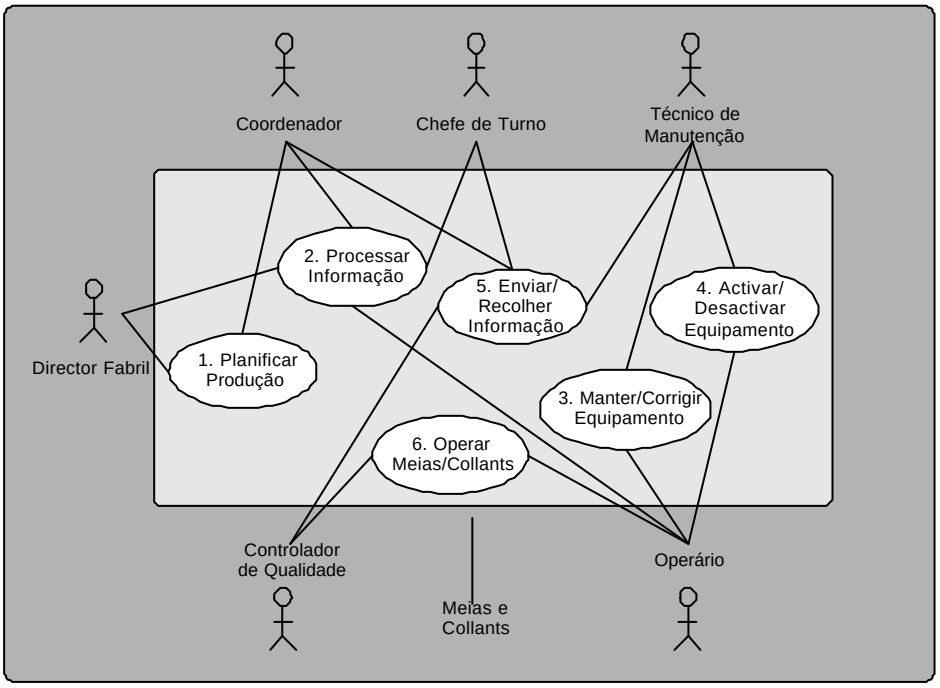


Figura 1.1 – Diagrama UML de funcionalidades da produção de meias/collants.

Em todo o processo de fabrico, existe um conjunto de actores, tal como está representado na Figura 1.1. Cada um interage com o sistema de uma forma muito própria. As meias e os collants também são actores do sistema. Cada actor *utiliza* o sistema de uma forma característica, pelo que é possível descrever cada forma de utilização. A esta utilização podemos chamar funcionalidade, característica do sistema, disponibilizada por este aos actores, que a cada uma têm acesso.

Fluxo de Materiais (área de tricotagem)

1. Área de Stocks

Entre cada área de produção existem áreas de armazenamento (stocks) de matéria-prima, produto semi-acabado e produto final. Serão referenciadas com a seguinte notação:

S_{iop}

i representa a origem (in) dos materiais em stock, $i \in \{0, T, F, R, G, E\}$

o representa o destino (out) dos materiais em stock, $o \in \{T, F, R, G, E, 0\}$

p representa o produto, $p \in \{M, C\}$

em que :

0 - entrada ou saida da fabrica
T - Tricotar
F - Confeccionar
R - Revistar
G - Tingir
E - Embalar
M - Meias
C - Collants

Figura 1.2 – Representação das notações usadas nas áreas de stocks

Exemplo: S_{0TC} = stock de entrada de matéria na área de Tricotar da produção de Collants.

Assim, a sequência dos stocks para a produção de meias é:

$S_{0TM} \longrightarrow S_{TFM} \longrightarrow S_{FGM} \longrightarrow S_{GRM} \longrightarrow S_{REM} \longrightarrow S_{E0M}$

No caso da produção de collants, a sequência é:

$$S_{0TC} \longrightarrow S_{TFC} \longrightarrow S_{FRC} \longrightarrow S_{RGC} \longrightarrow S_{GEC} \longrightarrow S_{E0C}$$

2. Entrada de Matérias-Primas na Área de Tricotagem

A Figura 1.3 mostra a sequência de acções relativas à entrada de matérias-primas, na área de tricotagem. As matérias-primas dão entrada nos stocks S_{0TC} e S_{0TM}. O operário de texturização regista a entrada no sistema. Consoante a produção necessita de matéria-prima o operário de abastecimento retira-a dos stocks e coloca num *buffer* intermédio, onde o operário de máquina vai buscar, sempre que necessita de reabastecer uma máquina.

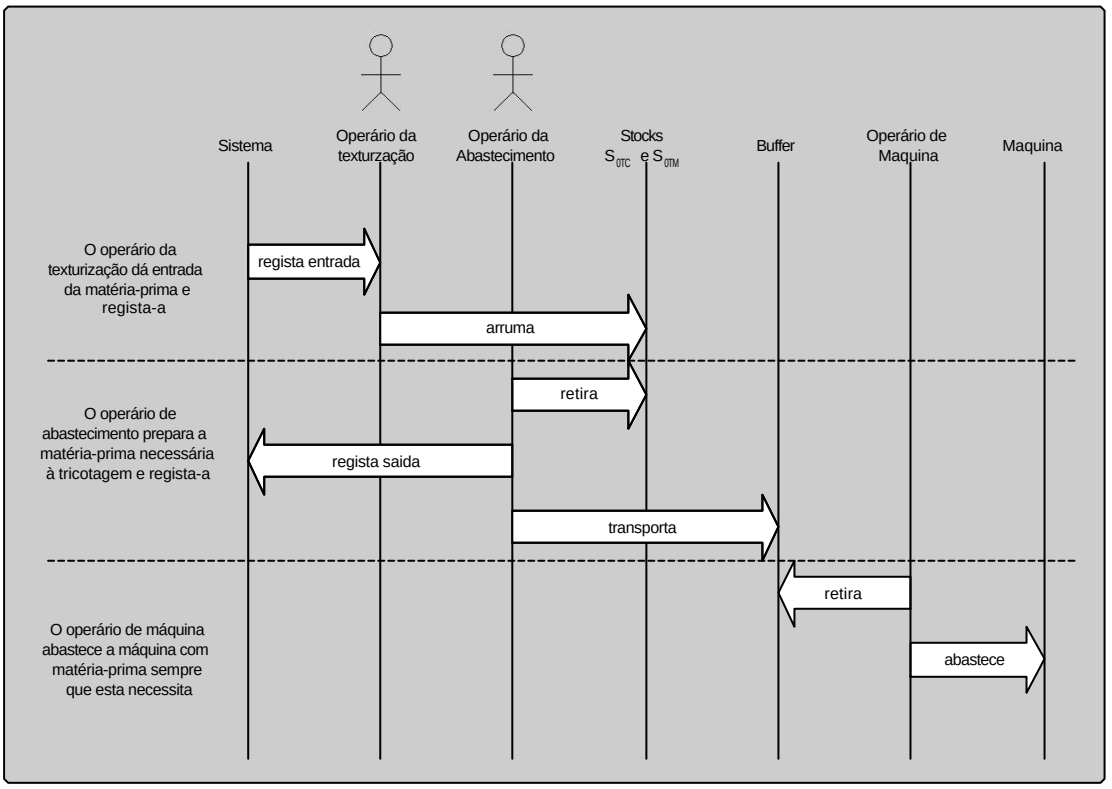


Figura 1.3 – Sequência de entrada de matéria-prima na área de tricotagem [RODR00]

3. Descarregar Produto da Máquina na Área de Tricotagem

A Figura 1.4 ilustra a sequência de acções relativas ao descarregar do produto da máquina, O operário retira o produto acabado da máquina e coloca-o num *buffer*. Faz o

registo no sistema. O operário de abastecimento confere a quantidade e regista-a no sistema. Em seguida levanta o material do *buffer* e transporta-o para os stocks S_{TCM} e S_{TCC}.

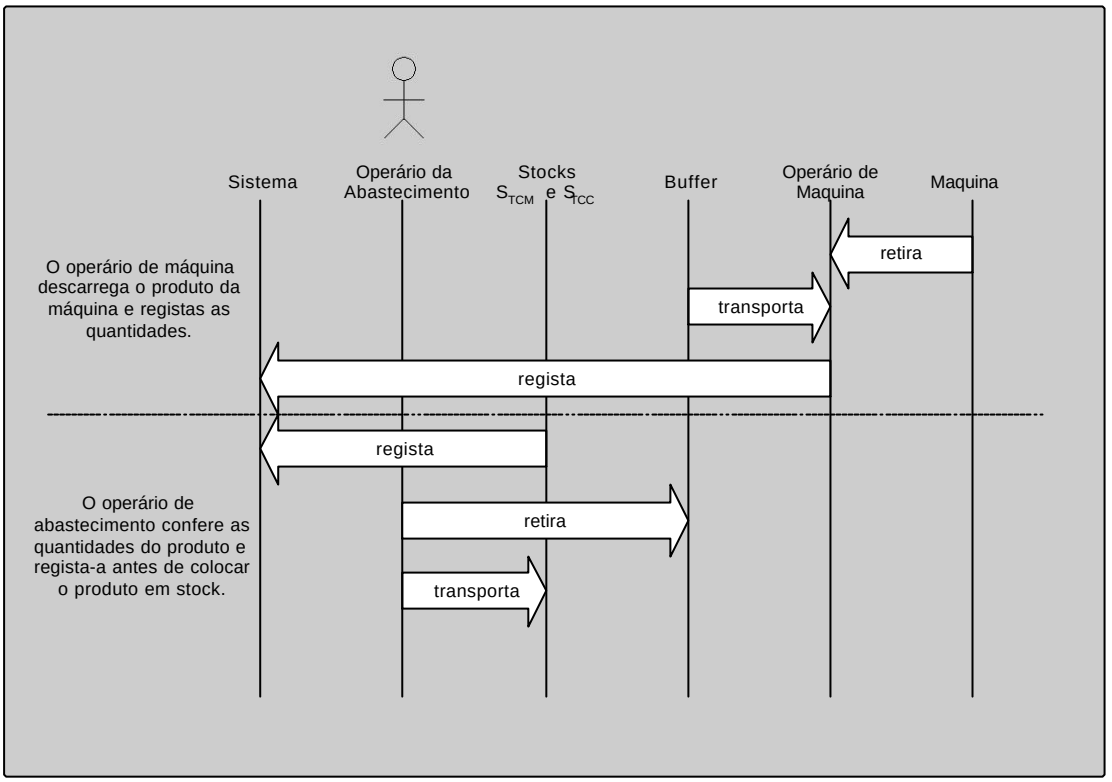


Figura 1.4 – Sequência do descarregar produto das máquinas na área de tricotagem[RODR00]

Tricotagem

A área de tricotagem trabalha de Segunda a Sexta, 24 horas por dia, em 3 turnos fixos. Consiste num parque de máquinas que executa a tricotagem da meia tubular, em tamanhos e formas diversas, e da malha de reforço. O parque de máquinas está dividido em 5 zonas, cada uma com um operador de máquina. Chefiando esta área existe o chefe de turno, que é também técnico de manutenção. As fases mais importantes do processo abordadas nesta área produtiva são:

1. Planeamento

O Planeamento é elaborado pelo Director Fabril com o Coordenador, numa reunião semanal. Tem como base as necessidades mensais dos clientes e a produção realizada até aquele dia. É o Coordenador que atribui a produção a cada máquina. Todas as máquinas têm capacidade de produzir todo o tipo de produto, à excepção da única máquina que executa a tricotagem da malha de reforço.

2. Armazém Entrada/Saída de Matérias-Primas

Neste armazém dão entrada três matérias-primas: fio texturizado (medidas, de 20 a 70, duas torções, lotes diferentes), fio poliester (várias cores), lycra (um único tipo 195 DTEX). Estes materiais são devidamente acondicionados, identificados e registados. As saídas de matérias-primas são registadas pelo operador de máquina ou pelo operário de abastecimento, procurando cumprirem o FIFO.

A gestão de stocks é feita semanalmente depois da reunião semanal de produção. O responsável pelo planeamento requisita os lotes de fio que no momento se estão a consumir para evitar alterações nas máquinas. A quantidade é calculada com base na produção prevista para a semana, mais uma percentagem para desperdício interno.

3. Setup das Máquinas

O setup ou programação das máquinas é feita pelo técnico de manutenção, através de um interface próprio de cada máquina.

4. Arranque da Produção dum Novo Artigo

O arranque é feito pelo técnico de manutenção. Este acompanha o operário de máquina e das operárias de revista para controlarem as primeiras peças, logo após o arranque.

5. Sequência de Produção

Cada máquina produz lotes de 10 dúzias de collants (20 dúzias de meia tubular). Entre cada lote a máquina coloca o seu contador a zero. Cada conjunto de 50 dúzias de collants (100 dúzias de meia tubular), do mesmo tipo, é colocado num saco, pelo operário de máquina.

Na produção de meias, a situação é idêntica à anterior com a excepção das quantidades que são todas a dobrar, por exemplo, produz 20 dúzias de meias (40 dúzias de meia tubular).

Quando existe uma mudança de lote do fio texturizado ou uma alteração de produção, sempre que o lote de 10 (20) dúzias não estiver completo, é o operário de máquina que coloca o contador a zero. O operário de máquina retira os lotes de 10 (20) dúzias, das máquinas e coloca-os dentro de um saco. Depois de completar o lote de 50 (100) dúzias, coloca o saco numa zona específica. O operador de abastecimento retira os sacos desta zona específica e coloca-os no stock.

6. Qualidade da Produção

O operário de máquina controla a última meia tubular de cada lote de 10 (20) dúzias. Se encontrar defeito na meia tubular chama o operário de revista. Este utiliza em seguida o seu procedimento habitual (controlo periódico). No fim, o operário de máquina é informado pelo operário de revista de quantas meias tubulares defeituosas se retiraram. O operário de revista desconta essa quantidade no contador da máquina.

O operário de revista percorre sequencialmente todos os teares, várias vezes, controlando a última meia tubular produzida. Se uma meia apresentar defeito vai controlar todas as meias já produzidas, até encontrar a primeira que já não apresenta defeito.

O Operário de revista controla periodicamente três factores:

1) a medição das biqueiras e dos elásticos, por máquina (uma vez por semana)
2) a medição das meias tubulares, por máquina (duas vezes por semana)
3) a pesagem das meias tubulares (uma vez por mês)

7. Paragens da Produção

Dependendo da causa da paragem (tipo de erro) da máquina, o operário de máquina trata do problema, ou chama o técnico de manutenção. Quando existem duas, ou mais, máquinas paradas, é o técnico de manutenção que decide em qual vai intervir primeiro.

8. Manutenção

O coordenador elabora um plano de manutenção preventiva, mecânica e eléctrica.

A Figura 1.5 reflecte toda a sequência de processos que envolvem a produção de um collant ou de uma meia [RODR00].

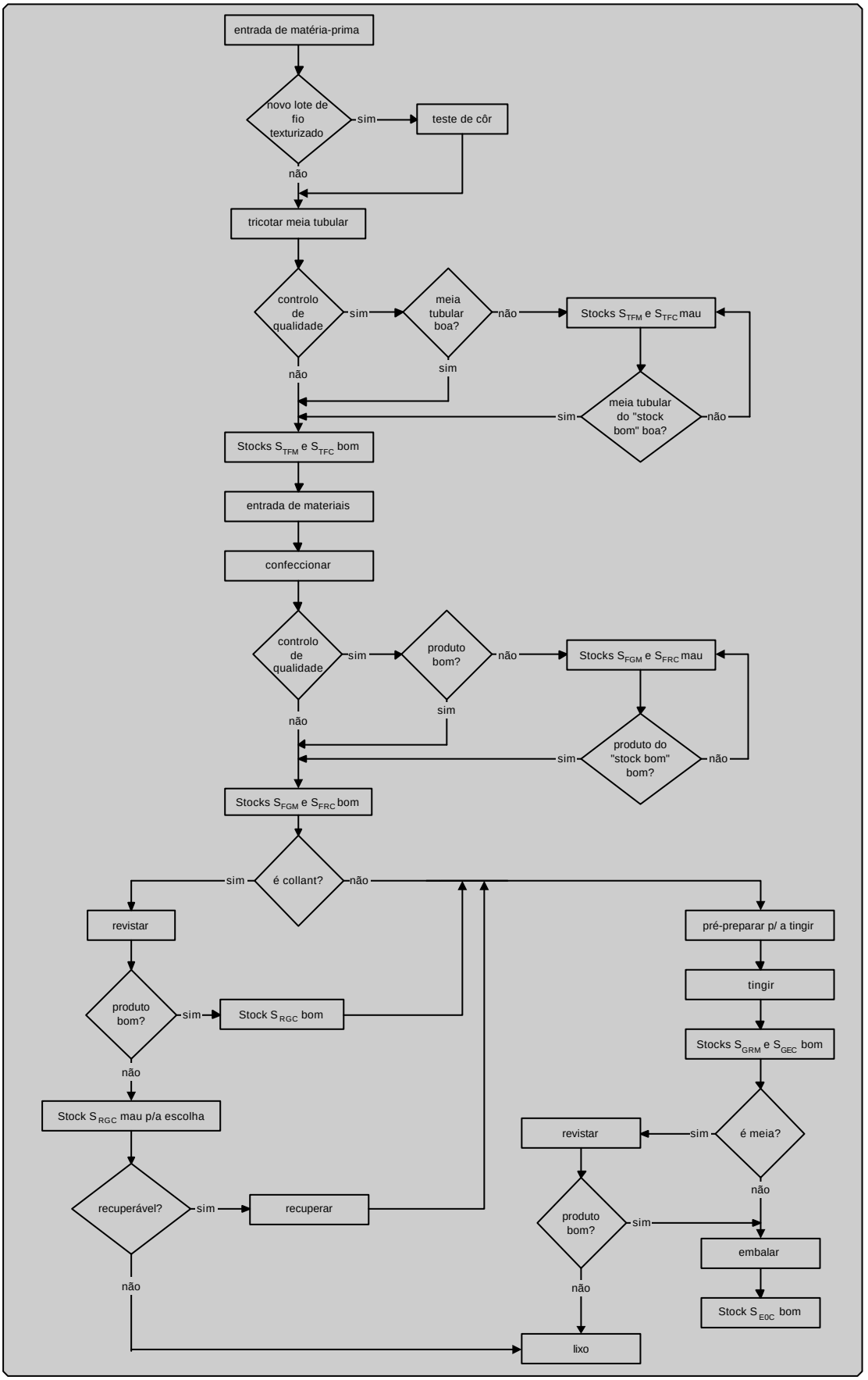


Figura 1.5 –Fluxograma geral de produção [RODR00]

1.2 Objectivos

O principal objectivo é desenvolver um SAD que possibilite a geração automática de modelos de simulação para linhas de produção para a indústria têxtil. A geração automática de modelos de simulação permite elaborar múltiplos cenários de simulação sem ter que redesenhar os modelos.

Este SAD deverá ser dotada de estratégias de controlo que permita ao utilizador o controlo sobre os factores que influenciam o desempenho do sistema.

A ferramenta a desenvolver deveria apresentar as seguintes características:

1. Genérica e flexível quanto à configuração física das linhas de produção e às estratégias de controlo. Facultar o máximo de parametrização do modelo ao utilizador.

2. Proporcionar uma diversidade de estratégias de controlo dos principais índices de desempenho do sistema. Possibilitar ao utilizador a escolha dessas estratégias.

3. Reportar ao estudo o desempenho do modelo e o impacto que a escolha das estratégias de controlo tem no desempenho da produção. Como o sistema é parametrizado pelo utilizador pretende-se que lhe facilite a informação necessária para que possa reajustar as estratégias, de forma a melhorar o desempenho do modelo em testes subsequentes.

A possibilidade de utilização de diversas combinações entre as estratégias dos factores mais relevantes no sistema produtivo é enorme e a capacidade do usar funções estatísticas em algumas parametrizações dificulta a elaboração de um procedimento para conhecer a melhor política[1] para o modelo em estudo.

1.3 Metodologia de Investigação

Este item tem como objectivo descrever sucintamente a organização do processo de investigação conduzido nesta dissertação.

Para garantir um conhecimento minucioso do problema, foi necessário efectuar um estudo aprofundado do ponto de vista funcional ao sistema em questão. Avaliar todos os intervenientes, conhecer as suas funções e distinguir os principais factores e políticas que têm impacto directo ou indirecto na produção de meias. O conhecimento do sistema real é vital para quem pretende executar, em simulação, uma réplica perfeita do sistema real. Esta primeira parte do trabalho teve um objectivo exploratório visando tornar o fenómeno

[1] Conjunto de estratégias aplicadas a um único exemplo

investigado mais claro para que se pudesse estruturar um problema de investigação mais consistente.

A etapa seguinte, a revisão bibliográfica, envolveu a pesquisa, estudo e análise de trabalhos provenientes das áreas da simulação, da produção, da gestão de materiais, de escalonamento e manutenção. Paralelamente, adquiriu-se conhecimentos da linguagem de simulação e de programação que serviu de suporte à ferramenta a elaborar.

O desenvolvimento da ferramenta de apoio à decisão e do *Modelo Base* do sistema, contendo todas as especificidades do problema, foi a fase que se seguiu.

Finalmente, a última fase, a escrita deste relatório, que se pretende ser elucidativo ao nível das ferramentas desenvolvidas, da interacção entre elas e das principais características a elas associadas

1.4 Estrutura da Dissertação

O tema fundamental deste trabalho é a simulação, utilizada para desenvolver modelos que sirvam como ferramenta de apoio à tomada de decisões.

No primeiro capítulo deste trabalho procurou-se fazer um enquadramento do caso de estudo. Descrever a empresa o ambiente produtivo existente, os factores intervenientes na produção e suas funções. É feita uma abordagem aos objectivos e problemas considerados relevantes, sob o ponto de vista científico, que levaram ao desenvolvimento deste trabalho. É delineada a metodologia seguida e a organização dos capítulos.

O referencial teórico, mencionado no segundo capítulo, trata do tema que constitui o principal assunto envolvido na pesquisa: a simulação. Resultando de uma pesquisa sobre a bibliografia existente, descreve-se, neste capítulo, os principais tópicos e os que mais envolvem este assunto.

O terceiro capítulo é dedicado ao desenvolvimento de procedimentos, critérios e estratégias implementadas para a resolução dos problemas apresentados. Foi dada especial relevância à elaboração de uma variedade assinalável de estratégias para cada um dos factores que poderão influenciar os principais índices de desempenho.

Estas estratégias são conceptualmente preparadas e traduzidas em linguagem de programação e simulação. Resultou uma ferramenta informática e um *Modelo Base* de simulação.

O quarto capítulo descreve a geração automática de modelos de simulação e a interacção entre a ferramenta informática e o modelo de simulação. São enunciadas as principais funcionalidades de cada.

A avaliação geral, conclusão e recomendações a trabalhos futuros surgem no quinto capítulo.

Capítulo 2

A Simulação

2.1 Introdução

Desde meados dos anos oitenta que a técnica de simulação tem vindo a ocupar um lugar privilegiado entre as ferramentas de investigação operacional.

A simulação é uma das mais poderosas ferramentas de análise disponível para projecto e operação de processos ou sistemas. A simulação pode ser útil em qualquer uma das fases do ciclo de vida de um sistema de manufactura: desde a fase de análise do problema e definição de requisitos, até as fases de projecto, justificação, implementação e operação.

Porém, apesar de por um lado se reconhecer um enorme potencial na simulação como suporte da tomada de decisões, as dificuldades na aplicação desta técnica na realidade das empresas (modelos custosos de construir e validar, muito pouco flexíveis frente a condições

mais instáveis, e habitualmente concebidos por especialistas e não por reais utilizadores do sistema) atentavam contra a sua efectiva aplicação no âmbito empresarial.

A simulação é uma abordagem de estudo que vem sendo cada vez mais utilizada nas mais variadas áreas de conhecimento. A crescente complexidade dos problemas com que se defronta e a maior disponibilidade de recursos computacionais são dois factores que vêm contribuindo para esse crescimento. Entretanto, é importante comentar que a simulação sempre foi uma técnica extremamente dependente dos recursos computacionais.

As drásticas reduções nos tempos necessários para completar um processo de simulação, passando de projectos que levavam meses, a calendários que podem hoje ser medidos em dias e a possibilidade de serem elaborados pelos próprios responsáveis pelas áreas, sem intervenção dos especialistas de sistemas, permitiram uma transformação na utilização destas técnicas como apoio às decisões.

As primeiras aplicações de simulação foram desenvolvidas em linguagens de programação formais, como FORTRAN. Estas simulações exigiam um enorme esforço de modelação, o que tornava muitas vezes inviável o uso da simulação.

As primeiras linguagens específicas para simulação surgiram na década de 60. Estas linguagens forneciam ao utilizador um conjunto de facilidades para a transformação do modelo formal do sistema num programa computacional, e tornava disponíveis funções e rotinas destinadas a amostragens, análises estatísticas e controle do avanço do tempo na simulação. Embora haja uma simplificação do trabalho de programação, a flexibilidade e a eficiência computacional são em parte sacrificadas. Além disso, o custo de manutenção tende a ser elevado, principalmente pela pouca disponibilidade de pessoal habilitado, em virtude da reduzida difusão destas linguagens. Linguagens construídas dentro deste sistema, pode-se citar o GPSS (*General Purpose Simulation System*), GASP (*Graph Algorithm and Software Package*) e SIMULA.

Estas linguagens responderam à procura por um longo tempo. No entanto, os sistemas ficaram cada vez mais complexos, e tornava-se necessário que, além de nos conduzir a resultados confiáveis, as linguagens de simulação mostrassem às pessoas da produção que os seus benefícios eram reais. Neste ponto, surgiram as animações, que são "softwares" acoplados aos simuladores, capazes de reproduzir os sistemas graficamente. Dessa maneira, ficou muito mais fácil para os analistas, que tinham que mostrar os resultados para o utilizador, e para o próprio utilizador, que compreendia muito mais facilmente a simulação. Este tipo de Software acabou por proporcionar outras vantagens, como formação de pessoal, e maior visualização do sistema produtivo das indústrias. Alguns softwares construídos segundo essa filosofia são o SIMAN/CINEMA e GPSS/H.

Mas a evolução não parou por aí. Apesar do sucesso na indústria norte americana, a simulação necessitava ainda de um tempo de formação muito grande. A construção dos

modelos e animações era demorada, e os analistas precisavam de ter conhecimento do sistema que estavam a simular. Tornou-se necessário assim, que os próprios utilizadores dos modelos de simulação fossem os analistas. Surgiu então uma nova tecnologia de desenvolvimento de aplicações de simulação, chamada VIS (*Visual Interactive Simulation*). Esta tecnologia usa a modelação através de ícones, que agrupam comandos das linguagens tradicionais de simulação, e tornam o trabalho de desenvolvimento mais fácil, com uma interface semelhante à do Windows®, muito mais amigável. Com isso, o tempo de formação para utilizadores reduziu-se drasticamente.

O Papel dos Computadores na Simulação

A utilização da simulação no ambiente empresarial tem-se tornado viável com a evolução da informática. Com o poder computacional dos computadores pessoais a simulação de problemas mais complexos tornou-se computacionalmente possível.

Esse facto impulsionou o avanço das aplicações de simulação que hoje apresentam soluções dos mais variados tipos. Aplicações de simulações podem ser feitas utilizando desde folhas de cálculo até softwares específicos de simulação.

Os softwares com animação gráfica possibilitaram que a simulação se associasse a ferramentas visuais para tomada de decisão. As interfaces mais amigáveis e as linguagens de programação menos complicadas (de mais alto nível) também têm contribuído para a popularização da simulação como uma técnica de apoio à decisão.

Assim, com a consolidação das plataformas gráficas (por exemplo: Windows®, Macintosh®), a simulação começou a recuperar o terreno perdido, constituindo hoje uma ferramenta imprescindível em áreas tais como a investigação, o desenvolvimento de novos produtos, alterações de métodos de fabrico e outros.

2.1 Conceitos de Simulação

Definição de Simulação

Para se entender melhor o que é simulação, precisa-se conhecer também as definições de sistemas e modelos. Um sistema é um conjunto de elementos distintos, que exercem entre si uma interacção ou interdependência. Por natureza, os sistemas são limitados, ou seja, deve-se definir limites ou fronteiras. Portanto, pode-se definir sistemas dentro de outros

sistemas, e assim por diante. Um modelo, segundo Hillier [HILL88], é uma representação de um sistema real, na qual somente os aspectos relevantes para uma determinada análise deste sistema são considerados.

Existe um grande número de definições para simulação. A seguir são apresentadas algumas delas, provenientes de livros clássicos sobre o assunto.

Simulação é o processo de elaborar um modelo de um sistema real e conduzir experiências com esse modelo tendo como propósito a compreensão do comportamento do sistema ou a avaliação de diversas estratégias (dentro dos limites impostos por um critério ou conjunto de critérios) para a operação do sistema [SHAN75].

O processo de projectar um modelo computacional de um sistema real e conduzir experiências, com o propósito de entender seu comportamento e/ou avaliar estratégias para sua operação [PEGD91].

Para Hillier [HILL88], a simulação é nada mais, nada menos, que a técnica de fazer experiências amostrais no modelo de um sistema. As experiências são feitas no modelo, ao invés de no próprio sistema real, porque é mais conveniente e menos dispendioso.

Chase [CHAS89] entende que a melhor forma para definir e entender simulação é considerando-a em duas partes: Primeiro, deve haver um modelo do que quer que seja simulado. Existem várias classificações de modelos, mas os tipos mais comuns são: físicos (modelo de avião), esquemáticos (diagramas de circuitos eléctricos), e simbólicos (programa de computador ou modelo matemático que represente um funcionário bancário). Na simulação computacional, está-se particularmente interessado nos modelos simbólicos, que se usa para representar um sistema real num computador. O principal ponto que se tem que considerar aqui, é que um modelo é criado para representar alguma coisa, e é estático, isto é, mostra apenas um instante no tempo e não muda.

A segunda parte a ser considerada, é mover o modelo ao longo do tempo. Simulação traz *vida* ao modelo. Na formação de pilotos, por exemplo, o discente fica numa cabina completa (modelo de um avião real). Este é um exemplo de um modelo físico. O discente então passa por uma variedade de situações, na medida em que o modelo *vive* e move-se ao longo do tempo. Os parâmetros dos instrumentos do modelo variam e o discente deve responder às indicações. Estas respostas são levadas a um computador, que cria novos valores aos quais a formação deve responder novamente. Desta maneira, a formação pode experimentar várias manobras possíveis, e viver as suas consequências. Assim a simulação é uma série de acções do modelo, com reacções do ambiente.

A simulação de um sistema ou de um organismo é a operação de um modelo ou simulador que é uma representação deste sistema ou organismo. O modelo é sensível a manipulações que seriam impossíveis, muito caras ou de execução impraticável nas entidades

que representam. A operação do modelo pode ser estudada e, a partir daí, propriedades relacionadas com o comportamento do sistema real, ou dos seus subsistemas, podem ser inferidas [NAYL66].

É a técnica de resolver problemas seguindo as variações ocorridas ao longo do tempo num modelo dinâmico do sistema [GORD78]

Além de auxiliar na tomada de decisão, é importante enfatizar a contribuição da simulação para a compreensão do sistema estudado pois, como afirma D. Knuth [KNUT69], "...frequentemente nos enganamos, pensando saber mais do que realmente se sabe sobre uma coisa, até que se tenta simulá-la num computador". Através da simulação o utilizador pode comparar os seus resultados com os do sistema e validar os seus próprios processos de raciocínio.

Segundo G. Doukidis [DOUK87] a função primária de um modelo de simulação é examinar como o sistema se comporta durante um período de tempo. Para atingir este objectivo, o modelo deve providenciar facilidades, para representar o estado actual do sistema, e várias pré-condições que, se satisfeitas, irão resultar num estado futuro.

A simulação de sistemas é, portanto, uma metodologia experimental que busca descrever o comportamento de um sistema. Esta metodologia constrói formas de quantificar o comportamento observado, prevendo o comportamento futuro. A proposta da simulação é produzir dados (e ela é uma grande geradora de números) que, quando analisados, identificarão importantes aspectos do sistema estudado, auxiliando na explicação, compreensão e melhoria do mesmo.

O Processo de Simulação

Entende-se por processo de simulação o desenvolvimento de um modelo de simulação, a sua experimentação do mesmo e a implementação dos resultados.

O desenvolvimento de um processo de simulação é mais uma extensão das artes do que das ciências [SHAN75]. Esta característica talvez explique a dificuldade de se apresentar um mecanismo sistemático para que um utilizador interessado possa desenvolver simulações. Apesar das dificuldades, pode-se estabelecer os elementos que participam num modelo de simulação e alguns passos que, se seguidos, podem ajudar um *modelador*, mesmo principiante, a ter um bom desempenho no desenvolvimento de simulações.

Todos os modelos de simulação possuem, de forma combinada ou isolada, os seguintes elementos [SHAN75]:

- **Componentes**: são as partes (ou subsistemas) integrantes do sistema. Entende-se por sistema, um conjunto de objectos, que interagem entre si, para atingir um objectivo

comum.

- **Parâmetros e Variáveis**: são elementos do sistema que recebem valores. Os parâmetros podem receber valores aleatórios, enquanto que as variáveis recebem valores associados à função à qual estão ligadas. Existem dois tipos de variáveis: Endógenas (Dependentes) e Exógenas (Independentes). Variáveis endógenas são aquelas produzidas dentro do sistema ou resultantes de causas internas. São também chamadas variáveis de estado (pois mostram o estado do sistema) ou variáveis de saída (pois são responsáveis por gerar e apresentar os resultados oriundos do sistema). Variáveis exógenas, também chamadas variáveis de entrada, são originárias de (ou produzidas por) causas externas.
- **Relações Funcionais**: são normalmente apresentadas na forma de equações matemáticas, que relacionam as variáveis endógenas com as exógenas. Essas relações podem ser de ordem determinística (onde para uma dada entrada existe uma única saída) ou estocástica (onde para uma dada entrada existe(m) incerteza(s) associada(s) à saída).
- **Restrições**: são limitações, impostas pelo *modelador* ou pela natureza do problema, que restringem os valores das variáveis.
- **Objectivos**: é o estabelecimento das metas do sistema e como elas podem ser avaliadas. A manipulação do modelo é orientada de forma a satisfazer esses objectivos.

Terminologia Utilizada na Simulação

A terminologia utilizada na simulação não é única, mas há uma certa tendência geral em aceitar a que é apresentada a seguir.

- **Modelo**: é a representação de um sistema.
- **Entidade**: também chamada de transacção, é um elemento essencial para o modelo. Cada tipo de entidade (pessoa ou objecto) possui um ciclo de vida, onde estados activos e passivos se alternam. Uma entidade pode ser classificada como temporária ou permanente. As entidades temporárias entram no sistema, percorrem o seu ciclo de vida e abandonam o sistema. As entidades permanentes executam a sua função sem abandonar o sistema. A chegada de entidades ao sistema é gerada por um procedimento externo, ou de acordo com a necessidade imposta pelo estado do sistema. Na visão da simulação discreta convencional, as entidades temporárias são criadas primeiro. Elas acoplam actividades e requisitam recursos. Uma vez terminada a actividade, a entidade é removida. Trata-se de uma transacção passiva, onde nenhum mecanismo considerado *inteligente* é assumido ou modelado.

- **Atributo**: são as propriedades características de cada entidade.
- **Actividade**: A actividade corresponde a um estado activo, comum a uma ou mais entidades (ou classe de entidades). A duração de uma actividade pode ser determinística ou estocástica. Assume-se que uma actividade é indivisível. Uma vez iniciada, ela não é mais interrompida. Segundo R.Reddy [REDD86], as actividades podem ser físicas ou cognitivas. As actividades físicas são as formas convencionais de actividades, em torno das quais a simulação tem sido desenvolvida. As actividades cognitivas são formas de tomada de decisão racional e inteligente. Da mesma forma que as actividades físicas, as actividades cognitivas possuem tempo finito de duração, que pode ser aleatório ou dependente das informações e/ou decisões do sistema.
- **Acontecimento**: as actividades são iniciadas e terminadas por acontecimentos. Acontecimentos são instantes no tempo, enquanto actividades possuem durações no tempo. Acontecimento é o ponto no tempo no qual acontece alguma mudança no sistema modelado. O processamento de um acontecimento é realizado por uma rotina. Normalmente a ocorrência de um acontecimento gera a ocorrência de outro(s) acontecimento(s). Da mesma forma que as actividades, os acontecimentos podem ser físicos ou cognitivos. As rotinas associadas aos acontecimentos físicos são análogas às rotinas dos acontecimentos tradicionais da simulação discreta. Os acontecimentos cognitivos envolvem processamento de conhecimento, análogo ao procedimento cognitivo desempenhado por quem decide. Rotinas de acontecimentos cognitivos irão conter conhecimentos, representados por regras de produção e heurísticas, que serão processados quando uma decisão precisar ser tomada [REDD86].
- **Contadores**: são variáveis que permitem medir o desempenho do sistema. Possuem esse nome porque vão acumulando valores no tempo, para depois receberem tratamento de análise e gerar as estatísticas da simulação.
- **Relógio de simulação**: variável que marca o tempo da simulação.
- **Variáveis de Estado**: é o conjunto de variáveis que identificam o estado do sistema num determinado instante de tempo.
- **Recurso**: o termo recurso é, basicamente, definido como um lugar em que a entidade fica durante determinado tempo, seja para processamento, espera para ser libertado de uma fila, etc. Um recurso pode ser uma máquina ou um funcionário.
- **Filas**: constituem locais de espera onde as entidades dinâmicas esperam a sua vez de seguir através do sistema. As filas podem ser chamadas de áreas de espera ou pulmões (*buffers*). Depois de dar entrada na fila a entidade é retirada seguindo algum tipo de critério, por exemplo, FIFO (*First In First Out*), LIFO (*Last In Last Out*), HVF (*High Value First*), LVF (*Lower Value First*) ou um critério aleatório [COST02].

Classificação de Sistema de Simulação

Define-se sistema como um grupo ou conjunto de objectos unidos através de alguma forma de interacção ou interdependência. Aqui, o objecto é associado à ideia de *ente*, *entidade.*

Quando se diz que se vai estudar um sistema temos que defini-lo. Se como o sistema é uma parte da realidade, deve-se definir as suas fronteiras. O que fica fora da fronteira do sistema chama-se Meio Ambiente. Surge assim a primeira forma de classificar os sistemas, segundo a sua interacção com o meio ambiente:

- **Aberto**: o meio ambiente afecta o sistema, o comportamento normal do sistema é afectado pelas características do meio.
- **Fechado**: o meio ambiente não afecta o sistema, o comportamento do sistema não é função do meio em que está.

Outro ponto de vista para classificar os sistemas é pela forma como é alterado:

- **Continuo**: segundo alguns autores são contínuos os sistemas em que as alterações são suaves. Na realidade o que interessa é como se comportam as variáveis que medem o estado do sistema no tempo. Diz-se que um sistema é contínuo quando as variáveis que determinam o seu estado podem variar em cada instante ou unidade de tempo.
- **Discreto**: as alterações no sistema são descontínuas, por saltos, súbitas. São sistemas cujas alterações se produzem em instantes de tempo determinados, entre os quais o sistema permanece sem variações. Isto é, as variáveis que definem o seu estado alteram de forma instantânea.

Existem poucos sistemas completamente contínuos ou exclusivamente discretos, mas em geral predomina uma das formas.

Outra forma de classificar os sistemas é pela existência de alterações aleatórias dentro deles:

- **Determinísticos**: as mudanças produzem um único resultado, o comportamento do sistema está determinado
- **Estocásticos**: as alterações produzem resultados aleatórios mais ou menos previsíveis.

Uma Metodologia para Desenvolver Simulações

Ao desenvolver um projecto usando a simulação pode-se distinguir as seguintes etapas [BANK96]:

- **Formulação do Problema**: Neste passo deve-se estabelecer exactamente o objectivo da simulação. O cliente e o *modelador* devem detalhar o mais possível os seguintes factores: os resultados que se esperam da simulação, o plano de experiências, o tempo

disponível, as variáveis com interesse, o tipo de perturbações a estudar, o tratamento estatístico do resultado, a complexidade do interface do simulador, saber se o utilizador terá opção de introdução de dados ou de critérios para a simulação ou se apenas receberá os resultados e, finalmente, saber se o utilizador necessita de um trabalho de simulação ou de optimização.

- **Definição do Sistema**: O sistema a simular deve estar perfeitamente definido. O cliente e o *modelador* devem estabelecer a fronteira do sistema a estudar e as interacções com o meio ambiente que serão consideradas.
- **Formulação do Modelo**: Esta etapa é uma arte. Começa com o desenvolvimento de um modelo simples que capta os aspectos relevantes do sistema real. Os aspectos relevantes do sistema dependem da formulação do problema. Este modelo simples será enriquecido com o resultado de várias iterações. A formulação do problema é uma etapa vital para todo o processo de simulação. Deve ser feita com muito cuidado e competência. Nesta fase cada informação relevante deve ser analisada a fim de definir o problema ao nível de objectivos, restrições e complexidade.
- **Recolha de Dados**: A natureza e quantidade de dados necessários estão determinados pela formulação do problema e do modelo. Os dados podem ser fornecidos por registos históricos, experiências em laboratório, ou medições realizadas no sistema real. Estes devem ser processados adequadamente para terem o formato exigido pelo modelo.
- **Implementação do modelo no computador**: O desenvolvimento do modelo vai depender da ferramenta computacional escolhida. O resultado poderá ser um simulador tradicional ou um complexo sistema de simulação. Nessa fase são criados os códigos de simulação, por geração automática ou não. A dificuldade, e consequente velocidade no desenvolvimento do simulador, estará associada tanto à ferramenta computacional utilizada, como à complexidade do modelo e experiência do *modelador*. Existem linguagens específicas de simulação que facilitam esta tarefa.
- **Verificação**: é a etapa onde o *modelador* confirma se o modelo desenvolvido corresponde ao idealizado. Neste ponto é verificado se o modelo foi construído correctamente. Nesta fase procura-se fazer testes exaustivos no simulador. O *modelador* precisa ficar convencido de que o simulador está correcto e a funcionar bem. Na verificação de um modelo devem-se variar os valores dos parâmetros de entrada (inclusive utilizando as fronteiras do intervalo de valores) e analisar se os resultados são coerentes. Nesse momento é possível sentir a força da simulação, que agiliza a etapa de testes, tornando-os viáveis em termos de tempo e dinheiro.
- **Validação**: é a etapa onde será verificado se o modelo desenvolvido representa bem o sistema real. É a procura da resposta para a pergunta: foi desenvolvido o modelo correcto? A ideia é passar confiança ao utilizador, mostrando que qualquer experiência

com o modelo irá gerar resultados que se coadunam com a realidade do sistema estudado. A validação é normalmente conseguida executando o modelo e comparando os seus resultados com os oriundos do sistema real. Se os resultados da simulação se aproximarem dos valores reais, dentro de um nível de confiança desejado, o simulador será validado. Segundo Annino [ANNI81] a técnica de validação mais efectiva é apresentar o programa a alguém que conheça muito bem o sistema em estudo. Num esquema interactivo entre o especialista no funcionamento do sistema real e o *modelador*, será mais fácil corrigir desvios do modelo em relação ao sistema real. A Validação do modelo é extremamente importante pois os simuladores tendem a parecer reais e tanto o *modelador* como o utilizador, passam a acreditar nele.

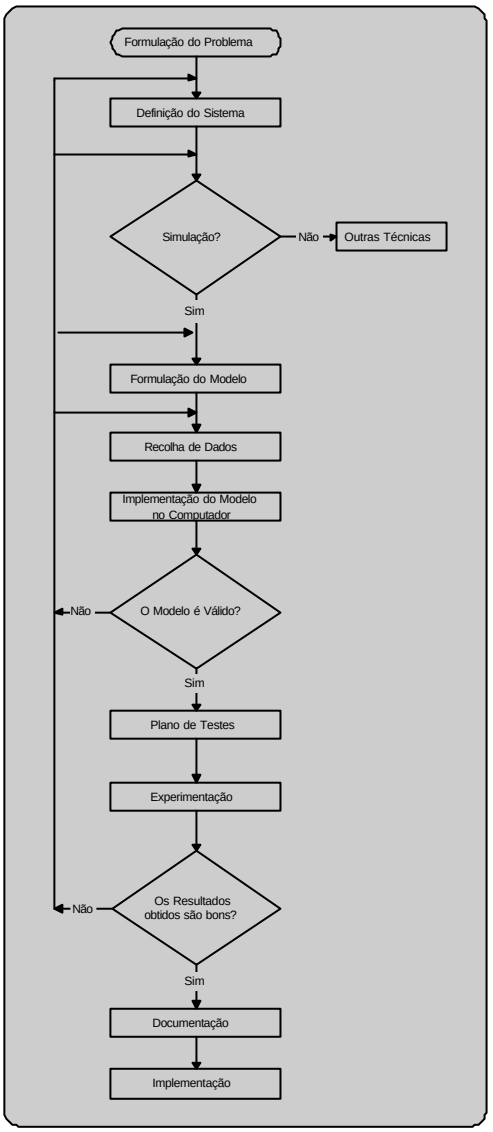


Figura 2.1 – Uma metodologia para desenvolver simulações

- **Plano de testes**: É importante projectar as experiências visando alcançar os objectivos estabelecidos. Portanto deve-se combinar os valores dos parâmetros que *optimizarão* as variáveis de resposta, e explicar as relações entre as variáveis de respostas e os factores controláveis do sistema. É a fase que consiste em executar o modelo computacional sob os diversos cenários de simulação estabelecidos e onde se decide quando começa a simulação, qual o tempo de simulação e o número de simulações. É bom lembrar que a simulação, a partir de situações iniciais diferentes, oferece linhas de acção alternativas para o agente de decisão. E como a fase de execução do modelo é, regra geral, extremamente rápida, pode-se abusar do número de cenários a serem simulados.
- **Experimentação**: Nenhum projecto de simulação pode ser considerado completo se não for aceite, entendido e utilizado. Para isso devem ser definidas as condições iniciais, e como elas afectam o equilíbrio do sistema. O pano de fundo desta fase é a necessidade de diminuir a variância das respostas e, ao mesmo tempo, procurar

minimizar o tamanho das amostras necessárias. Nesta etapa realizam-se as simulações de acordo com o plano de testes e, o resultado é o estabelecimento dos diversos cenários que serão simulados.

- **Interpretação**: Analisa a sensibilidade do modelo relativamente aos parâmetros que tenham associado maior grau de incerteza. É uma fase crítica, demorada, difícil e muito importante. O resultado de uma simulação é uma avalanche de números, que precisam ser tratados e analisados.
- **Implementação**: O pouco tempo gasto na implementação normalmente não é suficiente para as tarefas existentes nessa fase. Fazem parte dessas tarefas: lapidação e ajuste do modelo, formação do utilizador e a garantia da validade dos resultados. Esta última tarefa, que só é possível com a implementação, tende a ser, segundo R.Shannon [SHAN75] o problema mais difícil a ser enfrentado pelo analista.
- **Documentação**: Inclui a elaboração de documentação técnica e manuais de utilização. A documentação técnica deve conter uma descrição detalhada do modelo e dos dados, também deve incluir a evolução histórica das distintas etapas de desenvolvimento. Esta documentação será muito útil, pois além de facilitar a implementação e a possível necessidade de alterações no modelo, auxilia o *modelador* em futuros projectos, podendo forçá-lo a questionar os seus próprios procedimentos.

Causas de Insucesso no Desenvolvimento de Simulações

- Falha na obtenção de um conjunto bem definido de objectivos no início do estudo da simulação
- Inapropriado nível de detalhe
- Falha de comunicação com o responsável do sistema a ser simulado durante o estudo da simulação
- Interpretações equivocas por parte da equipa da simulação do sistema a ser simulado
- Falha de compreensão da simulação por parte da administração
- Software e/ou linguagem imprópria
- Modelos inválidos (simulação não representa a realidade)
- Maus geradores de números aleatórios
- Software de simulação muito complexo e com documentação inadequada
- Semente inapropriada
- Tempo de simulação inadequado
- Uso inadequado da animação

- Uso de distribuições incorrectas, isto é, que não correspondem ao comportamento real
- Uso de medidas de desempenho inadequadas
- Executar uma única vez a simulação e considerar os dados obtidos como a resposta verdadeira.

Vantagens da Simulação

- A modelação de sistemas reais obriga a organização a entender o papel de cada componente do sistema e as possíveis interacções entre eles.
- O desenvolvimento do modelo de simulação ajuda a organização a separar os parâmetros controláveis daqueles que não o são e estudar a influência de cada um deles no sistema.
- O resultado da simulação permite que a gerência avalie os recursos necessários, ou como os recursos disponíveis devem ser alocados.
- É uma técnica flexível com relação às limitações impostas aos modelos.
- Aplica-se à análise de problemas de grande escala e complexidade que não podem ser resolvidos por técnicas tradicionais de gestão de operações.
- Particularidades da situação real podem ser consideradas, como por exemplo a utilização de qualquer curva de probabilidade que o problema exija, e que seja diferente do padrão assumido por técnicas analíticas.
- Sistemas com longos períodos podem ser analisados num curto espaço de tempo.
- Permite a análise de sensibilidade do tipo " que aconteceria se ... ", Várias políticas de decisão podem ser testadas e comparadas rapidamente.
- A simulação possibilita o estudo individual de cada componente ou variável do modelo para determinar qual é realmente importante.
- Possibilita antever os possíveis problemas que ocorrem na altura da implantação de um sistema real
- Para aqueles problemas que na prática são resolvidos por regras intuitivas (*rules of thumb*) a simulação é uma ferramenta forte para o apoio à decisão permitindo que soluções potencialmente boas sejam encontradas. [LOBA95]

Regra geral, é apropriada quando:

- desenvolver um modelo matemático é muito difícil ou até impossível
- o sistema tem uma ou mais variáveis aleatórias relacionadas
- a dinâmica do sistema é muito complexa

- o objectivo é observar o comportamento do sistema num determinado período
- é necessário e importante mostrar animação gráfica
- a construção de protótipos para verificação da correcção do projecto ou avaliação de propriedades tem custo proibitivo.

Desvantagens da Simulação

- Um bom modelo de simulação pode tornar-se caro e levar vários meses para desenvolvimento, especialmente quando os dados são de difícil obtenção.
- A simulação não gera bons resultados sem entradas (*inputs)* adequados. A construção e a alimentação do modelo requerem um trabalho árduo e criterioso.
- Cada modelo de simulação é único. Geralmente não é possível a utilização de um modelo em diferentes situações, prejudicando a possibilidade de ganhos de escala.
- Apesar dos novos softwares de simulação possuírem ferramentas adequadas que podem ajudar na obtenção de bons resultados, a simulação não é uma técnica exacta.

2.2 Modelos

No encaminhamento de um estudo de simulação, uma das principais etapas consiste na modelação do sistema sob estudo, para que se possa observar o seu comportamento sob determinadas condições, de forma a, cientificamente, estudá-lo e entendê-lo. Neste processo, procura-se imitar e criar uma história artificial da actuação e desempenho do sistema real, o que implica a realização de um procedimento experimental, posterior à etapa de modelação.

A modelação de um sistema dependerá, fundamentalmente, do propósito e da complexidade do sistema sob investigação. São vários os tipos de modelos que podem ser usados, tais como modelos matemáticos, modelos descritivos, modelos estatísticos e modelos do tipo entrada-saída.

A decisão sobre o uso de modelos descritivos, matemáticos, estatísticos, etc., ao contrário de modelos voltados para a simulação do sistema, depende de diversos factores. Se o sistema no qual se tem interesse for suficientemente simples, isto é, se as inter-relações entre os seus elementos podem ser bem descritas pelos primeiros, então, o uso do cálculo, da álgebra ou da teoria de filas, por exemplo, podem trazer resultados e respostas aceitáveis e, muitas vezes exactas. No entanto, os sistemas do mundo real costumam ser mais complexos do que o desejado e, acima de tudo, não apresentam um comportamento previsível.

Simplificações sobre estes sistemas objectivando estudos analíticos, podem levar a soluções pobres e, pouco confiáveis. Neste caso, um modelo voltado à simulação do sistema

pode ser a decisão mais correcta.

Classificação dos Modelos de Simulação

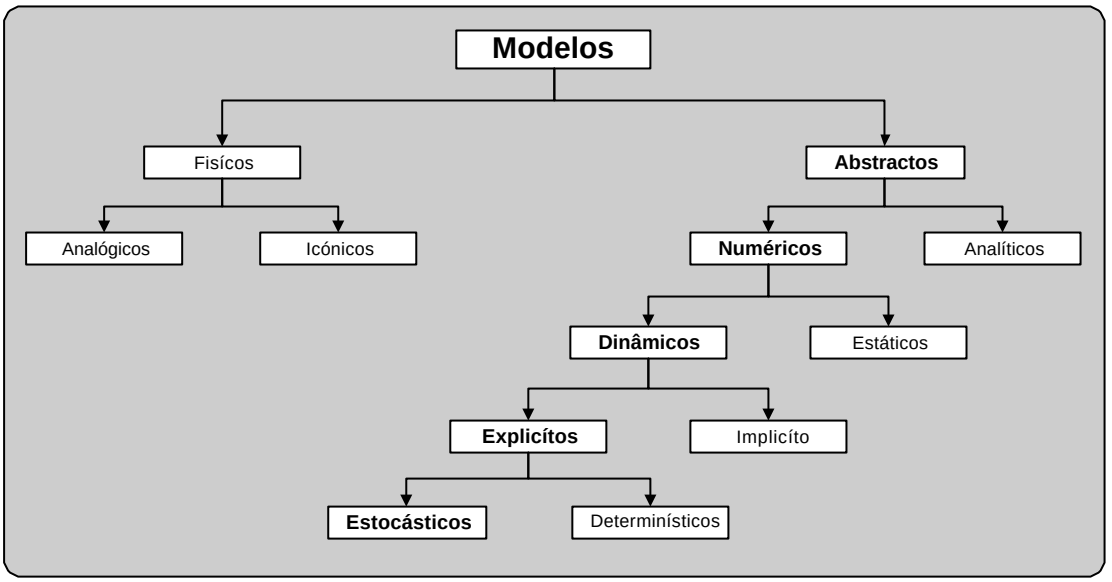


Figura 2.2 – Esquema da classificação dos modelos

Existem várias formas de se classificar modelos. A seguir, é reproduzida uma abordagem sintética sobre modelos segundo [RODR96]:
Quanto à sua Natureza, os modelos podem ser classificados como:

- **Físicos**
  - **Icónicos**: Nos quais existe apenas uma modificação de escala em relação aos sistemas reais. São deste tipo os modelos de aviões e carros ensaiados em túneis de vento.
  - **Analógicos**: Modelos em que uma determinada grandeza física é utilizada para obter conclusões sobre propriedades do sistema real. É deste tipo um modelo implementado num computador analógico, onde se utilizam tensões para, por exemplo, descrever o movimento da queda de um corpo.
- **Abstractos**
  - **Analíticos**: São modelos expressos através de formas analíticas. Estes modelos assumem a possibilidade de quantificar todas as variáveis fundamentais do sistema e também o conhecimento de expressões que definem a dependência relativa dessas variáveis. Inclui-se nesta classificação, por exemplo, o Modelo de Programação Linear.

- **Numéricos**: Estes modelos lidam com valores numéricos, mas sem que se conheçam equações que relacionem as variáveis do sistema. È nesta classificação que se inclui o tipo de simulação que se irá descrever.

A SIMULAÇÃO LIDA COM MODELOS ABSTRACTOS E NUMÉRICOS

Quanto à sua evolução no tempo podem classificar-se como:

- **Estáticos**: Modelos que não se modificam com o tempo. Assim, por exemplo, um Modelo de Programação Matemática pode ser parametrizado, mas as relações expressas que definem o modelo não se alteram com o tempo.
- **Dinâmicos**: Representam a forma como o tempo modifica o sistema real. È neste tipo de classificação que se inclui o tipo de modelos que se utiliza em simulação.

A SIMULAÇÃO LIDA COM MODELOS DINÂMICOS

Quanto à forma como representa o comportamento interno do sistema, podem ser classificados como:

- **Implícitos**: São modelos do tipo *input-output.* Incluem-se nesta classificação os modelos utilizados em simulação normalmente designados por *jogos de gestão*, em que após submeterem um conjunto de decisões, os participantes obtêm informação sobre medidas de desempenho das respectivas empresas.
- **Explícitos**: Nestes modelos, o comportamento interno é detalhado, sendo a observação deste funcionamento um dos objectivos a cumprir com o modelo.

A SIMULAÇÃO LIDA COM MODELOS EXPLÍCITOS

Quanto aos valores que as variáveis que definem o modelo podem assumir, os modelos podem classificar-se em:

- **Determinísticos**: Os valores dos parâmetros de funcionamento do sistema são constantes em cada utilização do modelo sendo, também, perfeitamente determinado o resultado da aplicação do modelo. Assim, por exemplo, um Modelo de Transportes produzirá sempre a mesma solução para um conjunto de valores de disponibilidade, procura e custos unitários de transporte.
- **Estocásticos**: São modelos que integram na definição do seu funcionamento um elemento aleatório. Assim, o valor dos parâmetros de funcionamento do sistema não serão determinísticos mas sim valores de variáveis aleatórias provenientes quer de

distribuições teóricas, quer de distribuições empíricas (definidas por histogramas de probabilidades).

A SIMULAÇÃO LIDA COM MODELOS ESTOCÁSTICOS

Uma outra classificação é apresentada graficamente por [LEEM99]:

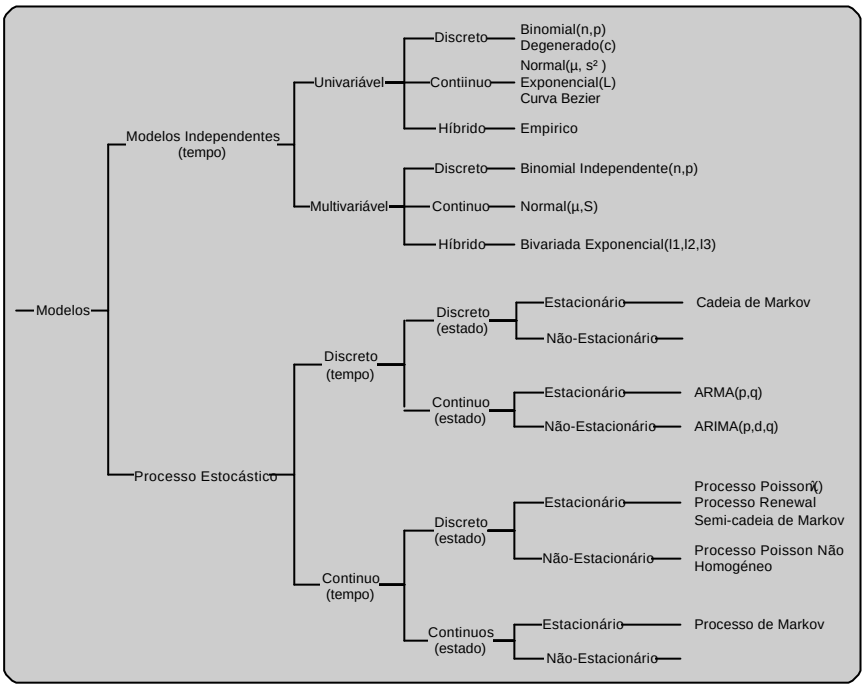


Figura 2.3 – Uma taxinomia para entradas de modelos

O modelo de simulação pode operar em tempo real ou em tempo simulado. Em tempo real a escala de tempo é a real, isto é os acontecimentos ocorrem e são tratados na mesma escala de tempo correspondente ao sistema real. Neste caso o operador interage com o simulador em tempo real.

A operação em simulado não acompanha a escala de evolução do tempo real. Um ano do tempo de simulação pode decorrer em poucos segundos de processamento.

2.3 Simulação Discreta

Um modelo descreve a composição dinâmica do sistema, ou seja, a maneira como este executa um determinado trabalho. Três componentes (actividades, processos e acontecimentos) são utilizados para descrever o comportamento dinâmico de sistemas discretos e sobre os quais as linguagens de simulação para esses sistemas são baseados. Um sistema é visto dinamicamente como uma colecção de processos interactivos, cada um

composto por diversas actividades, com as interacções controladas pela ocorrência de acontecimentos [MACD87, [SOA92]. A relação entre essas três componentes pode ser melhor entendida observando a Figura 2.4.

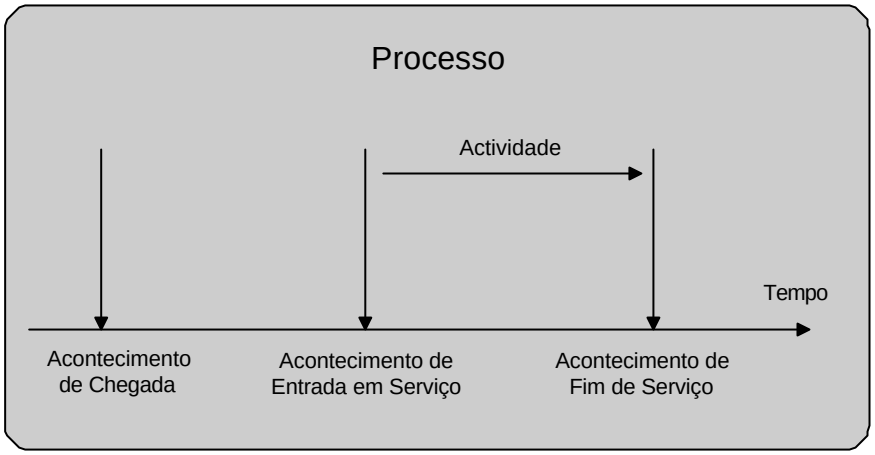


Figura 2.4 – Relação entre acontecimento, actividade e processo

Um conjunto de actividades logicamente relacionadas constitui um processo. O tempo de execução de um processo é a soma dos tempos de execução e espera das actividades. A definição dos processos de uma simulação depende do nível de abstracção adoptado. Por exemplo, um programa inteiro pode ser visto como um processo (compreendendo actividades de computação, leitura e escrita de dados). Num outro nível, a execução de uma actividade de escrita pode ser vista como um processo compreendendo posicionamento, latência e transferência de dados. Um acontecimento causa uma mudança de estado de alguma entidade do sistema. A inicialização ou término de uma actividade são acontecimentos [MACD87].

Filosofias de Modelação

Filosofia de modelação é a forma como os modelos de simulação podem ser estruturados. Um modelo de simulação discreta pode ser estruturado segundo os seguintes métodos:

- **orientado ao acontecimento** (*Event-Driven Simulation*): o sistema é modelado pela definição das mudanças que ocorrem no tempo de acontecimento;
- **orientado à actividade** (*Activity-Driven Simulation*): O sistema é modelado através da descrição das actividades nas quais os objectos do sistema se envolvem;
- **orientado ao processo** (*Process-Driven Simulation*): o sistema é modelado por meio da descrição dos processos através dos quais os objectos flúem.

- **orientado pelo método das três fases** (*Three Phase-Driven Simulation*): combina a simplicidade conceptual dos modelos por actividades e a eficiência do método orientado a acontecimentos
- **orientado pelo tempo** (*Time-Driven Simulation*): o sistema é sempre actualizado em períodos de tempo definidos.
- **orientado por dados** (*Data-Driven Simulation*): actualiza o sistema quando existir toda a informação sobre determinados blocos.

Simulação Orientada ao Acontecimento

Na simulação orientada ao acontecimento, a tarefa do modelador é determinar os acontecimentos que podem causar a mudança no estado do sistema e então desenvolver a lógica associada com cada tipo de acontecimento. A simulação do sistema é produzida pela execução dessa lógica, numa sequência no tempo [SOAR92].

Um mecanismo de escalonamento de acontecimentos mantém a sequência ordenada dos acontecimentos no tempo. Esse mecanismo utiliza uma estrutura tipo *array* ou lista. Quando um acontecimento, como por exemplo a chegada do próximo cliente, está para ser escalonado, o processo escalonador é chamado para criar e adicionar uma entrada na lista de acontecimentos. Nesta entrada é incluído o instante da ocorrência do acontecimento, a identificação do acontecimento que vai ocorrer e a identificação do cliente associado.

Como exemplo, considere o atendimento a clientes, por um funcionário de um banco [MARY80]. Ao chegar ao banco o cliente entra na fila e espera pela sua vez de ser atendido. Após terminar o atendimento, o cliente deixa a agência bancária. Mudanças de estado no sistema ocorrem devido aos seguintes acontecimentos:

- um cliente chega
- um cliente já foi atendido por um funcionário e parte do sistema

A lógica associada aos acontecimentos pode ser descrita, por exemplo, considerando apenas dois acontecimentos:

```
Gerar próxima chegada
Se o funcionário está ocupado
        Então
                Aumentar o número de clientes na fila
        Senão
                Alterar o estado do funcionário para ocupado
                Gerar o fim de atendimento para tempo_actual + tempo_de_atendimento
```

Figura 2.5 – Procedimento chegada de cliente

```
Alterar o estado do funcionário para livre
Se existe algum cliente na fila
        Então
                Alterar o estado do funcionário para ocupado
                Retirar da fila para atendimento
                Gerar o fim de atendimento para tempo_actual + tempo_de_atendimento
```

Figura 2.6 – Procedimento fim de atendimento

O programa de simulação pode ser organizado da seguinte forma:

```
INICIO
        Gerar a primeira chegada
        Seleccionar o primeiro evento
        Avançar o tempo de simulação para o tempo de ocorrência do evento
        Enquanto < fim_de_simulação
                Então
                        Imprimir estatísticas
                Senão
                        Se evento = CHEGADA_DE_CLIENTE
                                Então
                                        Processa CHEGADA_DE_CLIENTE
                                Senão
                                        Processa FIM_DE_ATENDIMENTO
FIM
```

Figura 2.7 – Programa orientado ao acontecimento

Simulação Orientada à Actividade

Na simulação orientada à actividade, o modelador descreve cada actividade possível para cada entidade do sistema, definindo as condições que causam o seu início e o seu fim. Os acontecimentos que iniciam ou terminam uma actividade não são escalonados pelo modelador, mas são iniciados a partir das condições específicas para a actividade. Com o avanço do tempo simulado, as condições para o início ou fim de uma actividade são examinadas e, no caso de alguma condição ser satisfeita, a actividade correspondente é escalonada para a execução no tempo de simulação actual.

Esta orientação é pouco utilizada para a modelação de sistemas [MACD75, MARY80, SOAR92]. No exemplo dos clientes sendo atendidos por funcionários de um banco, tem-se as seguintes actividades:

- Chegada de um cliente (CHEGADA_DE_CLIENTE)
- Fim de atendimento ao cliente (FIM_DE_ATENDIMENTO)

A estrutura básica do programa de simulação é mostrada a seguir:

```
Tempo_de_simulação = 1
Enquanto tempo_de_simulação < fim_de_simulação
        Verificar o conjunto de actividades
        Se chegou um cliente
                Então
                        Processa CHEGADA_DE_CLIENTE
        Se fim de atendimento
                Então
                        Processa FIM_DE_ATENDIMENTO
        tempo_de_simulação = tempo_de_simulação + 1
        Imprimir estatísticas
```

Figura 2.8 – Programa orientada à actividade

Simulação Orientada ao Processo

O programa de simulação é organizado como um conjunto de processos, os quais são executados concorrentemente durante a simulação. A descrição de um processo tem a forma de um procedimento de uma linguagem de programação [MACD87].

Como a orientação a acontecimentos, também existe um mecanismo de escalonamento e uma estrutura de lista, com a diferença de que cada entrada na lista define um processo e o seu ponto de reactivação (para processos que são bloqueados e novamente postos a executar).

Basicamente uma simulação orientada a processo deve seguir os seguintes passos [SPOL94]:

- definir as entidades do sistema
- criar um processo para cada objecto do sistema descrevendo as suas etapas
- executar concorrentemente os processos.

No exemplo dos clientes sendo atendidos por funcionários de uma agencia bancária, descrito na secção anterior, seguindo este tipo de orientação tem-se o processo cliente organizado da seguinte forma:

```
Se funcionário está livre
        Então
                Alterar o estado para ocupado
                O cliente ocupa o funcionário pelo tempo de atendimento
                Gerar processo para tempo_actual + tempo_de_atendimento
                Alterar o estado do funcionário para livre
                O cliente sai do sistema
        Senão
                O cliente entra na fila de espera
```

Figura 2.9 – Procedimento cliente

A chegada de novos clientes ao sistema (agência bancária) pode ser descrita como no processo Chegada de clientes:

```
Indicar chegada de novo cliente
Gerar processo para tempo_actual + tempo_chegada
```

Figura 2.10 – Procedimento processo chegada de clientes

O programa de simulação executa sincronamente vários Processos Cliente que chegam segundo uma distribuição de probabilidade, requisitam o serviço e abandonam o sistema.

Simulação Orientada pelo Método das Três Fases

È uma forma de estruturação de programas de simulação a acontecimentos discretos introduzida por Tocher em 1963. Existem dois tipos de acontecimentos associados ao método das três fases: os acontecimentos que caracterizam o início e aqueles que caracterizam o fim da actividade. Tais acontecimentos são conhecidos como acontecimentos do tipo C (início da actividade) e acontecimentos do tipo B (fim da actividade). Todo o acontecimento C é tido como condicional, pois para uma actividade ter início, uma ou mais condições devem sempre ser satisfeitas. A cada actividade está associado um único acontecimento C, que está condicionado à existência de entidades nas filas e/ou fontes que precedem a actividade. Como a duração de cada actividade é determinada antes do seu início, através de amostragem ou deterministicamente, os acontecimentos

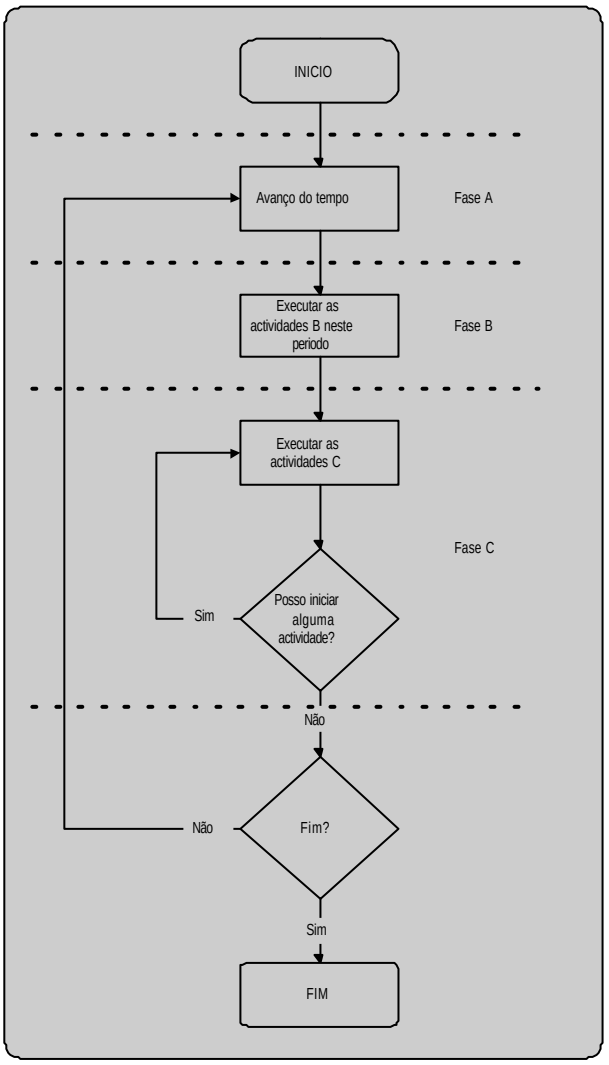


Figura 2.11 – Esquema do método das 3 fases

B têm os seus inícios ou ocorrências predeterminadas. Para cada entidade participante da actividade é definido um acontecimento B distinto. Assim, a cada actividade está associado um único acontecimento C e tantos acontecimentos B quantas forem as entidades que nela participam. O acontecimento B liberta as entidades que participaram na actividade a ele relacionada para as filas subsequentes àquela actividade. As três fases que caracterizam o método são:

Fase A: Corresponde ao avanço do relógio da simulação. Nesta fase o relógio avança para o instante que coincide com o fim da próxima actividade, isto é, o programa consulta a lista de acontecimentos B e encontra aquele que ocorrerá em primeiro lugar.

Fase B: Termina a actividade e executa todos os acontecimentos B programados para aquele momento, libertando as entidades para as filas subsequentes.

Fase C: Executa a verificação das condições de início de todas as actividades do sistema. Inicia as actividades cujas condições foram satisfeitas, calculando as suas durações e programando os acontecimentos B associados a elas. O início de cada actividade é caracterizado pela saída das entidades das filas que a precedem. Uma actividade é iniciada enquanto as suas condições iniciais forem satisfeitas.

A ordem de teste das condições iniciais das actividades segue a ordem das prioridades predefinidas. Quando todas as actividades passíveis são iniciadas termina a fase C e retorna-se à fase A até à interrupção do programa e/ou ao seu fim.

Como pode ser notado no início da simulação, ou seja, quando o relógio está no instante zero, nenhum acontecimento B acontece, pois não existe nenhuma actividade em andamento. Sendo assim, o sistema deve ser modelado de tal forma que no instante zero possa ocorrer o início de pelo menos uma actividade. As entidades permanentes devem ser colocadas em filas adequadas antes do início da simulação. Para as entidades temporárias não existe esta preocupação.

Simulação Orientada pelo Tempo (Time-Driven Simulation)

No modo mais simples de implementação desta técnica, cada bloco do modelo de simulação é chamado uma vez a cada intervalo de avanço do tempo de simulação (*simulation clock*). Ou seja, a cada incremento no tempo de simulação, todos os blocos do modelo são chamados, actualizam os seus estados internos para corresponder ao tempo actual, independentemente da existência ou não de alguma tarefa a ser executada. Isto torna esta técnica bastante ineficiente do ponto de vista computacional. O tempo global de simulação é actualizado através de um incremento constante.

Simulação Orientada por Dados (Data-Driven Simulation)

Nesta técnica de simulação, um bloco só é chamado se todos os dados necessários na sua entrada estiverem disponíveis. Desta forma, se em cada bloco do modelo de simulação for conhecida previamente a disponibilidade de tais dados, é possível a construção de uma sequência de chamada de blocos antes do início da simulação. Caso contrário, a disponibilidade de tais dados deve ser verificada durante a simulação e os blocos deverão ser chamados dinamicamente.

Técnicas de Avanço no Tempo

A execução da simulação pode operar-se em duas formas [BALL96]:

- **Avanço regular ou fixo** (*time slicing*)
- **Avanço para o próximo acontecimento** (*next event*)

Avanço Regular ou Fixo

Inspecciona o estado do sistema através de amostragens sucessivas em intervalos de tempo regulares. A equação seguinte descreve o processo:

$$\text{tempo\_modelo}(t + \Delta t) = \text{tempo\_modelo}(t) + \text{intervalo}(\Delta t)$$

Este método é muito sensível à escolha de um intervalo de tempo apropriado. Se o intervalo de tempo for demasiado pequeno, são necessárias computações extra dentro do modelo. Se o intervalo de tempo for demasiado grande, alguns acontecimentos serão omitidos.

- **Vantagens**: simplicidade de implementação
- **Limitações**: não há sincronismo entre as transições de estado do sistema e o processo de amostragem.
- **Desvantagens**: fraca eficiência, pode-se estar ao longo de muito tempo a recolher amostras do mesmo estado, originando um grande número de amostras supérfluas, sem informação.

Exemplo:

e_i (i=1,2,...) tempo real de ocorrência do acontecimento.

No intervalo [0, Δt] ocorre o acontecimento e_1, que é considerado no modelo como ocorrido no tempo Δt.

No intervalo [Δt, 2Δt] não ocorrem acontecimentos.

No intervalo [2Δt, 3Δt] ocorrem os acontecimentos e_2 e e_3, que se consideram ocorridos no tempo 3Δt.

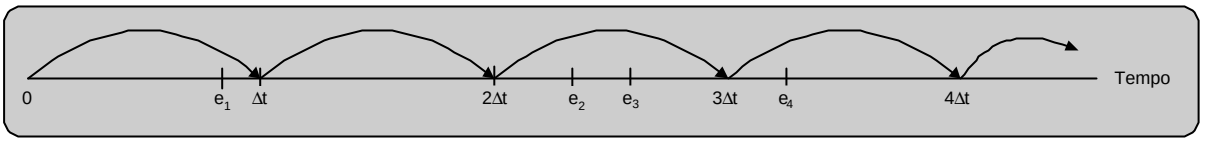


Figura 2.12 – Esquema da técnica do avanço regular

Avanço para o Próximo Acontecimento

O tempo de simulação avança para o próximo acontecimento.

Esta técnica tem a vantagem de ser independente das frequências envolvidas nas mudanças de estado, reduzindo ao mínimo o número de amostras recolhidas na simulação.

A Principal desvantagem é a complexidade de implementação, já que de cada vez que se dá um acontecimento é necessário calcular o tempo em que surgirá o próximo.

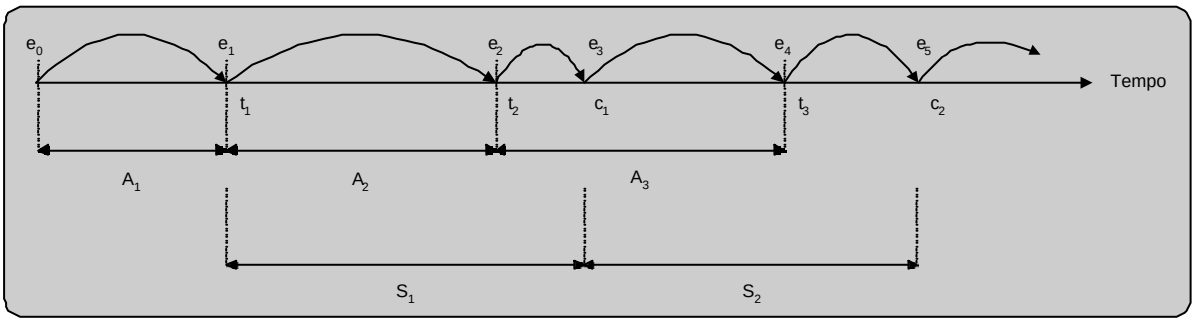


Figura 2.13 – Esquema da Técnica do Avanço para o Próximo Acontecimento

Exemplo:

t_i .....tempo de chegada do enésimo cliente

$A_i = t_i - t_{i-1}$.....tempo entre chegadas de clientes

S_i .....tempo de serviço

d_i.....tempo de espera na fila

$c_i = t_i + d_i + S_i$ .tempo de saída do cliente

e_i.....tempo de ocorrência de um acontecimento

No tempo $e_0=0$, o servidor está livre e o tempo de chegada t_1 do primeiro cliente é calculado como $A_1+ e_0$.

O relógio de simulação é avançado de e_0 até o próximo acontecimento $e_1=t_1$.

Chegada do cliente, servidor está livre, então $d_1=0$. É calculado o tempo de serviço S_1 e $c_1=t_1+S_1$. É calculado o tempo de chegada do próximo cliente $t_2=t_1+A_2$.

Se $t_2<c_1$, o relógio de simulação é avançado de e_1 até o tempo do próximo acontecimento $e_2=t_2$.

Se $c_1<t_2$, o relógio seria avançado até $e_2=c_1$.

Chegada do novo cliente no tempo t_2, servidor ocupado, cliente fica na fila. É calculado o tempo de chegada do próximo cliente $t_3=t_2+A_3$.

Se $c_1<t_3$, o relógio de simulação é avançado de e_2 para o tempo do próximo acontecimento $e_3=c_1$.

Cliente completa o serviço, servidor fica livre, primeiro cliente da fila entra no servidor e o seu tempo de espera é calculado $d_2=c_1.t_2$. Tempo de serviço $c_2=c_1+S_2$ é calculado.

Se $t_3<c_2$, o relógio é avançado para o próximo acontecimento $e_4=t_3$,...

2.3 Ferramentas e Linguagens de Simulação

O intenso uso de simulação como abordagem de desempenho de sistemas deu origem a uma série de ferramentas específicas projectadas para este fim. Estas diversas linguagens e ferramentas de modelação impõem uma certa estruturação e simplificam as soluções [SOAR92].

Um requisito importante para as linguagens de simulação é a presença de mecanismos para a representação do tempo. Enquanto o sistema modelado executa em tempo real, a simulação trabalha com um relógio próprio, que marca a passagem do tempo no programa de simulação [MACD75]. Outro requisito importante é a facilidade com que se obtém informações estatísticas e relatórios. Para a implementação de um modelo de simulação pode-se usar as seguintes linguagens de programação:

- linguagens de programação convencionais;
- linguagens de simulação;
- ferramentas de uso específico;
- extensões funcionais.

Linguagens de Programação Convencionais

Em princípio, toda a linguagem de programação é uma candidata a linguagem de simulação [PEGD91]. Porém, para desenvolver um programa de simulação convencional, o programador deve criar todo um ambiente necessário para a simulação pois estas linguagens não oferecem todas as ferramentas necessárias para um ambiente de simulação como por exemplo, tratamento estatístico e emissão de relatórios.

A desvantagem de se utilizar linguagens convencionais é que o programador deve ter conhecimento tanto da linguagem como de simulação para poder criar o ambiente de simulação.

Uma vantagem é que o programador não necessita aprender uma nova linguagem já que ele tem a liberdade de utilizar uma que já é do seu conhecimento. Esta abordagem oferece ainda uma grande flexibilidade uma vez que o utilizador pode utilizar todos os recursos oferecidos pela linguagem.

Linguagens de Simulação

São linguagens projectadas para a modelação de sistemas de vários tipos [SOAR92]. Estas linguagens já contêm todas as estruturas necessárias para a criação de um ambiente de simulação. As linguagens de simulação podem ser classificadas como: orientadas a acontecimento, actividade ou processo e, a escolha irá depender da orientação do modelo.

- **Linguagens Orientadas a Acontecimento**

  Nas linguagens orientadas a acontecimento, o programa de simulação é organizado como um conjunto de rotinas ou secções de acontecimentos [MACD75].

  Estas linguagens tendem a impor uma visão global e de alto nível do sistema ao modelador, o que as torna mais adequadas a modelos de pequena e média complexidade. Com o aumento da complexidade, sugere-se a estruturação do modelo para utilização de uma linguagem orientada ao processo. Exemplos de linguagens de simulação orientadas a acontecimento são [MACD75]: SIMSCRIPT, SLAM II, GASP, SMPL.

- **Linguagens Orientadas a Processo**

  Uma linguagem orientada ao processo é mais utilizada em análise de sistemas mais complexos, como por exemplo sistemas das áreas de robótica, projecto de sistemas computacionais e redes de comunicação [SAYD85].

Nesta abordagem o sistema é visto como uma colecção de processos interactivos. Um processo tem as seguintes características [SAYD85]:

- pode ser activado, ficar em estado suspenso ou terminar a sua execução num certo instante de tempo ou baseado em alguma condição;
- uma vez activado, repete o seu comportamento até ser colocado no estado suspenso ou terminar a sua execução.

Programas de simulação escritos nestas linguagens são construídos de uma forma mais natural, tornando o sistema e o modelo similares. Exemplos de linguagens de simulação orientadas a processo são [SAYD85]:

GPSS *(General Purpose Simulation System)*, *SIMULA*, SIMPL/I X, SIMAN *(Simulation Modeling And Analysis)*, ASPOL e SOL.

- **Linguagens Orientadas a Actividade**

  Para escrever um programa numa linguagem orientada à actividade devem descrever-se as actividades existentes no sistema. Para certos tipos de problemas essa abordagem fornece-nos um modelo exacto, adequado para situações onde a duração da actividade é indefinida [SOAR92]. Exemplos de linguagens orientadas a actividade são:

  - CSL (*Control and Simulation Language*) [RODR96].
  - ECSL (*Extended CSL*) [RODR96].

Características desejáveis nas linguagens de simulação

- Flexibilidade de modelação.
- Facilidade para o desenvolvimento e rastreabilidade do modelo.
- Execução rápida do modelo.
- Permitir a elaboração de modelos enormes.
- Disponibilidade em várias plataformas.
- Capacidade de animação (visualização gráfica do sistema).
- Capacidades estatísticas (incluir distribuições de variáveis de entrada e saída, réplicas das experiências, intervalos de confiança, etc.).
- Relatórios dos resultados (relatórios gerais ou específicos, tabelas ou gráficos, acesso a amostras individuais).

Ferramentas de Simulação

Os simuladores são ferramentas de simulação orientadas para aplicações particulares. Com estas ferramentas os modelos constroem-se mediante janelas de diálogo, menus, ou gráficos, sendo estes últimos os mais usados. As principais vantagens dos simuladores são a facilidade de aprendizagem e de manuseamento. Como contrapartida, apresentam menor flexibilidade que as linguagens de simulação. Dentro dos simuladores pode-se distinguir dois tipos: os de propósito geral como o *Arena*® e os específicos, desenhados para uma aplicação concreta. Devido ao facto de serem muito específicas para uma determinada aplicação, estes pacotes oferecem facilidades em pontos particulares para a qual foram desenvolvidos, mas são pouco flexíveis a alterações [SANT94]. Nestes pacotes, a formulação do modelo é construída na própria ferramenta, sendo os parâmetros do modelo especificados de forma definida pela ferramenta. Alguns exemplos são [SOAR92]:

- BEST/1, CMF, FIVE, PERFORMS, UM Predictor, XL: para modelação de sistemas de computação e respectivos sistemas operativos.
- PET, NETWORLK: para modelação de redes de computadores.
- BETHSIM: para modelação de operações de siderurgia.
- IDSS: Para modelação de sistemas aeroespaciais.
- ServioceModel: para a reengenharia de processos e serviços.
- SAINT, HOS e MOPADS: para modelação de tarefas realizadas por seres humanos.
- MedModel: para a área da medicina.

Qualquer dos simuladores citados oferece um conjunto de ferramentas avançadas para a planificação, simulação, execução e controlo dos processos de uma empresa; em particular de uma empresa do sector industrial em que se desenvolvem processos produtivos (automatizados ou não). Com eles pode-se determinar, entre outras coisas, qual a quantidade adequada de produtos a fabricar, o momento mais oportuno para a sua realização e o método mais rentável.

Algumas destas ferramentas possibilitam o escalamento total e trabalhar em tempo real. O escalamento torna-os facilmente adaptáveis a empresas de qualquer dimensão e a qualquer alteração.

Estas ferramentas têm características necessárias à simulação que diminuem o tempo de desenvolvimento do modelo de simulação. Possuem recursos adicionais para visualização, animação e tratamento de dados. Em geral exige formação na ferramenta.

Bibliotecas de Simulação

Para facilitar o trabalho do programador, algumas bibliotecas (conjunto de rotinas disponíveis ao programador) podem ser inseridas em linguagens convencionais, compondo assim um ambiente de simulação. O trabalho do programador é facilitado pois este não necessita aprender uma nova linguagem. Algumas bibliotecas são:

- da linguagem C: SMPL [MACD87] (orientada a objecto), CSIM [EDWA92] e EFC [SOUZ92] (orientadas a processo);
- da linguagem C++: SIMPACK [FISH95];
- da linguagem modula 2: EFM [SPOL92], HPSIM [SHAR88] (orientadas a processo);
- da linguagem SIMAN: *Arena*® (orientada a processo).

Ferramentas e Linguagens de Simulação

A escolha da ferramenta de simulação considera factores tais como:

- Características da simulação a ser realizada;
- Características das ferramentas disponíveis;
- Ambiente disponível para simulação;
- Nível técnico da equipa de projectistas que realizarão a implementação do simulador;
- Orçamento disponível para compra da ferramenta caso não exista disponível;
- Tempo disponível para realizar a simulação.

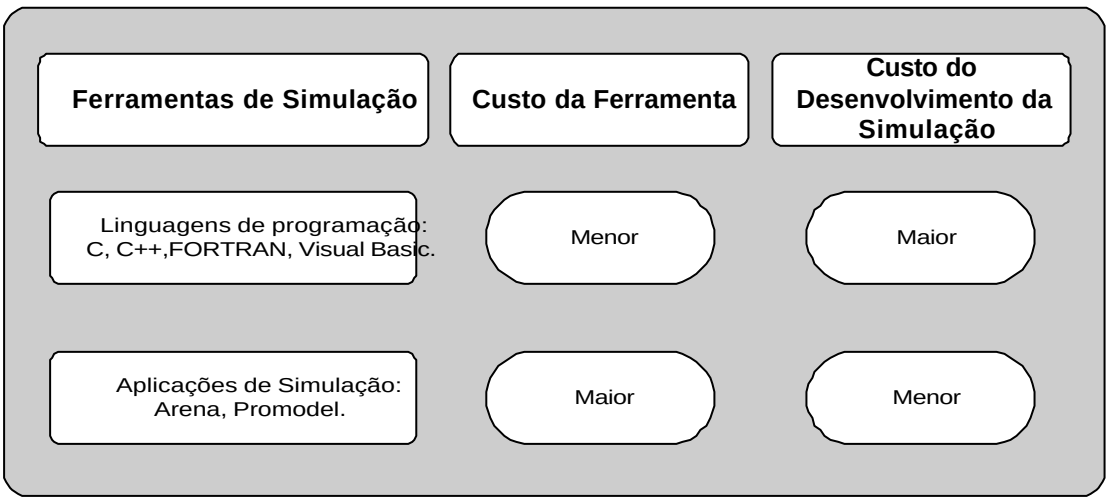


Figura 2.14 – Esquema do custo das ferramentas de simulação

As Figuras 2.14 e 2.15 representam duas perspectivas diferentes de classificar ferramentas de simulação. Na Figura 2.14 observa-se a relação entre o custo da ferramenta e o custo do desenvolvimento de um modelo de simulação.

A Figura 2.15 confronta a flexibilidade das ferramentas de de simulação com a facilidade de utilização por parte do utilizador.

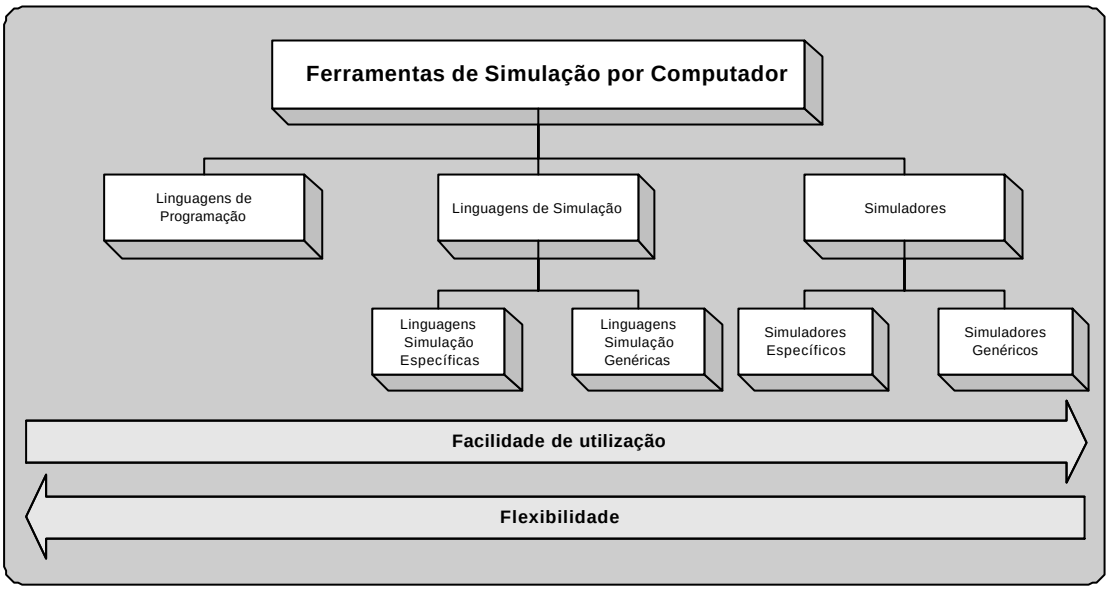


Figura 2.15 – Esquema da complexidade das ferramentas de simulação

2.4 Aplicações da Simulação

Devido à sua versatilidade, flexibilidade e poder, a simulação pode ser aplicada em qualquer estudo ou pesquisa na utilização de técnicas de investigação operacional [SHAN81]. Quase todos os tipos de sistemas foram (ou podem ser) simulados, e a larga faixa de aplicações quase extrapola a classificação. Algumas aplicações representativas estão descritas a seguir [PEGD90]:

- **Sistemas Computacionais**: componentes de hardware, software, redes de computadores, estruturas e gestão de base de dados, processamento da informação, confiabilidade de hardware e software.
- **Manufactura**: sistemas de manuseamento e armazenamento de materiais, linhas de montagem, recursos automatizados de produção e armazenamento, sistemas de controlo de stocks, estudos de manutenção, *layout* de unidades fabris, projecto de máquinas.

- **Negócios**: análises de produtos, política de preços, estratégias de marketing, estudos de aquisição de empresas, análise de fluxo de caixa, previsão, alternativas de transporte, planeamento de aumento de trabalho.
- **Governo**: armamentos e tácticas militares, previsão de crescimento populacional, uso do solo, sistemas de saúde, sistemas contra incêndio, polícia, justiça criminal, projectos de estradas, controle de tráfego, serviços de saneamento.
- **Ecologia e Meio Ambiente**: poluição e purificação de água, controle de desperdícios, poluição do ar, controle de pragas, previsões climáticas, análise de terramotos e tempestades, exploração e extracção mineral, sistemas de energia solar.
- **Sociedade e Comportamento**: análises alimento/população, políticas educacionais, estrutura organizacional, análise de sistemas sociais, administração universitária.
- **Biociências**: análise de performance desportiva, ciclos de vida biológicos, estudos biomédicos.

Segundo Pegden [PEGD90], esta lista não abrange todas as aplicações da simulação. Ela apenas sugere a grande utilização da simulação para ajudar a resolver uma grande gama de importantes problemas. É uma ferramenta barata para se *testar* sistemas propostos, planos, ou políticas antes de se assumir com os custos de protótipos, campos de testes, ou implementações reais. A simulação pode ser encarada como uma apólice de seguro do sistema estudado.

Quando a simulação é usada como ferramenta de projecto, o estudo é tipicamente motivado por questões como:

Qual será a capacidade de produção do projecto?

Vai de encontro aos nossos objectivos?

Onde estão os gargalos?

O que pode ser mudado para aumentar a capacidade?

Qual é a melhor de várias alternativas de projecto?

A performance do sistema varia em função do número e tipo das máquinas e do número de operadores?

Qual é a confiabilidade do sistema?

As paragens das máquinas afectarão a capacidade?

Capítulo 3

Caso de Estudo – Resolução

Neste capítulo, é apresentada a proposta para resolução do caso de estudo tratado nesta dissertação. Inicialmente é introduzida uma breve revisão aos problemas do projecto e aos assuntos considerados mais relevantes. A aplicação da simulação para o caso de estudo, associada à resolução de alguns dos objectivos definidos, implicou a elaboração de uma ferramenta informática. Para a implementação dos modelos em simulação, utilizou-se o simulador *Arena*®, onde foi desenhado o modelo característico do sistema produtivo. Este modelo, representação "fidedigna" do sistema real, contém também, um módulo de controlo, elaborado em VBA para aplicar os critérios e estratégias definidas pelo utilizador.

A existência da ferramenta informática é essencial para a implementação das estratégias e monitorização da simulação. Nesta ferramenta informática, tirou-se proveito de

um interface gráfico mais amigável para simplificar toda a parametrização do modelo e a escolha das estratégias para o projecto que se pretende estudar.

De seguida, são enumerados os temas que tidos como principais factores intervenientes na produção que mereceram maior atenção na elaboração da ferramenta informática e na resolução do caso de estudo.

À gestão da manutenção foi dedicada especial atenção. Realizar a gestão adequada do equipamentos, tendo em conta a disponibilidade dos meios, materiais, técnicos e máquinas, é um objectivo a atingir. Para resolver os problemas associados à equipa de manutenção estabeleceu-se um procedimento para as situações em que ocorrem avarias nas máquinas. Uma vez que existe apenas uma equipa de manutenção, esta ocorre com prioridade à máquina com maior interesse para a organização.

A definição do sistema de gestão responsável pela deslocação do operário perante um conjunto de máquinas, sob a sua responsabilidade, necessita de uma avaliação do impacto que algumas estratégias de controlo podem provocar do sistema produtivo. Implementaram-se algumas regras de prioridade para determinar para que máquina se deve deslocar o operador se, no mesmo instante, estiver parada mais do que uma máquina.

A gestão dos materiais é uma constante preocupação por parte de qualquer gestão empresarial. Neste projecto, assume-se este tema como um dos principais assuntos a estudar. O fluxo de matérias-primas é monitorizado desde a entrada dos materiais no armazém de matérias-primas até ao armazém de produto acabado, neste caso, a área de tricotagem.

Como normalmente é conhecida a carteira de encomendas, a capacidade de produção, o número de máquinas e operários disponíveis, objectiva-se satisfazer as encomendas em períodos de tempo determinados e definir a data de conclusão para essas mesmas quantidades. Se se sabe quando, quanto e o que vai ser necessário em termos de matérias-primas pode-se evitar comprar quantidades desadequadas às necessidades da produção, reduzir aos stocks, e melhorar o processo de compra das matérias-primas, melhorando a gestão de stocks e aprovisionamentos.

O escalonamento dos produtos às máquinas é outra situação a que será dada a devida importância, saber que máquina vai produzir o quê, quanto e porquê são questões cruciais que qualquer organização industrial tem interesse em conhecer profundamente.

É importante ter em atenção que determinadas máquinas podem ser mais aptas a fazer uns produtos que outros, estarem restritas a fazer um único produto ou até impossibilitadas de executar determinados produtos. A aplicação destas restrições condiciona muito a planificação, tornando-a muito complexa e demorada para um parque de máquinas razoável.

Tendo conhecimento da relevância que este aspecto tem para a obtenção de bons índices de rentabilidade das máquinas, são anexadas algumas alternativas para o escalonamento de produtos às máquinas.

3.1 Gestão e Controlo da Equipa de Manutenção

Missão e Objectivo

Todos os equipamentos, sistemas e instalações, sejam eles mecânicos, eléctricos, hidráulicos ou pneumáticos, estão sujeitos a ver degradadas as suas condições normais de operacionalidade com o decorrer do tempo, em consequência do uso e até por causas fortuitas. É missão da *Manutenção* repor essa operacionalidade em níveis correctos.

Para cumprir a sua missão, a Manutenção recorre a um conjunto diversificado de tarefas seleccionadas e programadas, de acordo com as características e utilização do seu objecto e os padrões de serviço que lhe foram fixados. Essas tarefas são, por exemplo, a lubrificação, a limpeza, o ensaio, a reparação, a substituição, a modificação, a inspecção, a calibração, a revisão geral ou o controlo de condição.

A seguir, descreve-se algumas noções, cuidados ou pontos relevantes a ter com a Manutenção, que foram considerados importantes na elaboração da estratégia de gestão e controlo da manutenção neste projecto.

Formas de Manutenção

A forma de manutenção mais antiga é a que consiste em deixar operar o equipamento até à ocorrência de uma avaria para então proceder à sua reparação. É a manutenção designada por *resolutiva*, *curativa* ou *correctiva.* Por reagir ao acontecimento depois da sua ocorrência diz-se que é uma manutenção do tipo *reactivo.*

Esta técnica de manutenção será sempre usada, pelo menos em situações em que não haja meios para prevenir a ocorrência da avaria.

Progressivamente têm vindo a ser introduzidas outras técnicas de manutenção que, por agirem antes da ocorrência da avaria, são designadas por *proactivas.*

Um tipo de manutenção proactivo é a manutenção preventiva que consiste na execução planeada com periodicidades fixadas, de tarefas de manutenção como lubrificação, a regulação, a substituição ou a revisão geral.

Esta manutenção, que visa reduzir o risco de ocorrência de avarias, é adequada para órgãos, equipamentos ou sistemas que exibam um padrão de comportamento com certa regularidade, que permita estimar com algum rigor quando as suas características se vão degradar abaixo dos níveis mínimos aceitáveis.

Em relação à manutenção resolutiva ou curativa, a manutenção preventiva apresenta as seguintes vantagens [PINT94]:

-Prolonga a vida útil dos equipamentos evitando a ocorrência de avarias graves;

-Aumenta a disponibilidade dos equipamentos para operação porque reduz a taxa de paragens por avaria;

-Melhora o aproveitamento da mão-de-obra de manutenção porque permite efectuar um planeamento da sua utilização;

-Melhora as relações com a produção porque as paragens das máquinas podem ser (dentro de certos limites) programadas respeitando as conveniências da produção;

-Melhora a produtividade do técnicos da manutenção porque as tarefas são repetitivas, permitindo aprender com a experiência, e previsíveis, permitindo formação planeada;

-Reduz a necessidade de imobilizar material em stock porque se pode conjugar a sua compra com a data prevista para a sua utilização:

-Aumenta a segurança dos operadores e do equipamento.

Planeamento da Manutenção

O trabalho de manutenção pode ser extremamente improdutivo. Mais de metade do tempo de presença do pessoal de manutenção pode ser perdido por razões como:

- Deslocações de um local para outro em resultado dos circuitos estarem mal organizados;
- Espera por material que pode nem sequer existir em stock;
- Espera por ferramentas e equipamentos de ensaio e medida ou deslocações para trocar por outros por estarem inoperativos ou não serem adequados;
- Falta de documentação técnica necessária à execução do trabalho;
- Interrupções de trabalho para atender a outros considerados mais urgentes;
- Indisponibilidade da produção para parar o equipamento que deveria ser submetido a manutenção.

Estas situações serão familiares na maior parte das empresas. A única forma de lhes por termo é adoptar um sistema adequado de planeamento e controlo de manutenção.

Registos Históricos

Cada equipamento deve ter o seu registo histórico onde constem, além da sua identificação e localização, todas as intervenções de manutenção resolutiva ou preventiva.

Nele constarão, nomeadamente, a data de cada participação de avaria, a descrição da avaria e da respectiva acção correctiva, a identificação dos componentes substituídos, a mão-de-obra gasta e o tempo de paragem do equipamento, e os custos da reparação, em mão-de-obra e materiais.

Finalmente, para completar a história do equipamento, são registadas na sua ficha todas as intervenções de manutenção preventiva, com a indicação da data e da tarefa de manutenção que foi efectuada.

Este registo é essencial não só para apoio à pesquisa de avarias mas também para habilitar a tomar decisões de carácter económico como, por exemplo, a oportunidade óptima de proceder à substituição do equipamento por se ter tornado antieconómica a sua manutenção.

O ficheiro histórico pode ser explorado, designadamente, para [PINT94]:

a) **Fiabilidade** -Determinação das leis de fiabilidade, perfil de avaria, taxa de avaria, etc.
b) **Disponibilidade** -Determinação da disponibilidade média do equipamento.
c) **Métodos** -Determinação de pontos fracos do equipamento e de avarias mais frequentes.
d) **Gestão de Stocks** -Determinação dos consumos habituais de peças e módulos.
e) **Gestão de Manutenção** -Determinação de custos por equipamento, por tipo de avaria, por tipo de intervenção, etc.

Uso de Computadores em Manutenção

Os computadores podem ter múltiplas aplicações directa ou indirectamente relacionadas com a manutenção. Algumas delas, no entanto, não serão aqui consideradas por serem do domínio da actividade industrial, outras aplicações, embora respeitando directamente à manutenção, são exclusivamente de carácter técnico. O objectivo desta secção é o uso dos computadores em funções relacionadas com a gestão da manutenção. A utilização de computadores na gestão da manutenção apresenta os seguintes benefícios potenciais [PINT94]:

a) Maior produtividade da manutenção, devido a uma melhor utilização de todos os recursos (mão-de-obra, materiais, equipamentos, ferramentas, instalações).

b) Redução dos custos de manutenção, porque há conhecimento mais rápido e rigoroso de todos os factores de custo permitindo tomar decisões correctas em tempo útil.

c) Redução dos tempos de imobilização não programada dos equipamentos, porque é possível utilizar de forma mais extensa a manutenção preventiva.

d) Aumento do tempo de vida dos equipamentos, por beneficiarem de mais e melhor manutenção.

e) Redução de todos os tempos de espera, por melhor organização do trabalho e melhor informação sobre as localizações dos materiais, ferramentas, equipamentos, documentação técnica, etc.

F Menor tempo de imobilização por avaria, porque há um acesso rápido e selectivo à história da máquina e seus modos de avaria característicos, permitindo uma detecção mais eficaz.

g) Menor perturbação do ritmo de produção, por ser mais fácil articular o plano de manutenção com o plano de produção.

h) Maior eficácia da gestão, porque pode apoiar as suas decisões de carácter técnico ou económico em informações actualizadas e fidedignas e é alertada para desvios relevantes logo que eles ocorram.

i) Melhor organização da manutenção, porque a análise que precede a especialização de um sistema de gestão informatizada da manutenção revela, geralmente, insuficiências, desajustes ou redundâncias que devem ser corrigidos.

A gestão informatizada da manutenção pode cobrir as seguintes funções:

1. Planeamento e controlo da manutenção
2. Manutenção Programada
3. Orçamento e custos de Manutenção
4. Informação para Gestão
5. Outras funções subsidiárias, de carácter técnico ou administrativo.

Estratégia Desenvolvida para a Área da Manutenção

Em resultado das informações obtidas por pesquisa bibliográfica ou por pareceres de responsáveis de manutenção entendeu-se fazer a gestão da manutenção tal como se pode observar na Figura 3.3.

No caso de estudo, tratando-se de uma unidade industrial relativamente pequena, considerou-se uma equipa de manutenção para toda a área de produção, significando que,

independentemente do número de linhas existentes, ter-se-á apenas uma equipa para dar apoio a todas as máquinas de qualquer linha. A equipa de manutenção opera 24 "horas" por dia para todas as linhas de fabrico de meias tubulares.

Nesta altura, convém esclarecer que *linha*, não significa uma linha de produção tradicional com uma sequência de tarefas definidas, significa um conjunto de máquinas, sob a responsabilidade do operário de máquina, que garante a sua operacionalidade. Recorda-se que a máquina necessita da presença do operador para retirar o lote, quando completo, da máquina para que esta possa retomar o trabalho até à conclusão da planificação prevista para essa mesma máquina.

O termo linha surge num sentido mais amplo. Na Figura 3.1 observa-se 3 linhas, a linha 1, a linha 2 e a linha 3 com 3, 2 e 4 máquinas respectivamente. As linhas podem ter diferentes quantidades de máquinas e cada linha pode ter vários tipos de máquinas (diferenciadas pela imagem).

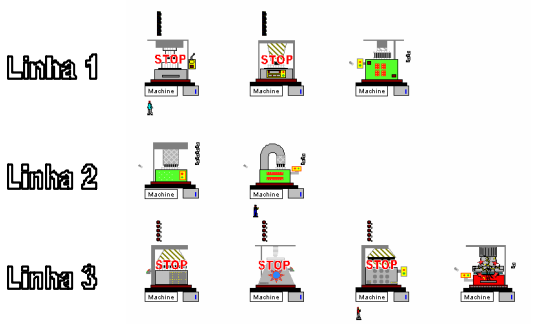


Figura 3.1 – Exemplo de uma máquina avariada

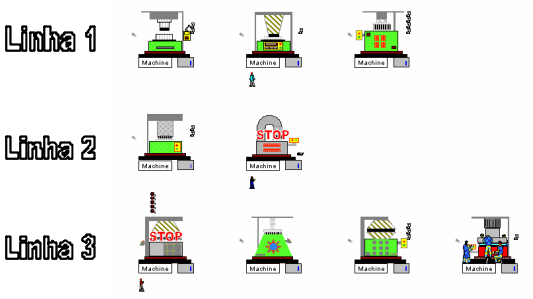


Figura 3.2 – Exemplo de uma máquina em manutenção

O processo de manutenção é activado sempre que exista uma máquina avariada e a equipa de manutenção esteja livre. No caso da equipa estar ocupada a avaria passa para a fila de espera. Se mais máquinas avariarem enquanto a equipa estiver ocupada, a equipa deslocar-se-á à máquina seleccionada entre todas as que estiverem avariadas pelo procedimento para a gestão da equipa de manutenção.

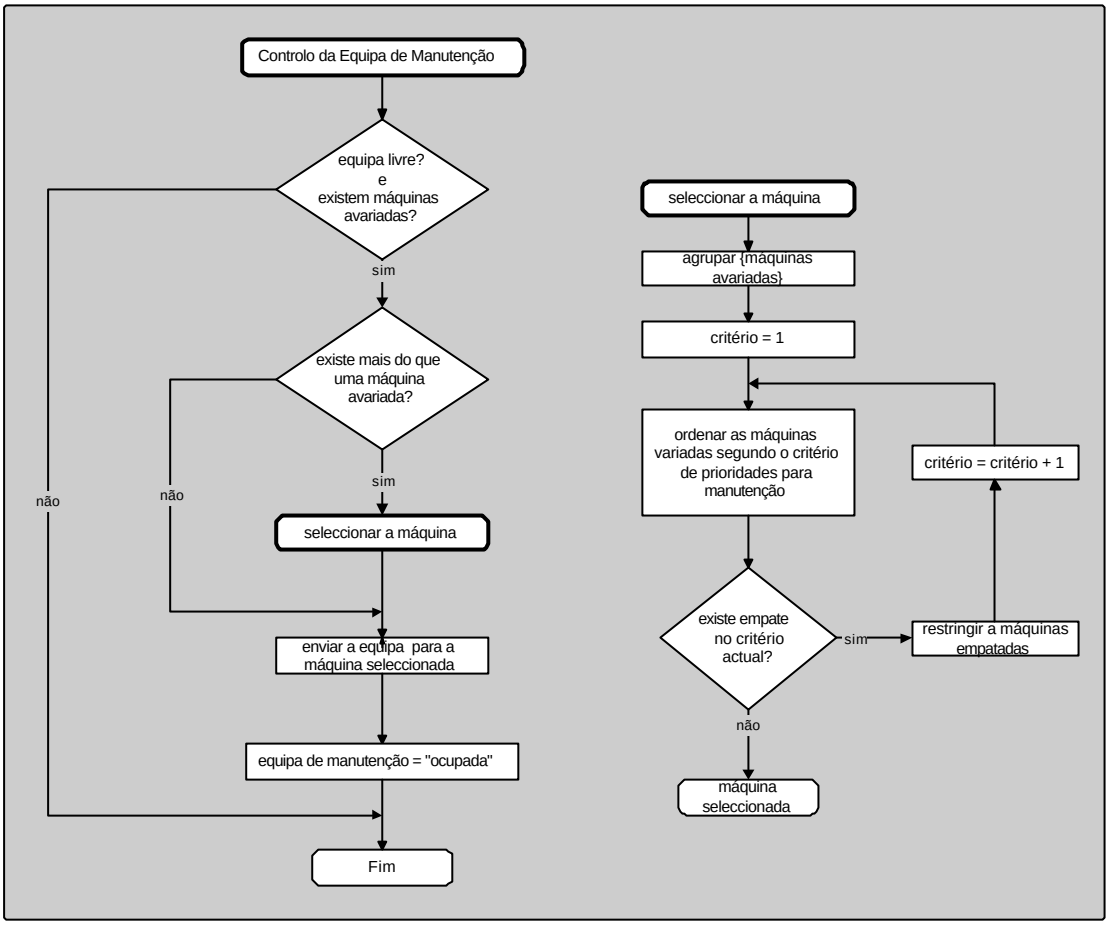


Figura 3.3 – Procedimento para a gestão da equipa de manutenção

Existem três situações possíveis em que a equipa de manutenção é solicitada, cada uma com o seu procedimento:

1. não existem máquinas avariadas, a equipa prossegue o plano de manutenções preventivas;
2. existe apenas uma máquina avariada, a equipa desloca-se a essa máquina para a reparar;
3. existem várias máquinas avariadas, o procedimento selecciona a máquina através de um processo iterativo de prioridades.

Relativamente à situação do ponto 1 para cada máquina é elaborado um plano de manutenções preventivas, criado automaticamente aquando o modelo. Este plano é estabelecido em função das características técnicas de cada máquina, normalmente o factor predominante para esse cálculo é a quantidade produzida.

Existindo apenas uma máquina avariada (ponto 2) a equipa de manutenção deslocar-se-á para essa máquina salvo as excepções em que esteja ocupada.

Ocorrendo as situações descritas no ponto 3, existe um procedimento *seleccionar máquina* a percorrer, este procedimento prevê 4 níveis de critérios para priorizar a máquina que vai ser reparada pela equipa de manutenção. Os critérios são:

- **Velocidade** : rapidez a que a máquina está a funcionar ou tem capacidade de trabalhar em relação a outras.
- **Setup**: tempo de preparação que a máquina necessita para retomar o normal funcionamento quando existe uma mudança de tipo de produto (quando a máquina deixa de produzir *n* lotes do produto *x* para produzir *m* lotes do produto *y*.
- **Distância**: pretende reflectir o tempo que a equipa de manutenção demora a chegar a uma determinada máquina, é medida instantaneamente, isto é, dependendo do local da equipa, são calculados os tempos que a equipa demoraria desde o local onde se encontra a qualquer máquina existente.
- **Fila de Espera**, pretende dar prioridade à quantidade da fila de espera das máquinas, ou seja, a quantidade de trabalho programado ou planificado que as máquinas têm para efectuar.

Estes critérios podem ser seleccionados pela ordem preferencial que se pretender, em cada critério tem-se a opção de maior ou menor, isto é, pode-se escolher entre ordenação por ordem crescente ou decrescente.

Para esclarecer melhor como funciona a gestão da equipa de manutenção, elaborou-se um exemplo ao qual se vai aplicar o procedimento da Figura 3.3.

Escolhemos os seguintes critérios com a prioridade:

Prioridade	**Critério**	**Ordenação**
1º	Velocidade	Maior (a mais veloz)
2º	Setup	Menor (a que tem menor Setup)
3º	Distância	Menor (a que se situa mais perto)

Tabela 3.1 – Definição de critérios para aplicação no exemplo

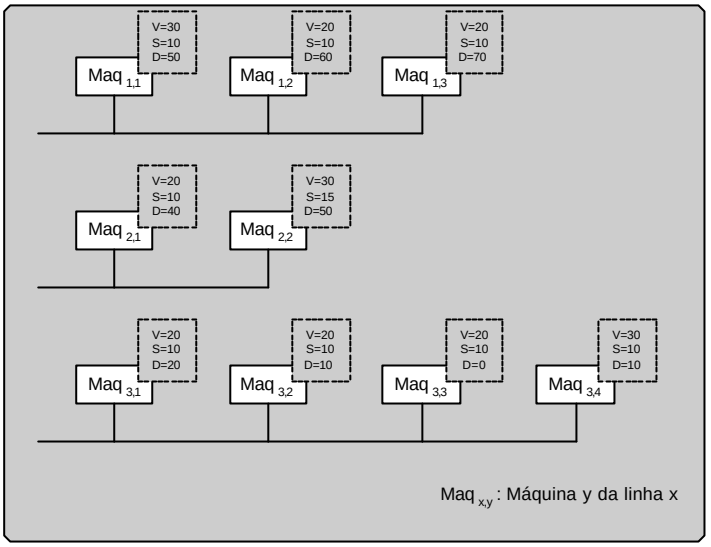


Figura 3.4 – Exemplo de um caso prático

Partindo do principio que todas as máquinas estão avariadas, e a equipa de manutenção está livre, podendo assim deslocar-se de imediato a qualquer máquina avariada.

Iteração	Critério	Máquinas Avariadas		Máquina Seleccionada
1	1º	$Maq_{1,1}$, $Maq_{2,2}$, $Maq_{3,4}$, $Maq_{1,2}$, $Maq_{1,3}$, $Maq_{2,1}$, $Maq_{3,1}$, $Maq_{3,2}$, $Maq_{3,3}$	Ordenadas	-
		$Maq_{1,1}$, $Maq_{2,2}$, $Maq_{3,4}$	Empatadas	-
2	2º	$Maq_{1,1}$, $Maq_{3,4}$, $Maq_{2,2}$	Ordenadas	-
		$Maq_{1,1}$, $Maq_{3,4}$	Empatadas	-
3	3º	$Maq_{3,4}$, $Maq_{1,1}$	Ordenadas	
			Empatadas	**$Maq_{3,4}$**

Tabela 3.2 – Exemplo de aplicação da Figura 3.3

Na 1ª iteração temos a $MAQ_{1,1}$, a $MAQ_{2,2}$ e a $MAQ_{3,4}$ velocidade 30 e $MAQ_{1,2}$, $MAQ_{1,3}$, $MAQ_{2,1}$, $MAQ_{3,1}$, $MAQ_{3,2}$, $MAQ_{3,3}$ com velocidade 20.
Avançamos para a 2ª iteração e ficamos com a $MAQ_{1,1}$ e a $MAQ_{3,4}$ que têm setup=10, que é menor ao da $MAQ_{2,2}$ com setup=14.

Com o critério de menor distância, a 3ª iteração encontra duas máquinas com distâncias diferentes, seleccionamos a $MAQ_{3,4}$ que tem a menor distância=10 em vez da $MAQ_{1,1}$ que se encontra a uma distância=50, termina, então, o processo de escolha da máquina que deve ser reparada de seguida.

Aos critérios existentes para seleccionar as máquinas, pode-se, a qualquer modelo acrescentar mais critérios para a gestão e controlo da equipa de manutenção seleccionar a máquina para a qual se deslocará em seguida.

No caso do procedimento chegar ao fim de todos os critérios e continuar com máquinas para optar, a equipa de manutenção deslocar-se-á para a máquina que se encontrar na primeira

posição da lista, isto é, se se considerar as máquinas como Maq_{xy} com x = linha da máquina e y = a posição da máquina nessa linha, procura a máquina com o menor x e menor y respectivamente.

3.2 Gestão e Controlo do Operador de Máquina

Pressupostos de Estudo do Posto de Trabalho

A adequação do trabalho é a procura da cooperação ideal entre o Homem, o meio de produção e o objecto do trabalho, de acordo com a tarefa e mediante uma correcta organização de sistemas de trabalho, atendendo à capacidade de rendimento do Homem às suas necessidades [BILS94].

O Homem é o trabalhador, homem ou mulher, que empregam a sua força de trabalho na execução de tarefas, e que conjuntamente com o meio de produção constituem a capacidade de trabalho. O Homem, através da percepção adquirida na formação, *manuseia* o meio de produção (carrega botões, acciona alavancas, etc.), respondendo, o meio de produção à acção. O meio de produção reage através de sinais acústicos, ópticos, etc.; ele comunica com o Homem. Nessa continua relação, na execução dessa tarefa, origina-se o Processo [BILS94].

A tarefa é uma exortação do Homem ao executar actividades para uma determinada finalidade. Ela caracteriza o sistema de trabalho [BILS94].

Para um melhor cumprimento das tarefas que são imputadas ao homem, é imprescindível que tenha sido treinado e familiarizado sistematicamente nas respectivas tarefas, para agir eficientemente nesse sector.

Com referência aos objectivos, observa-se, também, o equilíbrio que deve existir entre a humanização do trabalho e o aumento da rentabilidade, ou seja, não se pode falar de organização empresarial onde o aumento de rentabilidade se consegue em detrimento da humanização do trabalho.

A ergonomia faz parte da ciência do trabalho que, com conhecimentos anatómicos, psicológicos, fisiológicos sociológicos e técnicos, fornece métodos para determinar os limites de exequibilidade e suportabilidade do trabalho humano.

Missão e Objectivo

As máquinas têm uma determinada quantidade planificada. Essa quantidade é dividida em pequenas quantidades a produzir, isto é, um lote. Quando a quantidade do lote é

atingida, a máquina requer a presença do operário para retirar essa quantidade, coloca-a num saco e activa a máquina para produzir o lote seguinte. Quando as máquinas terminam o lote, emitem um sinal, sendo, portanto, perfeitamente identificáveis por parte do operador de máquina.

Como se pode imaginar, quando o operador se desloca a uma máquina ou está a executar a tarefa de repor uma máquina em funcionamento, várias máquinas podem parar. Pode-se, então, colocar a seguinte questão "para que máquina o operador vai a seguir?", para a que está mais perto, a que está mais longe, a mais rápida ou a que tem produção mais atrasada? Outra questão importante é a da responsabilidade, isto é, quem assume a decisão do operador ter escolhido a máquina *x* em vez da máquina *y*. Para dar resposta a estas questões foi adoptada uma estratégia que escolhe a *máquina seguinte*, máquina a que o operador se vai deslocar quando fica livre (ou termina a tarefa que estava a fazer).

A estratégia de gestão do operador foi elaborada para que os agentes de decisão possam decidir sob vários cenários, por exemplo, o que é mais rentável, mais eficiente ou urgente. Como facilmente se pode imaginar, ao que se pode chamar eficiência de operador pode não significar mais quantidade produzida, pode-se ter o operador com taxas de ocupação muito boas mas os resultados da produção não serem os melhores, pois, as máquinas mais eficientes podem estar muito tempo paradas.

Em situações críticas, no caso de se pretender toda a produção planificada terminada num determinado período (para entrega ao cliente ou fazer inventário), necessita-se que exista um equilíbrio entre a velocidade das máquinas, a quantidade planificada e o deslocamento do operador à máquina que esteja com a produção mais atrasada.

Para possibilitar todas estas situações e outras consideradas de interesse, foi definida uma estratégia por níveis de prioridade, cada nível pode ser ascendente ou descendente. Por exemplo, se a velocidade fosse o critério escolhido, poder-se-á ainda, definir se o operador se desloca às máquinas mais velozes ou preferencialmente às mais lentas.

A estrutura da estratégia de gestão do operador de máquina foi elaborada de forma a ser o mais flexível possível, que, pela sua configuração, permite que cada operador de máquina tenha a sua própria estratégia para seleccionar a máquina seguinte e, consequentemente, cada linha de produção.

Estratégia Desenvolvida para o Operador de Máquina

Observando a Figura 3.5, verifica-se que sempre que um operador está livre existe a preocupação em atribuir-lhe uma máquina. De facto, nesta actividade, o operador surge como o elemento crítico, decisivo para a obtenção de bons resultados de produção. A gestão deste recurso pode significar diferenças muito significativas nas taxas de ocupação das máquinas.

Depois de ter terminado o lote que estava a elaborar qualquer máquina pode esperar pelo operador bastante tempo, para ensacar o lote e permitir a máquina executar o lote seguinte. Esta operação é obrigatória, uma vez que todas as máquinas necessitam do operador para reiniciarem a actividade.

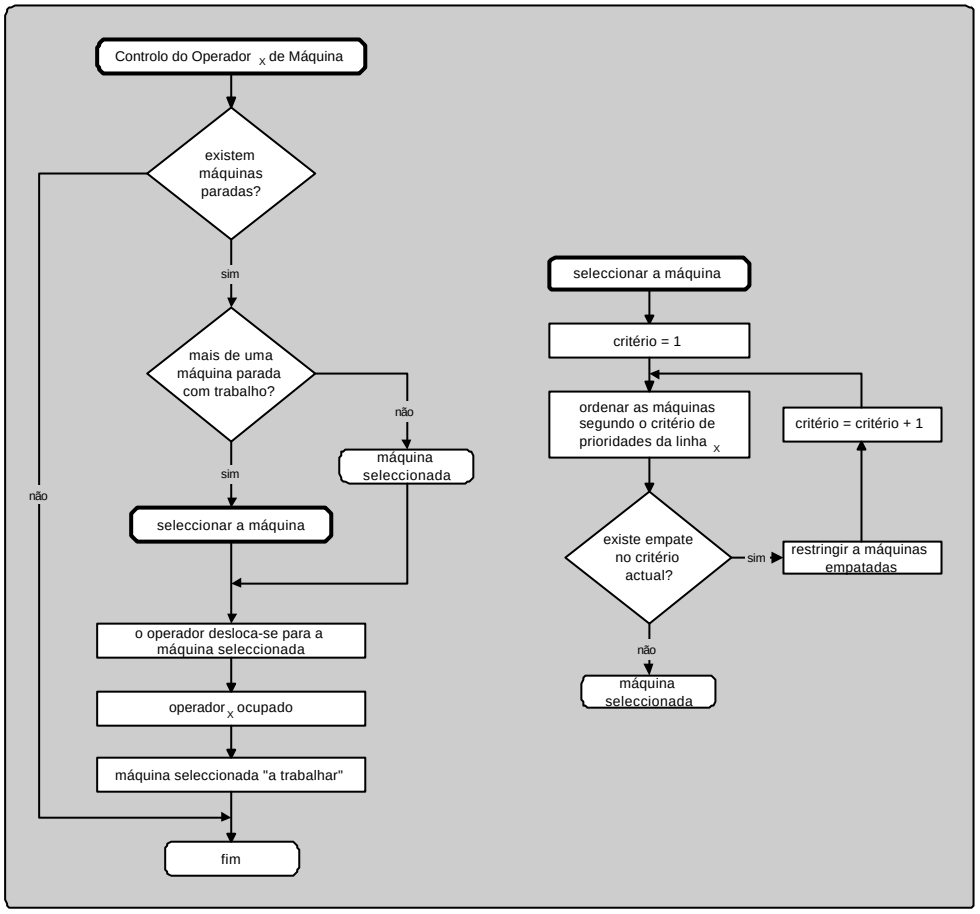


Figura 3.5 – Diagrama de gestão de um operador de máquina

Uma situação importante que está perfeitamente definida na Figura 3.5, é o facto de o operador não se deslocar para máquinas que apesar de estarem paradas não têm trabalho a efectuar, ou se se preferir têm fila de espera vazia. Considera-se uma máquina parada com trabalho, as que têm lotes a produzir e está a aguardar a presença do operador de máquina.

Um operador de máquina é responsável por um determinado número de máquinas (linha). A composição da linha pode ser uniforme ou constituída por vários tipos de máquinas. Como numa linha podem existir vários tipos de máquina com diferentes características, a gestão do operador é feito por prioridades dessas mesmas características e outras consideradas relevantes para o desempenho do processo produtivo. Inicialmente, foram criadas quatro prioridades para a avaliação do desempenho do sistema produtivo: a velocidade da máquina, a distância da máquina à posição do operador, a quantidade de entidades na fila de espera da máquina e o tempo de preparação da máquina (*setup*). O modelo desenvolvido permite a

remoção ou a adição de critérios para seleccionar a máquina para o operador. Em cada linha existe pelo menos um operador.

O diagrama representado na Figura 3.6 reflecte o funcionamento geral de todas as linhas existentes no nosso sistema produtivo. O procedimento é o seguinte: são percorridas todas as linhas existentes e para cada uma elas verifica-se se o operador necessita de ser enviado para outra máquina. O processo de enviar o operador de uma linha *x* de uma máquina para outra máquina dessa mesma linha obedece ao procedimento de seleccionar máquina e respectivos critérios correspondentes à linha *x*. O modelo desenvolvido para o operador de máquina permite estabelecer critérios diferentes para linhas diferentes. Portanto, o critério para atribuir a máquina seguinte ao operador não necessita, nem tem que ser, o mesmo para todas as linhas.

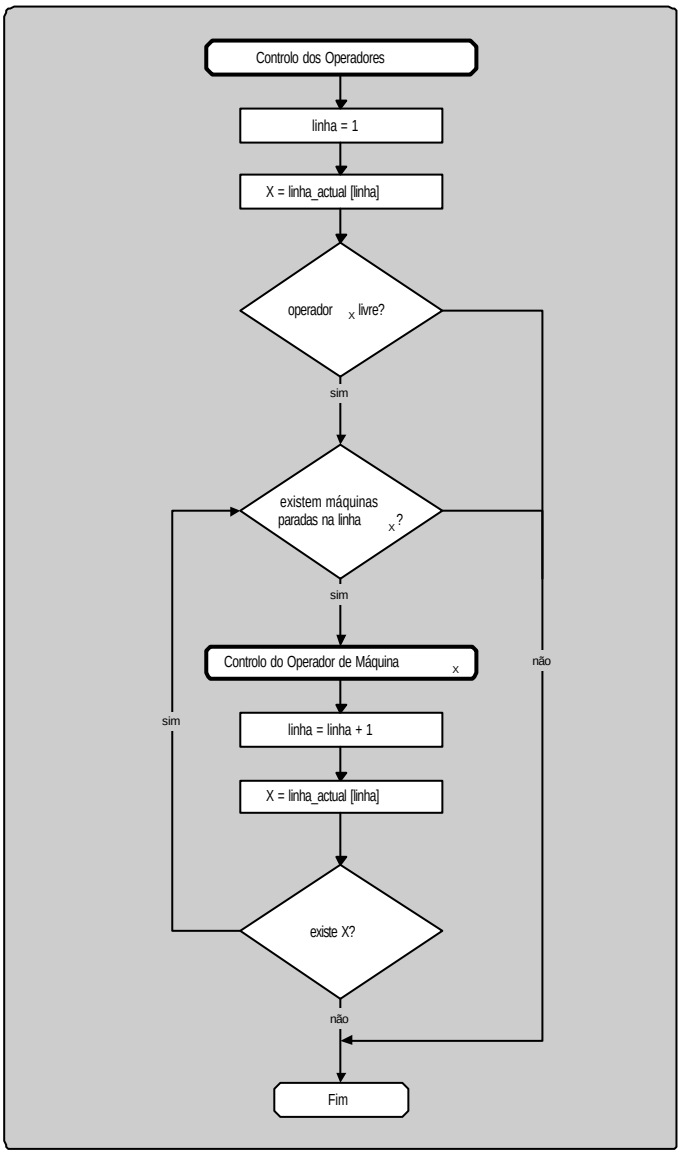


Figura 3.6 – Diagrama geral de controlo dos operadores de máquina

Conforme se pode constatar no diagrama da Figura 3.6, as linhas não necessitam de ser sequenciais. Num ambiente de simulação convém salvaguardar algumas situações como ofacto de ser permitido eliminar linhas num determinado modelo, ou evitando a elaboração de um novo modelo igual ao anterior com a excepção das linhas a eliminar. Daí que as linhas existentes sejam guardadas numa lista ordenada de forma crescente, em que a primeira posição dessa lista é ocupada pelo número da primeira linha existente, a segunda posição pela linha com o segundo número mais baixo e assim sucessivamente.

Controlo efectuado sobre a Máquina

O controlo da máquina, representado na Figura 3.7, relaciona-se com o estado em que a máquina se encontra. Está, principalmente, orientado a duas entidades, ao operador de máquina e à equipa de manutenção com o objectivo de aumentar a eficiência das máquinas reduzindo a duração da paragem. Quando uma máquina pára, porque terminou o lote, envia um sinal requisitando a presença do operador de máquina. Se a paragem da máquina foi originada por uma avaria, solicita a presença da equipa de manutenção. Depois da execução das tarefas do operador de máquina ou da equipa de manutenção, a máquina passa para o estado de *Trabalhar*.

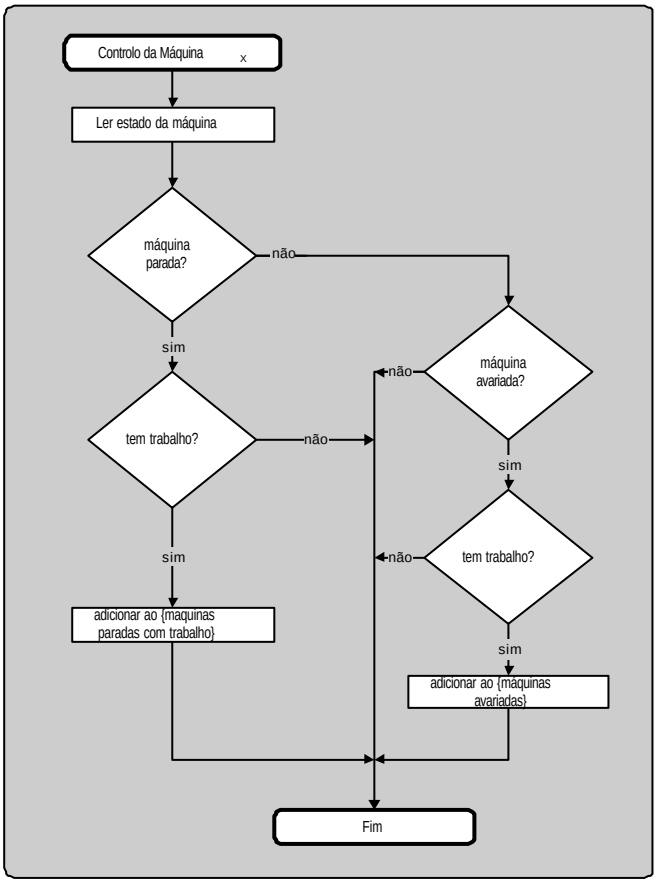


Figura 3.7 – Diagrama geral de controlo da máquina

3.3 Gestão dos Materiais

A gestão de materiais está presente em qualquer tipo de organização e é definida como o grupo de funções de gestão que dá apoio ao ciclo completo do fluxo de materiais: compras, planeamento e controlo da produção, processamento, armazenamento e distribuição do produto.

Como principal objectivo, a gestão de materiais pretende gerir, adequadamente e de forma eficiente e efectiva, o fluxo de materiais característico de uma empresa. Esta gestão é frequentemente caracterizada pela necessidade de tomadas de decisões rápidas que se reflectem com maior ou menor impacto nos diferentes sectores da empresa e que, consequentemente, devem ser profundamente analisadas. Dois factores importantes regem as tomadas de decisão a nível dos materiais: a importância dos materiais e o seu inventário.

Quanto ao tipo de utilização, o artigo comprado destina-se, num primeiro tempo, a ser armazenado, para ser consumido ou aplicado num segundo tempo. Este tipo de utilização faz com que os artigos ou produtos se designem por stock, requerendo uma gestão específica relacionada com a gestão de compras.

Um sistema de gestão de stocks é o conjunto de políticas e controlos que avaliam os níveis de inventário e determinam que níveis é que devem ser mantidos, quando é que o *stock* dever ser reposto e qual a dimensão das ordens de reposição.

Componentes da Gestão de Stocks [SEQU94]

A gestão de stocks tem as seguintes componentes: Gestão Provisional de Stocks, Gestão Administrativa de Stocks e a Gestão Física de Stocks.

Quando se pretende decidir o que é necessário comprar para stock, que quantidades se deve comprar e quando o fazer, ter-se-á que prever a utilização, consumo ou vendas. A decisão de compra tem por base a previsão de consumo bem como outros parâmetros condicionantes, daí se designar por *Gestão Provisional de Stocks* todo o processo que conduz à compra para stock.

Mas para gerir uma existência em armazém (stock) é indispensável conhecer o que existe, quanto existe, e onde se encontra. Esse conhecimento obtém-se a partir do registo de movimentos (entradas e saídas) e da própria identificação de cada artigo ou produto. A *Gestão Administrativa de Stocks* baseia-se nas medidas de controlo de existências,

acompanhando a par e passo, tudo o que se passa em termos de movimentação e que altere a situação anterior. Trata-se de assegurar o inventário permanente das existências.

Por último, mas não menos importante, está-se perante o facto de que cada artigo de stock passa pelo armazém onde permanece algum tempo (o menos possível), mas que a forma como passa ou está em armazém, em tudo se prende com a própria organização e gestão do armazenamento. Assim, a forma como se decide a arrumação e a movimentação, isto é, a organização do armazém, é designada por *Gestão Física de Stocks* (ou gestão material) e condiciona decisivamente, como qualquer das outras componentes, a gestão de stocks e, consequentemente, a própria gestão dos aprovisionamentos.

Na essência dos conceitos, poder-se-á dizer que cada uma das três componentes da gestão de stocks tem objectivos complementares entre si e complementares também com a finalidade da própria função aprovisionamento, podendo sintetizar-se da seguinte forma:

Gestão Provisional de Stocks, permite determinar:

O que comprar
Quanto comprar
Quando comprar

Tendo por objectivo assegurar que o utilizador interno disponha dos artigos de que necessita nas quantidades e datas exactas.

Gestão Administrativa de Stocks, permite conhecer permanentemente:

O que existe
Quanto existe
Onde está

Para que seja possível controlar as existências e fornecer indicações práticas que permitam apoiar a gestão provisional de stocks.

Gestão Física de Stocks, permite:

Recepcionar os produtos comprados
Armazená-los e movimentá-los
Distribuir aos utilizadores internos

Assegurando que o que foi comprado foi recebido; que durante a passagem pelo armazém pelo armazém os artigos foram devidamente protegidos e conservados; que os

utilizadores puderam satisfazer os seus pedidos em tempo útil.

Em paralelo com a Gestão de Stocks (nas três componentes), temos a:

Gestão de Compras, permite decidir:

A quem comprar

Como comprar

Assegurando a aquisição dos artigos e produtos necessários em tempo útil, nas melhores condições de preço, qualidade, prazo de entrega e prazo de pagamento.

Podere-se-à, deste modo concluir, que todas as actividades antes indicadas se integram no objectivo da gestão de aprovisionamentos, dando-lhe significado concreto em todas as situações.

Dimensionamento de Lotes

A técnica normalmente usada para dimensionamento de lotes na indústria é denominada por MRP (*Material Requirement Planning*, ou a sua tradução Planificação das Necessidades de Materiais).

O sistema MRP converte o programa director de produção em planos de lançamentos para a produção, montagem e/ou compra, faseados no tempo, para todos os componentes e subconjuntos de montagem. O que está em causa na problemática do dimensionamento de lotes é saber se será boa política fazer encomendas de acordo com os lançamentos previstos pelo MRP, ou se por outro lado, haverá alguma vantagem em termos económicos em usar outra política qualquer.

O sistema objectiva a redução de stocks, de tempos de percurso e melhoria do cumprimento de prazos, garantindo o abastecimento das matérias-primas, e ao consumidor os produtos de que precisa.

As características do MRP são:

1. Determinação das necessidades e a fixação de prazos acontecem em geral, sem consideração da limitação da capacidade. Isto exige a apresentação dum sistema de produção realizável.
2. A base do sistema MRP, o chamado *sistema levar*, ou seja, as peças são *levadas* após a elaboração do posto seguinte. São também *levadas* aquelas que ainda não são necessárias, o que logicamente origina grandes stocks intermédios.
3. Como o sistema é centralizado, pressupõe dados fiáveis e actuais, pessoal qualificado e disciplinado.

De seguida seguem-se algumas técnicas de MRP ou dimensionamento de lotes de produção implementadas no projecto. Além de uma breve descrição da sua aplicação e procedimento é apresentado um exemplo, por técnica, para melhor compreensão do seu funcionamento.

Lot-For-Lot é o método mais simples de todos. Encomenda exactamente a quantidade requerida em cada período. Minimiza o inventário, mas maximiza o número das ordens colocadas (assim que pode ser caro se os custos de encomenda forem significativos). Pode ser interessante para produtos com custos de inventário elevado, custos de encomenda baixos, para produtos caros ou produtos que são necessários ocasionalmente [JERS88].

Lot-For-Lot

Periodo	1	2	3	4	5	6	7	8	9	10	11	12	Totais
Pedidos	**10**	**62**	**12**	**130**	**154**	**60**	**43**	**52**	**129**	**72**	**42**	**41**	**807**
Lançamentos	10	62	12	130	154	60	43	52	129	72	42	41	**807**
Inventario Inicial	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Inventario Final	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Custo Posse	0	0	0	0	0	0	0	0	0	0	0	0	**0**
Custo Encomenda	54	54	54	54	54	54	54	54	54	54	54	54	**648**

Custo Total = 648

Figura 3.8 – Aplicação da técnica *Lot-For-Lot*

Periodic Order Quantity (POQ) determina o número de períodos que cada ordem vai satisfazer. O POQ usa a mesma lógica do EOQ mas converte o EOQ num número inteiro de períodos. O resultado é uma ordem fixa para um número fixo de períodos. O intervalo económico é calculado através da média dos pedidos arredondada ao valor inteiro mais perto maior do que zero. O tamanho dos lotes é a soma dos valores acumulados das necessidades para cada intervalo de períodos [JERS88].

$$POQ = \frac{EOQ}{\overline{R}} = \sqrt{\frac{2C}{\overline{R}Ph}} \qquad com \quad POQ = Period\ Order\ Quantity$$

R = média dos pedidos
h = custo de posse
P = custo de compra por unidade
C = custo de encomenda

Figura 3.9 – Procedimento *Period Order Quantity*

Period Order Quantity

Periodo	1	2	3	4	5	6	7	8	9	10	11	12	Totais
Pedidos	**10**	**62**	**12**	**130**	**154**	**60**	**43**	**52**	**129**	**72**	**42**	**41**	**807**
Lançamentos	72	0	142	0	214	0	95	0	201	0	83	0	**807**
Inventario Inicial	0	62	0	130	r d e r	60	0	52	0	72	0	41	**417**
Inventario Final	62	0	130	0	60	0	52	0	72	0	41	0	**417**
Custo Posse	24,8	0	52	0	24	0	20,8	0	28,8	0	16,4	0	**166,8**
Custo Encomenda	54	0	54	0	54	0	54	0	54	0			**270**

Custo Total = 490,8

Figura 3.9a – Aplicação da técnica *Period Order Quantity*

Wagner-Whitin Algorithm técnica matematicamente complexa de dimensionamento de lotes, que avalia todas as formas possíveis de se efectuar um pedido para cobrir as necessidades em cada período do horizonte de planeamento e chegar a uma estratégia óptima de pedido de todo o programa de necessidades [JERS88].

$$Q_i = \sum_{k=i}^{j} D_k \quad (j \geq i)$$

$$I_i Q_{i+1} = 0 \quad (i = 0....n)$$

Figura 3.10 – Algoritmo *Wagner-Whitin*

Wagner Whitin (optimo)

Periodo	1	2	3	4	5	6	7	8	9	10	11	12	Totais
Pedidos	**10**	**62**	**12**	**130**	**154**	**60**	**43**	**52**	**129**	**72**	**42**	**41**	**807**
Lançamentos	84	0	0	130	214	0	95	0	201	0	83	0	**807**
Inventario Inicial	0	74	12	0	0	60	0	52	0	72	0	41	**311**
Inventario Final	74	12	0	0	60	0	52	0	72	0	41	0	**311**
Custo Posse	29,6	4,8	0	0	24	0	20,8	0	28,8	0	16,4	0	**124,4**
Custo Encomenda	54	0	0	54	54	0	54	0	54	0	54	0	**324**

Custo Total = 448,4

Figura 3.10a – Aplicação da técnica *Wagner-Whitin*

Silver-meal, é uma heurística que equilibra o custo de encomenda com o custo de posse de inventário para dar uma solução mínima próxima do custo do inventário O procedimento calcula o número de períodos, acumulando as suas necessidades, para efectuar um pedido correspondente a esse intervalo [JERS88].

$$K(j,m) = \frac{1}{m}\left(C + hD_{j+1} + 2hD_{j+2} + \ldots + (m-1)hD_{j+m-1}\right)$$

$T = número\ de\ períodos$

$Passo1: J = 1$

$Passo2: calcular\ K(j,m)\ Para\ m = 1..T - j + 1\ e\ parar\ quando K(j,m+1) > K(j,m)$

$Passo3: Fazer\ lançamentos\ até\ m\ períodos$

$= D_j + D_{j+1} + \ldots + D_{j+m-1}$

$Passo4: j = j + m\ se\ j \geq T\ FIM\ senão\ Passo2$

Figura 3.11 – Algoritmo *Silver-Meal*

Silver-Meal

Periodo	1	2	3	4	5	6	7	8	9	10	11	12	Totais
Pedidos	**10**	**62**	**12**	**130**	**154**	**60**	**43**	**52**	**129**	**72**	**42**	**41**	**807**
Lançamentos	84	0	0	130	257	0	0	181	0	155	0	0	**807**
Inventario Inicial	0	74	12	0	0	103	43	0	129	0	83	41	**485**
Inventario Final	74	12	0	0	103	43	0	129	0	83	41	0	**485**
Custo Posse	29,6	4,8	0	0	41,2	17,2	0	51,6	0	33,2	16,4	0	**194**
Custo Encomenda	54	0	0	54	54	0	0	54	0	54	0	0	**270**

Custo Total = 464

Figura 3.11a – Aplicação da técnica *Silver-Meal*

A heurística **Least Unit Cost** (LUC), é similar à Silver-meal excepto que em vez de calcular a média sobre os períodos, calcula a média dos custos dos pedidos. Esta heurística determina o custo médio por unidade com a progressão dos períodos [JERS88].

$$K(j,m) = \frac{\left(C + hD_{j+1} + 2hD_{j+2} + \ldots + (m-1)hD_{j+m-1}\right)}{D_j + D_{j+1} + D_{j+2} + \ldots + D_{j+m}}$$

$T = número\ de\ períodos$

$Passo1: J = 1$

$Passo2: calcular\ K(j,m)\ Para\ m = 1..T - j + 1\ e\ parar\ quando K(j,m+1) > K(j,m)$

$Passo3: Fazer\ lançamentos\ até\ m\ períodos$

$= D_j + D_{j+1} + \ldots + D_{j+m-1}$

$Passo4: j = j + m\ se\ j \geq T\ FIM\ senão\ Passo2$

Figura 3.12 – Algoritmo *Least Unit Cost*

Least Unit Cost

Periodo	1	2	3	4	5	6	7	8	9	10	11	12	Totais
Pedidos	**10**	**62**	**12**	**130**	**154**	**60**	**43**	**52**	**129**	**72**	**42**	**41**	**807**
Lançamentos	84	0	0	284	0	103	0	181	0	114	0	41	**807**
Inventario Inicial	0	74	12	0	154	0	43	0	129	0	42	0	**454**
Inventario Final	74	12	0	154	0	43	0	129	0	42	0	0	**454**
Custo Posse	29,6	4,8	0	61,6	0	17,2	0	51,6	0	16,8	0	0	**181,6**
Custo Encomenda	54	0	0	54	0	54	0	54	0	54	0	54	**324**

Custo Total = 505,6

Figura 3.12a – Aplicação da técnica *Least Unit Cost*

Part Period Balancing, selecciona o número dos períodos tal que os custos totais do inventário estão o mais perto possível do custo de encomenda [JERS88].

$$PP_m = D_2 + 2D_3 + ... + (m-1)D_k$$
$$quando\ h * PP_m > C\ então$$
$$procurar\ à\ frente\ até\ h * PP_{m+1} - C > h * PP_m - C$$

Figura 3.13 – Procedimento *Part Period Balancing*

Part Period Balancing

Periodo	1	2	3	4	5	6	7	8	9	10	11	12	Totais
Pedidos	**10**	**62**	**12**	**130**	**154**	**60**	**43**	**52**	**129**	**72**	**42**	**41**	**807**
Lançamentos	84	0	0	284	0	155	0	0	243	0	0	41	**807**
Inventario Inicial	0	74	12	0	154	0	95	52	0	114	42	0	**543**
Inventario Final	74	12	0	154	0	95	52	0	114	42	0	0	**543**
Custo Posse	29,6	4,8	0	61,6	0	38	20,8	0	45,6	16,8	0	0	**217,2**
Custo Encomenda	54	0	0	54	0	54	0	0	54	0	0	54	**270**

Custo Total = 487,2

Figura 3.13a – Aplicação da técnica *Part Period Balancing*

Em todas as figuras, que ilustram as aplicações das técnicas, foram usados os valores de 54 para o custo de encomenda e 0,4 para o custo de posse.

3.4 Escalonamento da Produção

O escalonamento pode ser visto como uma ferramenta para modelação e resolução de muitos problemas da vida real. Dado um problema onde existam tarefas a executar e uma quantidade determinada de recursos, um escalonamento é uma função que aloca as tarefas aos recursos. Diz-se que um escalonamento é válido se ele obedece as restrições do problema (por exemplo a cada instante apenas uma tarefa pode utilizar um mesmo recurso). Além de encontrar escalonamentos válidos, também se está interessado em minimizar algum critério de optimização (por exemplo encontrar entre todos os escalonamentos válidos um com o menor tempo de execução).

Principais Conceitos

O termo *escalonamento* (*scheduling*) identifica o procedimento de ordenar tarefas na fila de espera. Uma escala de execução (*schedule*) é então uma ordenação ou lista que indica a ordem de ocupação do recurso por um conjunto de tarefas disponíveis na fila de espera. O *escalonador* (*scheduler*) é o componente do sistema responsável em tempo de execução pela gestão do recurso. É o *escalonador* que implementa uma política de escalonamento ao ordenar para execução sobre o recurso um conjunto de tarefas.

Políticas de escalonamento definem critérios ou regras para a ordenação das tarefas ou balanceamento dos recursos. Os *escalonadores* utilizando então essas políticas produzem escalas que, se forem realizáveis, garantem o cumprimento das restrições impostas às tarefas e aos recursos. Uma escala é dita óptima se o conjunto de tarefas, de acordo com os critérios preestabelecidos pela política de escalonamento, é a melhor possível no atendimento das restrições impostas.

Um problema de escalonamento, na sua forma geral, envolve um conjunto de recursos e um conjunto de tarefas especificadas segundo um modelo de tarefas definindo as diversas restrições. O escalonamento de tempo real, na sua forma geral, é identificado como um problema intratável (NP-completo) [AUDS90], [GARC79].

Muito frequentemente os algoritmos existentes representam uma solução polinomial para um problema de escalonamento particular, onde um conjunto de hipóteses podem expressar simplificações no modelo de tarefas ou mesmo na arquitectura do sistema, no sentido de diminuir a complexidade do problema. Quando nenhuma simplificação é usada para abrandar a complexidade no escalonamento, uma heurística é usada para encontrar uma escala realizável ainda que não sendo óptima mas que garanta as restrições do problema.

Um algoritmo é identificado com óptimo se minimiza/maximiza algum custo ou métrica definida sobre a sua classe de problema. Quando nenhum custo ou métrica é definido, a única preocupação é então uma escala realizável.

Divisão dos Algoritmos de Escalonamento

Para atingir seus objectivos, os modelos de escalonamento podem incluir estratégias migratórias. Por migração, neste propósito, refere-se à possibilidade de um processo mudar de máquina durante a execução. A migração de processos procura reduzir o tempo gasto pelos processos na fila de espera por processamento. O custo mais importante para se conseguir isto é uma sobrecarga no sistema de comunicação. A migração facilita a distribuição de carga entre os recursos.

Pode-se dividir os algoritmos de escalonamento em dois grandes grupos: os de distribuição estática e os de distribuição dinâmica. Os estáticos tratam do problema de repartir a carga, isto é, procuram distribuir equitativamente a carga de processamento antes mesmo do início da execução dos processos. Os dinâmicos tratam do problema de balanceamento de carga, ou seja, procuram ajustar as oscilações na carga de processamento durante a execução dos processos, para que todos os recursos tenham as suas potencialidades exploradas ao máximo.

Escalonamento para Sistemas de Recursos Paralelos

O escalonamento de tarefas é essencial para o funcionamento de sistemas de recursos paralelos. Escalonamento de tarefas dentro de um conjunto de recursos paralelos é um problema bem definido e documentado na literatura. Entretanto a maioria das técnicas disponíveis são baseadas em heurísticas que resolvem certas instâncias do problema de escalonamento eficientemente e numa quantia razoável de tempo.

Uma outra forma alternativa de se escalonar processos é através da programação linear (PL). A PL e o Método Simplex foram desenvolvidos por Dantzig em 1947, é frequentemente usada como parte de vários esquemas para solucionar problemas de programação não-linear, problemas discretos, problemas combinatórios, problemas de controlo e optimalidade e programação sob incerteza.

Um problema de PL é um problema de minimização ou maximização de uma função linear na presença de restrições lineares de uma inequação e/ou equação.

Programação Linear e Programação Inteira

A PL utiliza modelos matemáticos em função de variáveis contínuas, para representar comportamentos de sistemas reais. Entretanto, na vida real é muito comum que as variáveis precisem assumir valores inteiros e não contínuos. Quando se impõe a restrição de que as variáveis assumam valores inteiros o problema pode ficar muito mais difícil que a ideia natural de simplesmente arredondar os valores nem sempre traz bons resultados. Nestas circunstâncias, está-se perante modelos de programação inteira.

Estratégias de Escalonamento

Considerações Iniciais

Esta é uma área de actuação onde se pode definir uma enorme quantidade de estratégias sob diversos pontos de vista, quer ao nível das máquinas, quer ao nível da matéria-prima ou do produto. Porém, por muitas estratégias que sejam elaboradas corre-se sempre o risco de omitir alguma.

A implementação de estratégias de escalonamento foi elaborada de forma a minimizar esse risco. Assim, montou-se um esquema que permitisse combinar as estratégias pretendidas para as máquinas com as que se preferiam aplicadas ao produto. Além destes dois conjuntos de estratégias foram elaboradas outras duas, uma de aplicação geral, aplicada quando se pretende maximizar o lucro, e outra especifica referente à matéria-prima.

Convém referir que todas as estratégias elaboradas para as máquinas, para os produtos, para a matéria-prima e a que maximiza o lucro, podem ser combinadas e usadas como uma única estratégia de escalonamento.

Para melhor compreensão do funcionamento das estratégias será apresentado, além dos modelos matemáticos representativos das estratégias, um exemplo de aplicação por cada estratégia. Para a apresentação dos exemplos foram usados os seguintes dados:

As letras maiúsculas que aparecem na definição matemática do modelo representam parâmetros. Estes valores são retirados das características das máquinas, dos produtos e outras definições, todas elas inseridas previamente pelo utilizador, apresentados sob a forma de tabelas.

Produtos \ Máquinas	1	2	3	4	5
1	0	1	1	1	1
2	1	1	0	1	1
3	0	1	1	0	0

Tabela 3.3 – Restrições(0) / Permissões(1)

Máquinas	1	2	3	4	5
	5	10	2	20	20

Tabela 3.4 – Velocidade das máquinas

	Produtos 1	2	3
Tempo	20	10	30

Tabela 3.5 – Tempo de processo

	Produtos 1	2	3
Quant.	90	65	30
Lucro	10	20	40

Tabela 3.6 – Ordens de produção e lucro

Produtos \ Componentes	FN	FB	EN	ER	FR
1	10	5	0	2	0
2	8	4	4	0	0
3	0	8	0	6	15
Max	999	888	777	666	555
Min	10	11	22	33	44

Tabela 3.7 – Componentes

		Max. Quant.	Min. Quant.	Max Tempo	Min Tempo
Maquinas	1	100	5	500	20
	2	120	4	400	10
	3	300	5	600	15
	4	250	4	300	25
	5	100	5	200	20
Produtos	1	100	5	500	20
	2	120	4	400	10
	3	300	5	600	15

Tabela 3.8 – Limites

A Tabela 3.3 representa os produtos que podem ser produzidos pelas máquinas, no caso em que o cruzamento de linha (produto) com a coluna (máquina) da tabela tem 1. Se o elemento (*x*, *y*) da tabela tiver 0 (zero) a maquina *y* não pode produzir o produto *x*.

Na Tabela 3.4 pode-se observar a velocidade das máquinas, isto é, o factor multiplicativo com que a máquina vai executar os produtos. Por exemplo, a máquina 2 com velocidade 10 vai executar o produto 1, de tempo de processo 20, em 2 segundos.

O tempo de processo de cada produto é apresentado na Tabela 3.5. O tempo é em segundos e refere-se ao tempo que demora a executar uma unidade.

Ainda relativamente aos produtos tem-se a Tabelas 3.6, que além de nos indicar o lucro obtido por cada unidade produzida, informa-nos das ordens a produzir.

Na Tabela 3.7, tem-se uma relação da matéria-prima por produto, e os limites (Max e Min) de produção para que cada componente.

A Tabela 3.8 determina as quantidades mínimas e máximas para as máquinas e produtos.

A mesma lógica pode ser aplicada se, em vez de usar quantidades, se preferir usar os limites em função do tempo de execução dos produtos. Neste caso, o tempo real do produto

por máquina será $\frac{tempo\,de\,processo}{velocidade\,da\,máquina}$.

$$\text{Max } f(X)= \sum_{i}^{m}\sum_{j}^{p} x_{i,j} * R_{i,j}$$

$$\text{sujeito a: } \sum_{i=1}^{m}\left(\sum_{j}^{p} x_{i,j} * R_{i,j} \le Q_i\right)$$

$x_{i,j} = produto\,i\,elaborado\,na\,máquina\;j$

$p = número\,de\;produtos$

$m = número\,de\,máquinas$

$$R = \begin{cases} 0, se\,máquina\;j\,não\,executa\,produto\,i \\ 1, se\,máquina\;j\,executa\,produto\,i \end{cases}$$

$Q = quantidades\,de\,ordens$

Figura 3.14 – Estratégia elementar

O objectivo do modelo representado na Figura 3.14 é atribuir a quantidade máxima de ordens nas máquinas. A constante *R*, retirada da Tabela 3.3, garante a exequibilidade, ou não, do produto em determinada máquina. A restrição garante que a quantidade distribuída pelas máquinas não excede as quantidades planeadas para os produtos (Q), retiradas da Tabela 3.6. A função objectivo e a restrição estarão sempre presente em todas as estratégias.

$$\max\; x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} + x_{2,1} + x_{2,2} + x_{2,4} + x_{2,5} + x_{3,2} + x_{3,3}$$

$$\begin{aligned} x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} &\le 90 \\ x_{2,1} + x_{2,2} + x_{2,4} + x_{2,5} &\le 65 \\ x_{3,2} + x_{3,3} &\le 30 \end{aligned}$$

Figura 3.14a – Exemplo da estratégia elementar

A ideia expressa no modelo acima, estará sempre presente, quaisquer que sejam as estratégias escolhidas para o escalonamento da produção. Se não se escolher nenhuma estratégia, o modelo representado pela Figura 3.14 será o modelo definitivo de escalonamento. Se se escolher mais estratégias, elas serão acrescentadas a esta estratégia.

Estratégia 1 – Máquinas com Planificação Igual

$$\sum_{j=1}^{m-1}\left(\sum_{i=1}^{p}\left(x_{i,j} * R_{i,j} - x_{i,j+1} * R_{i,j+1}\right) = 0\right)$$

Figura 3.15 – Estratégia 1

O objectivo desta estratégia prende-se à ideia de equilibrar as quantidades planificadas por todas as máquinas, independentemente do tipo de produtos. Distribuir equitativamente a planificação pelas máquinas parece ser boa ideia – técnica muito utilizada em ambientes fabris. Porém, a própria dependência da presença do operador de máquina, sempre que termina um lote, os tempos de *setup* da máquina e a probabilidade de falha da máquina podem afastar o comportamento do modelo do equilíbrio inicialmente referido.

Exemplo da Estratégia 1

$$\begin{aligned}
-x_{1,2} + x_{2,1} - x_{2,2} - x_{3,2} &= 0\\
x_{1,2} - x_{1,3} + x_{2,2} + x_{3,2} - x_{3,3} &= 0\\
x_{1,3} - x_{1,4} - x_{2,4} + x_{3,3} &= 0\\
x_{1,4} - x_{1,5} + x_{2,4} - x_{2,5} &= 0
\end{aligned}$$

Figura 3.15a – Exemplo da estratégia 1

Estratégia 2 – Máquinas com Tempo de Produção Igual

$$\sum_{j=1}^{m-1}\left(\sum_{i=1}^{p}\left(x_{i,j} * W_0 - x_{i,j+1} * W_1\right) = 0\right)$$

$$W_0 = R_{i,j} * \frac{t_{i,j}}{v_{i,j}} \qquad W_1 = R_{i,j+1} * \frac{t_{i,j+1}}{v_{i,j+1}}$$

Figura 3.16 – Estratégia 2

Esta estratégia faz uma distribuição de quantidades de produtos para que a soma do tempo de produção dessas quantidades seja igual para todas as máquinas. O cálculo do tempo que cada produto demora em cada máquina, impõe a divisão de processo (Tabela 3.5) pela velocidade da máquina que o produzirá (Tabela 3.4).

Exemplo da Estratégia 2

$$\begin{array}{rrrrrrrrr} & -2x_{1,2} & +2x_{2,1} & -x_{2,2} & & & -3x_{3,2} & = & 0 \\ 2x_{1,2} & -10x_{1,3} & +x_{2,2} & & & +3x_{3,2} & -15x_{3,3} & = & 0 \\ 10x_{1,3} & -x_{1,4} & & -0.5x_{2,4} & +15x_{3,3} & & & = & 0 \\ x_{1,4} & -x_{1,5} & +0.5x_{2,4} & -0.5x_{2,5} & & & & = & 0 \end{array}$$

Figura 3.16a – Exemplo da estratégia 2

Estratégia 3 – Máquinas com Limites de Quantidades

$$\sum_{j=1}^{m}\left(\sum_{i=1}^{p}\left(x_{i,j} * R_{i,j}\right) \geq MINQ_j\right)$$

$$\sum_{j=1}^{m}\left(\sum_{i=1}^{p}\left(x_{i,j} * R_{i,j}\right) \leq MAXQ_j\right)$$

Figura 3.17 – Estratégia 3

Nesta estratégia está patente o interesse em garantir que as quantidades a produzir de determinado produto estejam no intervalo [MINQ, MAXQ].

Exemplo da Estratégia 3

$$\begin{array}{rrrr} & x_{2,1} & & \geq 5 \\ x_{1,2} & +x_{2,2} & +x_{3,2} & \geq 4 \\ x_{1,3} & & +x_{3,3} & \geq 5 \\ x_{1,4} & +x_{2,4} & & \geq 4 \\ x_{1,5} & +x_{2,5} & & \geq 5 \end{array} \qquad \begin{array}{rrrr} & x_{2,1} & & \leq 100 \\ x_{1,2} & +x_{2,2} & +x_{3,2} & \leq 120 \\ x_{1,3} & & +x_{3,3} & \leq 300 \\ x_{1,4} & +x_{2,4} & & \leq 250 \\ x_{1,5} & +x_{2,5} & & \leq 100 \end{array}$$

Figura 3.17a – Exemplo da estratégia 3

Estratégia 4 – Máquinas com Limites de Tempo

$$\sum_{j=1}^{m}\left(\sum_{i=1}^{p}\left(x_{i,j} * W_0\right) \leq MAXT_j\right)$$
$$\sum_{j=1}^{m}\left(\sum_{i=1}^{p}\left(x_{i,j} * W_0\right) \geq MINT_j\right)$$
$$W_0 = R_{i,j} * \frac{t_{i,j}}{v_{i,j}}$$

Figura 3.18 – Estratégia 4

Nesta estratégia (semelhante à anterior) as limitações estão associadas aos tempos de produção das máquinas.

Exemplo da Estratégia 4

$$\begin{array}{rcl} 2x_{2,1} & \geq & 20 \\ 2x_{1,2} + x_{2,2} + 3x_{3,2} & \geq & 10 \\ 10x_{1,3} + 15x_{3,3} & \geq & 15 \\ x_{1,4} + 0.5x_{2,4} & \geq & 25 \\ x_{1,5} + 0.5x_{2,5} & \geq & 20 \end{array} \qquad \begin{array}{rcl} 2x_{2,1} & \leq & 500 \\ 2x_{1,2} + x_{2,2} + 3x_{3,2} & \leq & 400 \\ 10x_{1,3} + 15x_{3,3} & \leq & 600 \\ x_{1,4} + 0.5x_{2,4} & \leq & 300 \\ x_{1,5} + 0.5x_{2,5} & \leq & 200 \end{array}$$

Figura 3.18a – Exemplo da estratégia 4

Estratégia 5 – Máquinas com o Mesmo Lucro

$$\sum_{j=1}^{m-1}\left(\sum_{i=1}^{p}\left(x_{i,j} * K_0 - x_{i,j+1} * K_1\right) = 0\right)$$
$$K_0 = R_{i,j} * P_i$$
$$K_1 = R_{i,j+1} * P_i$$

Figura 3.19 – Estratégia 5

Propõe-se nesta estratégia uma distribuição de quantidades produzidas pelas máquinas por forma a harmonizar o custo associado a cada.

Exemplo da Estratégia 5

$$\begin{aligned}
&-10x_{1,2} + 20x_{2,1} - 20x_{2,2} - 40x_{3,2} = 0\\
10x_{1,2} &- 10x_{1,3} + 20x_{2,2} + 40x_{3,2} - 40x_{3,3} = 0\\
10x_{1,3} &- 10x_{1,4} - 20x_{2,4} + 40x_{3,3} = 0\\
10x_{1,4} &- 10x_{1,5} + 20x_{2,4} - 20x_{2,5} = 0
\end{aligned}$$

Figura 3.19a – Exemplo da estratégia 5

Estratégia 6 – Produtos com Planificação Igual

$$\sum_{i=1}^{p-1}\left(\sum_{j=1}^{m}\left(x_{i,j} * R_{i,j} - x_{i+1,j} * R_{i+1,j}\right) = 0\right)$$

Figura 3.20 – Estratégia 6

Esta estratégia é direccionada aos produtos, utilizando, para cada produto, o mesmo padrão de distribuição pelas máquinas.

Exemplo da Estratégia 6

$$\begin{aligned}
x_{1,2} + x_{1,3} + x_{1,5} - x_{2,1} - x_{2,2} - x_{2,4} - x_{2,5} &= 0\\
x_{2,2} + x_{2,4} + x_{2,5} - x_{3,2} - x_{3,3} &= 0
\end{aligned}$$

Figura 3.20a – Exemplo da estratégia 6

Estratégia 7 – Produtos com Tempo de Produção Igual

$$\sum_{i=1}^{p-1}\left(\sum_{j=1}^{m}\left(x_{i,j} * W_0 - x_{i+1,j} * W_1\right) = 0\right)$$

$$W_0 = R_{i,j} * \frac{t_{i,j}}{v_{i,j}} \qquad W_0 = R_{i+1,j} * \frac{t_{i+1,j}}{v_{i+1,j}}$$

Figura 3.21 – Estratégia 7

Esta estratégia equilibra a distribuição de quantidades de produtos para que, dependendo das máquinas a que forem atribuídos, tenham o mesmo tempo de produção.

Exemplo da Estratégia 7

$$\begin{aligned} 2x_{1,2} + 10x_{1,3} + x_{1,5} - 2x_{2,1} - x_{2,2} - 0.5x_{2,4} - 0.5x_{2,5} &= 0 \\ x_{2,2} + 0.5x_{2,4} + 0.5x_{2,5} - 3x_{3,2} - 15x_{3,3} &= 0 \end{aligned}$$

Figura 3.21a – Exemplo da estratégia 7

Estratégia 8 – Produtos com Limites de Quantidades

$$\sum_{i=1}^{p}\left(\sum_{j=1}^{m}\left(x_{i,j} * R_{i,j}\right) \geq MINQp_i\right)$$

$$\sum_{i=1}^{p}\left(\sum_{j=1}^{m}\left(x_{i,j} * R_{i,j}\right) \leq MAXQp_i\right)$$

Figura 3.22 – Estratégia 8

Se se pretender garantir que determinada quantidade de um produto seja executada, então esta estratégia é a mais indicada. Assim, pode-se garantir que são produzidas as quantidades que correspondem ao intervalo ao [MINQ, MAXQ].

Exemplo da Estratégia 8

$$\begin{aligned} x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} &\geq 5 \\ x_{2,1} + x_{2,2} + x_{2,4} + x_{2,5} &\geq 4 \\ x_{3,2} + x_{3,3} &\geq 5 \end{aligned} \qquad \begin{aligned} x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} &\leq 100 \\ x_{2,1} + x_{2,2} + x_{2,4} + x_{2,5} &\leq 120 \\ x_{3,2} + x_{3,3} &\leq 300 \end{aligned}$$

Figura 3.22a – Exemplo da estratégia 8

Estratégia 9 – Produtos com Limites de Tempo

$$\sum_{i=1}^{p}\left(\sum_{j=1}^{m}\left(x_{i,j} * W_0\right) \geq MINTp_i\right)$$
$$\sum_{i=1}^{p}\left(\sum_{j=1}^{m}\left(x_{i,j} * W_0\right) \leq MAXTp_i\right)$$
$$W_0 = R_{i,j} * \frac{t_{i,j}}{v_{i,j}}$$

Figura 3.23 – Estratégia 9

Esta estratégia (semelhante à anterior) garante tempos de produção específicos.

Exemplo da Estratégia 9

$$\begin{array}{rcl} 2x_{1,2} + 10x_{1,3} + x_{1,4} + x_{1,5} & \geq & 20 \\ 2x_{2,1} + x_{2,2} + 0.5x_{2,4} + 0.5x_{2,5} & \geq & 10 \\ 3x_{3,2} + 15x_{3,3} & \geq & 15 \end{array} \qquad \begin{array}{rcl} 2x_{1,2} + 10x_{1,3} + x_{1,4} + x_{1,5} & \leq & 500 \\ 2x_{2,1} + x_{2,2} + 0.5x_{2,4} + 0.5x_{2,5} & \leq & 400 \\ 3x_{3,2} + 15x_{3,3} & \leq & 600 \end{array}$$

Figura 3.23a – Exemplo da estratégia 9

Estratégia 10 – Produtos com o Mesmo Lucro

$$\sum_{i=1}^{p-1}\left(\sum_{j=1}^{m}\left(x_{i,j} * K_0 - x_{i+1,j} * K_1\right) = 0\right)$$
$$K_0 = R_{i,j} * P_i$$
$$K_1 = R_{i+1,j} * P_i$$

Figura 3.24 – Estratégia 10

Esta estratégia prevê igualdade nos lucros associados a cada produto, independentemente da diversidade de produtos e respectivas quantidades.

Exemplo da Estratégia 10

$$\begin{aligned} 10x_{1,2} + 10x_{1,3} + 10x_{1,5} - 20x_{2,1} - 20x_{2,2} - 20x_{2,4} - 20x_{2,5} &= 0 \\ 20x_{2,2} + 20x_{2,4} + 20x_{2,5} - 40x_{3,2} - 40x_{3,3} &= 0 \end{aligned}$$

Figura 3.24a – Exemplo da estratégia 10

Estratégia 11 – Limites nos Componentes

$$\sum_{i=1}^{c}\left(\left(\sum_{j=1}^{p}\sum_{n=1}^{m}B_{i,j,n}\right)\geq BMIN_i\right)$$

$$\sum_{i=1}^{c}\left(\left(\sum_{j=1}^{p}\sum_{n=1}^{m}B_{i,j,n}\right)\leq BMAX_i\right)$$

$$B_{i,j,n} = R_{i,n} * BOM_{i,j}$$

Figura 3.25 – Estratégia 11

Nesta estratégia pode-se fazer depender a planificação com as existências de matérias-primas. Neste caso pode-se limitar o consumo de determinados componentes (BMAX), e/ou impor consumos mínimos de outros (BMIN).

Exemplo da Estratégia 11

$$\begin{aligned}
10x_{1,2} + 10x_{1,3} + 10x_{1,4} + 10x_{1,5} + 8x_{2,1} + 8x_{2,2} + 8x_{2,4} + 8x_{2,5} &\geq 10 \\
5x_{1,2} + 5x_{1,3} + 5x_{1,4} + 5x_{1,5} + 4x_{2,1} + 4x_{2,2} + 4x_{2,4} + 4x_{2,5} + 8x_{3,2} + 8x_{3,3} &\geq 11 \\
4x_{2,1} + 4x_{2,2} + 4x_{2,4} + 4x_{2,5} &\geq 22 \\
2x_{1,2} + 2x_{1,3} + 2x_{1,4} + 2x_{1,5} + 6x_{3,2} + 6x_{3,3} &\geq 33 \\
15x_{3,2} + 15x_{3,3} &\geq 44
\end{aligned}$$

$$\begin{aligned}
10x_{1,2} + 10x_{1,3} + 10x_{1,4} + 10x_{1,5} + 8x_{2,1} + 8x_{2,2} + 8x_{2,4} + 8x_{2,5} &\leq 999 \\
5x_{1,2} + 5x_{1,3} + 5x_{1,4} + 5x_{1,5} + 4x_{2,1} + 4x_{2,2} + 4x_{2,4} + 4x_{2,5} + 8x_{3,2} + 8x_{3,3} &\leq 888 \\
4x_{2,1} + 4x_{2,2} + 4x_{2,4} + 4x_{2,5} &\leq 777 \\
2x_{1,2} + 2x_{1,3} + 2x_{1,4} + 2x_{1,5} + 6x_{3,2} + 6x_{3,3} &\leq 666 \\
15x_{3,2} + 15x_{3,3} &\leq 555
\end{aligned}$$

Figura 3.25a – Exemplo da estratégia 11

Estratégia 12 – Maximizar o lucro da Função Objectivo

$$\text{Max } f(X) = \sum_{i}^{p}\sum_{j}^{m} x_{i,j} * K_{i,j}$$
$$K_{i,j} = R_{i,j} * P_i$$

Figura 3.26 – Estratégia 12

Esta estratégia foi desenhada para a situação em que se pretende fazer uma distribuição de quantidades de produtos pelas máquinas com o objectivo de maximizar o lucro.

Esta estratégia substitui a função objectivo da estratégia elementar.

Exemplo da Estratégia 12

$$\max 10x_{1,2} + 10x_{1,3} + 10x_{1,4} + 10x_{1,5} + 20x_{2,1} + 20x_{2,2} + 20x_{2,4} + 20x_{2,5} + 40x_{3,2} + 40x_{3,3}$$

$$\begin{aligned} x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} &\le 90 \\ x_{2,1} + x_{2,2} + x_{2,4} + x_{2,5} &\le 65 \\ x_{3,2} + x_{3,3} &\le 30 \end{aligned}$$

Figura 3.26a – Exemplo da estratégia 12

3.5 Conclusão

Neste capítulo foram abordados os problemas capitais que deram origem a esta dissertação. São apresentadas as estratégias de controlo para a resolução de cada um dos problemas. Houve uma preocupação de criar uma variedade de estratégias para cada um dos assuntos tratados de forma a possibilitar alternativas à gestão.

A variedade de estratégias implementadas permite avaliar e comparar o impacto de diferentes politicas no sistema produtivo. Apesar de não estarem todas as situações contempladas pelas estratégias aqui descritas, o autor está convicto que as estratégias disponíveis quer ao nível da diversidade quer da qualidade potencia a obtenção de bons resultados.

Para cada um dos assuntos tratados o nível de parametrização e flexibilidade foi muito elevado, permitindo ao utilizador definir por completo as características do modelo,

como a matéria-prima, as características para cada tipo de máquina e a quantidade das mesmas para cada linha, definir características individuais dos operadores e o número de linhas que o sistema em causa interessa ter.

A escolha destas estratégias aparece na ferramenta informática enquanto o controlo é garantido pelo *Modelo Base*. A ferramenta foi desenvolvida em VB e o *Modelo Base* está implementado em VBA no *Arena*®.

Capítulo 4

Geração Automática de Modelos de Simulação

Este projecto inclui uma ferramenta, elaborada em VB, que interagindo com o *Arena*®, gera automaticamente modelos de simulação com especificações e parametrizações personalizadas, evitando assim a criação de novos modelos sempre que se pretendam fazer alterações, respeitando o ambiente ou características do sistema produtivo.

A ferramenta foi implementada no sentido de permitir flexibilidade ao nível de materiais, do número de linhas de produção ou do número de máquinas.

Integrado no *Arena*® foi elaborado um código em VBA para implementar um controlo eficiente e uma gestão adequada de todas as variáveis envolvidas em todo o processo. Este código está inserido num ficheiro denominado por *Modelo Base.*

Uma condição fundamental para a tomada de decisões é a obtenção da informação da forma mais simples e adequada possível sobre os principais índices de desempenho envolvidos, por isso, facilitou-se a ferramenta com essa informação de maneira a possibilitar o seu uso noutras aplicações para potenciar melhorias nesses mesmos índices.

De seguida serão descritas a linguagem de simulação usada para a resolução deste problema, o modelo que serve de suporte ao caso de estudo e a ferramenta informática. Todos estes elementos interligados constituem a SAD que se propôs executar.

4.1 A Linguagem de Simulação

Nesta secção pretende-se esclarecer a preferência pelo *Arena*® como sendo a aplicação/ferramenta de simulação mais apropriada para desenvolver este projecto.

O *Arena*® foi considerado por especialistas de renome em simulação como "O mais inovador software de simulação", por unir os recursos de uma linguagem de simulação à facilidade de uso de um simulador, num ambiente gráfico integrado, que contém todos os recursos para modelação de processos, desenho e animação, análise estatística e análise de resultados. A linguagem incorporada ao *Arena*® é o SIMAN.

Arena® é, com efeito, uma aplicação gráfica de alto nível, baseada em linguagem SIMAN, onde se constróiem os modelos juntando determinados ícones ou blocos para definir a lógica de um modelo. Para se criar modelos não é necessário escrever nenhuma linha de código no *Arena*®, pois todo o processo de criação do modelo de simulação é gráfico e visual, e de maneira integrada, não sendo obrigatório conhecer a linguagem SIMAN.

A ligação ao utilizador é o ícone que representa um bloco ou um elemento que executa uma função específica ou um grupo de funções. Os parâmetros exigidos para levar a cabo estas funções podem ser inseridos dentro de uma caixa de diálogo obtida por duplo toque, com o rato, no ícone em causa.

Basicamente, um *template* é uma estrutura já bastante conhecida pelos programadores de qualquer linguagem de alto ou baixo nível: trata-se de uma *subrotina*, um segmento do programa que pode ser executado várias vezes ao longo do processamento com parâmetros diferentes. Tal *subprograma* é mantido separado do programa principal e accionado por este último sempre que a sua acção se fizer necessária. De modo semelhante, um *template* dentro da simulação representa um *submodelo*, que é accionado quando conveniente.

Nesta ferramenta de simulação estão disponibilizados, de origem, inúmeros *templates*. Através da utilização de *templates*, o *Arena*® pode ser transformado facilmente num simulador, específico para varias áreas de aplicação. O utilizador pode criar os seus próprios

templates, incorporando no ambiente corporativo o *know-how*, para uso de outras pessoas, através do uso de objectos/ *template*.

Além das vantagens supracitadas o *Arena*® apresenta:

-compatibilidade com Microsoft Office;

-inclui tecnologia ActiveX®/OLE 30 Automation, o que garante a sua conectividade com outras aplicações tipo Excel, Word, etc. Tendo sido desenvolvido na linguagem Visual C++, utilizando técnicas de orientação a objectos e recursos da Microsoft Foundation Classes (MFC), o *Arena*® faz uso das facilidades oferecidas pelo interface gráfico do ambiente (barras de ferramentas, caixas de diálogo, etc.) [WAGN00].

-o *Arena*® também gera código da simulação na linguagem SIMAN [PEGD91], linguagem específica para simulação, o que permite a depuração de erros da lógica do modelo com facilidade [TAKU97].

-inclui VBA (*Visual Basic for Applications*), permitindo ao utilizador desenvolver rotinas em linguagens de propósito geral, tais como Visual Basic®, FORTRAN, C e C++, visando automatizar tarefas, inserir multimédia, desenvolver ferramentas de formação, etc.

O Ambiente de Simulação em Arena®

O ambiente de simulação de propósito geral *Arena*® foi desenvolvido, em 1993 para a plataforma Microsoft Windows®, pela empresa Norte Americana Systems Modeling Corporation [TAKU97].

O *Arena*® apresenta uma estrutura hierárquica baseada em *templates*, que englobam painéis, que por sua vez englobam componentes. Os componentes de painéis, pertencentes a um mesmo *template* ou a *templates* distintos, podem ser combinados entre si, permitindo construir uma variedade maior de modelos [KELT02].

Componentes são elementos de software auto-suficientes e reutilizáveis, que podem interagir entre si, seguindo um grupo de regras preestabelecidas [ENGL97]. Estes podem ser utilizados tanto na construção de aplicações, como também na construção de outros componentes. Desta forma, o *Arena*® apresenta um grande número de alternativas verticais (Figura 4.1) para construção de modelos, possibilitando ao utilizador trabalhar com componentes específicos de alto nível, juntamente com a elaboração de rotinas em linguagens de propósito geral. Por um lado, o utilizador tem acesso a todos os recursos de simulação, tais como, suporte à decisão, animação gráfica e lógica de modelação, sem ter que desenvolver uma única linha de código. Um utilizador experiente pode aperfeiçoar a sua produtividade e

compartilhar o seu conhecimento com outros, ao construir componentes reutilizáveis [WAGN00A].

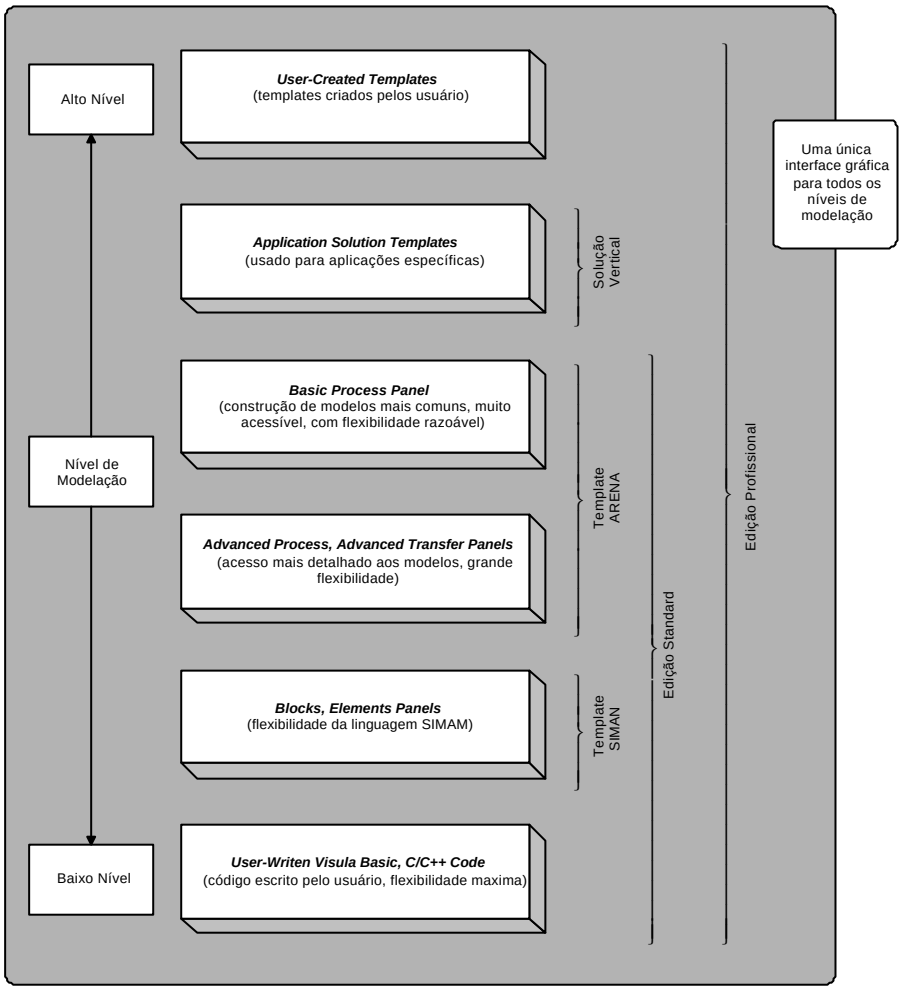


Figura 4.1 – Estrutura hierárquica do *Arena*® [KELT02]

Actualmente, o *Arena*® encontra-se disponível em duas versões: Profissional e Académica, sendo que esta última é distribuída de forma gratuita. As diferença entre as versões são as limitações impostas na versão Académica, que limita a utilização de modelos com o máximo de 150 elementos, além de não permitir a criação de novos *templates*. A versão profissional apresenta dois ambientes de trabalho, descritos a seguir.

O Ambiente *Arena Standard Edition* é desinado para a construção, simulação e análise dos modelos do utilizador. Ele é composto das seguintes ferramentas [KELT02] [WAGN00]:

Simulador: o ambiente em si, com interface padrão Windows®, é destinado à construção, execução e manipulação de modelos. Possui, entre outros, elementos de animação, modelagem e interacção com outras aplicações relevantes.

Analisador de Entradas (*Input Analyzer*): Dado um arquivo do tipo texto, contendo os

intervalos de perturbações ao sistema e/ou suas durações, este programa descreve o seu comportamento através de equações matemáticas. O arquivo, tanto poderá ser gerado através do registo de acontecimentos do dispositivo (*log*), como manualmente através de editor de texto. Outra função da aplicação é a geração de um arquivo do tipo texto baseado em distribuições estatísticas com o intuito de garantir a utilização de valores iguais em simulações diferentes. As distribuições estatísticas utilizadas pelo programa são: Beta, Empirical, Erlang, Exponencial, Gama, Johnson, Lognormal, Normal, Poisson, Triangular, Uniforme e Weibull [KELT02].

Analisador de Saídas (*Output Analyzer*): Responsável pela interpretação dos resultados da simulação que são recolhidos através do componente '*Statistics*' do painel *common*. Este programa permite a geração de tabelas, gráficos de barra ou pontos, histogramas, determinação de intervalos de confiança das amostras apresentadas e exportação dos valores para arquivos de texto padrão, para serem analisados por outras aplicações.

Gestor de Cenários (*Scenario Manager*): tem como objectivo a execução em lote de um modelo, podendo o modelo ser exposto às diferentes condições. Os resultados tanto podem ser analisados através da ferramenta *Output Analyzer*, como através de relatórios gerados pelo próprio programa. Procurando proporcionar modelos que se aproximem do comportamento do sistema que está a ser modelado, o *Arena*® dispõe de mecanismos e artifícios que permitem a integração com outros ambientes (Java, Delphi), conciliando as facilidades disponíveis em simuladores de alto nível, com a flexibilidade proposta pelas linguagens de programação, conforme apresentado a seguir [WAGN00]:

Fonte de dados externa: Permite ao *Arena*® utilizar registos de acontecimentos criados pelo utilizador, ou recolhidos de dispositivos presentes no sistema real, ou até mesmo por outra aplicação. Este recurso permite submeter o modelo a situações reais. Tal integração pode ocorrer utilizando elementos próprios para leitura e escrita em arquivos de textos, como os componentes READ e WRITE localizados no painel *Support* do *template Arena*®, ou estruturas de acesso à base de dados fornecidas pelo ambiente operacional, tipo Data Access Objects (DAO). Esta estrutura permite a utilização e recuperação de dados localizados em bases de dados como SQL Server, Oracle®, Informix, ou em documentos do Microsoft Office.

ActiveX: Esta tecnologia tem como objectivo automatizar acções que deveriam ser realizadas pelo operador. Através desta, o modelo é capaz de responder aos acontecimentos do sistema operacional, tais como abertura e fecho de documentos, início e fim de simulação, entre outros, permitindo inclusive a integração com outras aplicações projectads com este fim [KELT02]. Este recurso utiliza como linguagem nativa de programação o VBA, fazendo com que qualquer objecto (ex. aplicações do Microsoft Office) devidamente registado no sistema possa ser ligado pelo simulador.

Código Externo: Além dos recursos apresentados anteriormente, o *Arena*® permite o

acoplamento de rotinas desenvolvidas em C++ ou Visual Basic®. Esta flexibilidade auxilia o projectista no desenvolvimento de tarefas complexas que não possam (ou não seja interessante) ser representadas utilizando os recursos de alto nível ou da linguagem SIMAN (ex. funções recursivas).

Estas rotinas podem permitir a interactividade com o utilizador. Significa isto que é possível controlar e manusear o modelo, mantendo todos os requisitos e princípios de um simulador.

De notar que o código e ficheiros associados feitos em VBA são independentes da estrutura de dados interna do *Arena*® e que cada modelo de simulação tem o seu próprio projecto VBA, pois cada modelo guarda todos os componentes do projecto VBA no seu ficheiro.

O Ambiente *Arena Professional Edition* é destinado à construção, pelo próprio utilizador, de novos *templates* específicos para as aplicações do utilizador.

A tecnologia diferencial do *Arena*® são os *templates*, ou seja, uma colecção de objectos/ferramentas de modelação, que permitem ao utilizador, descrever o comportamento do processo em análise, através de respostas às perguntas pré-elaboradas, sem programação, de maneira visual e interactiva.

A aplicação *Arena*® possui mais de 60 módulos fornecidos de raiz como parte do sistema geral. Este conjunto de módulos foi projectado de forma a fornecer uma capacidade de modelar sistemas para todos os tipos de aplicações. Além de fornecer características de raiz para recursos, filas de espera, de inspecção, operações lógicas, e relação externa através de ficheiros, o *template* do *Arena*® fornece, também, módulos específicos como por exemplo para a manufactura e manipulação de materiais.

Três painéis compõem o *template* do *Arena*®: o painel comum contém módulos que representam os processos básicos para a simulação tais como chegadas, serviço, e partidas; O painel de suporte contém os módulos suplementares para acções específicas de decisão lógica; e o painel de transferência, cujos módulos são usados para solucionar o fluxo das entidades através do sistema. Os módulos destes painéis podem ser combinados no mesmo modelo ou podem ser usados com outros *templates*.

A animação é incluída automaticamente na maior parte dos *templates* para permitir uma maior rapidez no desenvolvimento dos modelos. Os símbolos/desenhos dos módulos estão incluídos nos *templates*, podendo ser modificados através de ferramentas internas ao *Arena*® (similares a aplicações de desenho gráfico) ou substituídos quer por ícones disponíveis nas bibliotecas do *Arena*® quer por outros de varias aplicações gráficas externas. Por exemplo, pode-se criar um *template* para a avaliação do desempenho de um conjunto de máquinas específicas e anexá-lo ao *Arena*®. Este *template*, por exemplo, poderia ter um painel composto

por elementos (componentes) que abstraem características próprias das máquinas em questão, outro painel com definições das tarefas dos operadores das máquinas, etc.

A Animação dos Modelos de Simulação

A animação na computação está a tornar-se cada vez mais uma importante ferramenta na aplicação de modelos de simulação de sistemas do mundo real. A animação permite que os modelos de simulação adquiram vida gerando uma inspecção visual da funcionalidade do modelo no monitor do computador. Com a animação o operador visa a execução do modelo; as entidades que estão nas filas de espera, a ocupação de recursos, o transporte entre estações, etc. Esta representação dinâmica da execução do modelo fornece valiosas informações sob o desempenho do modelo, que dificilmente são obtidas por uma análise estatística de saída [PEGD90].

A Animação na Avaliação do Modelo

Analisar-se-á a utilidade da animação em cada uma das fases de avaliação do modelo, isto é, na verificação, na validação, nas interacções e na apresentação de resultados do modelo.

-Verificação do Modelo: é um processo de comprovação se o modelo é executado como se pretende. Este processo não prova a infalibilidade do modelo representar o sistema real. Ele apenas detecta que o modelo tem erros.

Na fase de verificação do modelo são detectados muitos tipos diferentes de erros lógicos que foram introduzidos equivocadamente no desenvolvimento do modelo. O isolamento destes erros consomem tempo substanciável e muitas vezes eles passam despercebidos no modelo levando a conclusões erradas sobre o desempenho do sistema. Neste caso, a simulação conduz-nos a uma decisão ineficiente e custosa [LAW91].

Nesta fase de avaliação do modelo, a animação talvez seja a mais efectiva maneira de atacar o problema de verificação da funcionalidade do modelo, pois a animação permite visualizar imediatamente os erros lógicos. Por exemplo, assistindo a uma animação, o analista pode perceber que uma entidade mantém controle sobre um recurso mesmo depois de o ter deixado.

Tais observações directas dos erros na execução do modelo ajudam a compreender melhor o processo e ainda, a diminuir a probabilidade de ocorrência de erros.

-Validação do Modelo: é o processo que reflecte se o modelo responde adequadamente à

imagem do sistema real. A animação evidencia os impactos e as interacções das simplificações feitas na fase de modelação.

Facilita a explicação da funcionalidade do modelo às individualidades intervenientes no sistema real (por exemplo: os operadores do sistema), que conhecem o sistema, por pouco que conheçam de modelação/simulação. Dessa forma o analista, que usualmente não é conhecedor do funcionamento do sistema real pode, com os conhecedores do sistema, reduzir bastante o tempo de explicação de como a modelação está feita e ao mesmo tempo, o operador, que forneceu a informação de como opera o sistema, pode verificar se a sua informação foi modelada e interpretada correctamente.

-Interacções Dinâmicas: a animação contribui para o conhecimento da funcionalidade do sistema e permite verificar quais são as interacções existentes entre os componentes desse sistema. Considerando que os métodos estatísticos têm limitações na estimação e comparação do desempenho, a animação pode fornecer informações sobre os processos que estão a interagir e qual o seu desempenho. Assim a animação poderá propor melhoramentos, que serão testados no modelo antes de o implementar no sistema real.

Estas melhoras frequentemente não são percebidas com uma análise estatística ou gráfica das variáveis. Elas tornam-se óbvias visualizando-se através da animação.

-Apresentação dos Resultados do Modelo: ultimamente o sucesso de um projecto de simulação tem sido medido pelo impacto no desenho e operação do sistema real. Os gestores, que são os agentes de decisão sobre o sistema real devem conhecer e usar os resultados do modelo para que o estado de simulação alcance os efeitos desejados.

Desse modo, uma parte importante de qualquer projecto de simulação é a apresentação do modelo e seus resultados para os agentes que tomarão as decisões.

Como foi observado por Poorte [POOR89]: "O objectivo básico de qualquer esforço de modelação é prover informação à gestão, que tomará a decisão. Para isto é fundamental que a informação seja confiável". O gestor deverá escolher entre aceitar ou rejeitar as informações fornecidas pelos modeladores. A animação facilita quer a comunicação, que é de vital importância para a credibilidade do modelo, quer a negociação da proposta de solução.

A Animação do Layout

O *layout* é definido como uma disposição gráfica de objectos que representam o sistema que está a ser representado. Os objectos dentro do *layout* podem ser estáticos ou dinâmicos.

Os objectos estáticos do *layout* não mudam durante a execução da simulação. Eles geralmente são utilizados para representar a estrutura física e o meio ambiente em que a simulação está a ser realizada. Por exemplo, a animação de um objecto estático pode consistir

em muros ou paredes, estradas, separação entre os locais de trabalho, imobiliário, e dependências gerais do local que está ser modelado. Então, os objectos estáticos representam tudo o que auxilia para representar o cenário no monitor do computador mas que não necessita mudar de estado durante a simulação.

Por outro lado os objectos dinâmicos do *layout* estão ligados aos objectos estáticos de fundo mas que constantemente mudam de forma, cor e posição durante a simulação. Então, as peças, os trabalhadores, as máquinas, e os meios de transporte são apresentados como objectos dinâmicos do *layout.* Os objectos dinâmicos também incluem as mudanças digitais e representações gráficas dos valores das variáveis do sistema e as estatísticas registadas.

Usando VBA no Arena®

A interacção do simulador *Arena*® com folhas de cálculo e outros programas que eram convertidos de forma a serem compatíveis com SIMAN, embora pouco viáveis, foram operações por vezes usadas para importar/exportar informação para/da simulação.

Este tipo de funcionalidade alcançou uma evolução excepcional quando o *Arena*® introduziu o Visual Basic® para aplicações (VBA) da Microsoft®.

O VBA pode ser usado quando o projecto modelo está a ser lido, executado, terminado ou enquanto as entidades flúem através dos módulos do modelo do *Arena*®.

Trabalhar em VBA é realmente simples uma vez que o editor verifica a sintaxe de cada linha de código enquanto se está a escrever, eliminando assim erros indesejáveis na programação. Finalmente, usando o editor interactivamente, pode-se eliminar e corrigir outro tipo de erros, sem ter que reiniciar a execução do modelo. Através desta ferramenta interactiva pode-se fazer paragens na simulação, em qualquer linha de código, e acompanhar a evolução de valores das variáveis, das filas de espera, enfim de todas as alterações que estão a acontecer no modelo, inclusivé modificar as especificações de qualquer recurso, mesmo a meio da simulação.

Deve ter-se em conta que o código VBA é independente do *Arena*®. Esta separação de estruturas de dados tem vantagens pois permite-nos criar variáveis que não existem no *Arena*®.

A comunicação entre o VBA e o *Arena*® é feita através de acontecimentos segundo a Figura 4.2.

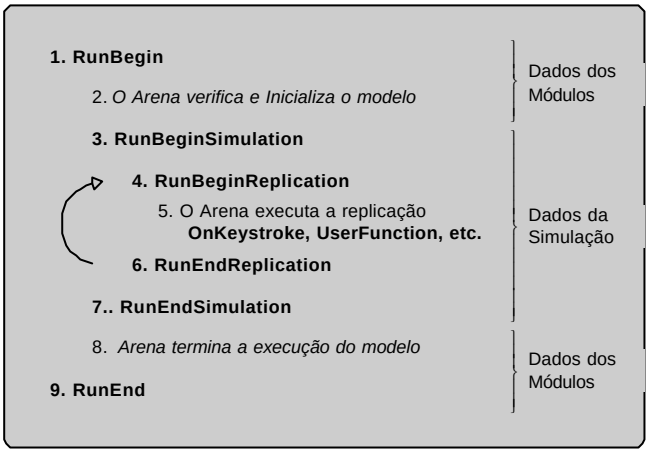


Figura 4.2 – Acontecimentos do *Arena*® [KELT02]

A Figura 4.3 descreve a altura em que se pode aceder à informação dos objectos SIMAN, do modelo, durante a simulação.

RunBegin	ocorre antes do inicio da simulação
RunEnd	ocorre depois da simulação ter terminado
RunBeginSimulation	precede a primeira replicação
RunBeginReplication	executado sempre que começa uma nova replicação
RunEndReplication	executado sempre que termina uma replicação
RunEndSimulation	ocorre depois da simulação ter terminado
RunStep	antes de cada run step
RunPause	ocorre quando se faz uma pausa na simulação
RunResume	quando a simulação recomeça
RunBreak	ocorre quando se força a paragem da simulação
OnKeyStroke	quando uma tecla é pressionada durante a execução da simulação
OnClearStatistics	quando as estatísticas são apagadas
RealTimeSend	quando é enviada uma mensagem com o tempo real

Figura 4.3 – Descrição dos Acontecimentos do *Arena*® [KELT02]

4.2. A Constituição do Sistema Proposto

Aparentemente a constituição do sistema proposto pode parecer um pouco complexa, no entanto, a sua utilização é bastante acessível. Um dos pressupostos na elaboração deste sistema foi facilitar o seu uso por parte de utilizadores sem conhecimentos profundos de informática ou de simulação.

O sistema é constituído por três elementos: a ferramenta informática, o *Arena*® e uma base de dados.

A ferramenta informática aqui denominada GAMSTAF (Gerador Automático de Modelos de Simulação para a Têxtil António Falcão) permite a definição das características

dos elementos que vão compor o nosso sistema produtivo. São também seleccionados os critérios e estratégias que serão aplicados no sistema.

O GAMSTAF faz a interligação entre o utilizador e o simulador, apoiando-o na parametrização de todos os tipos de definições e características necessárias ao modelo de simulação, na escolha das estratégias a aplicar, na criação de novos cenários de teste e na obtenção e análise de resultados. Toda a parametrização definida pelo utilizador fica registada na base de dados.

Antes do funcionamento propriamente dito, a ferramenta tem uma tarefa muito importante: a exportação para o modelo no *Arena*® da informação definida pelo utilizador.

Relativamente ao *Arena*® existem dois elementos fundamentais, o *Modelo Base* e o *template*.

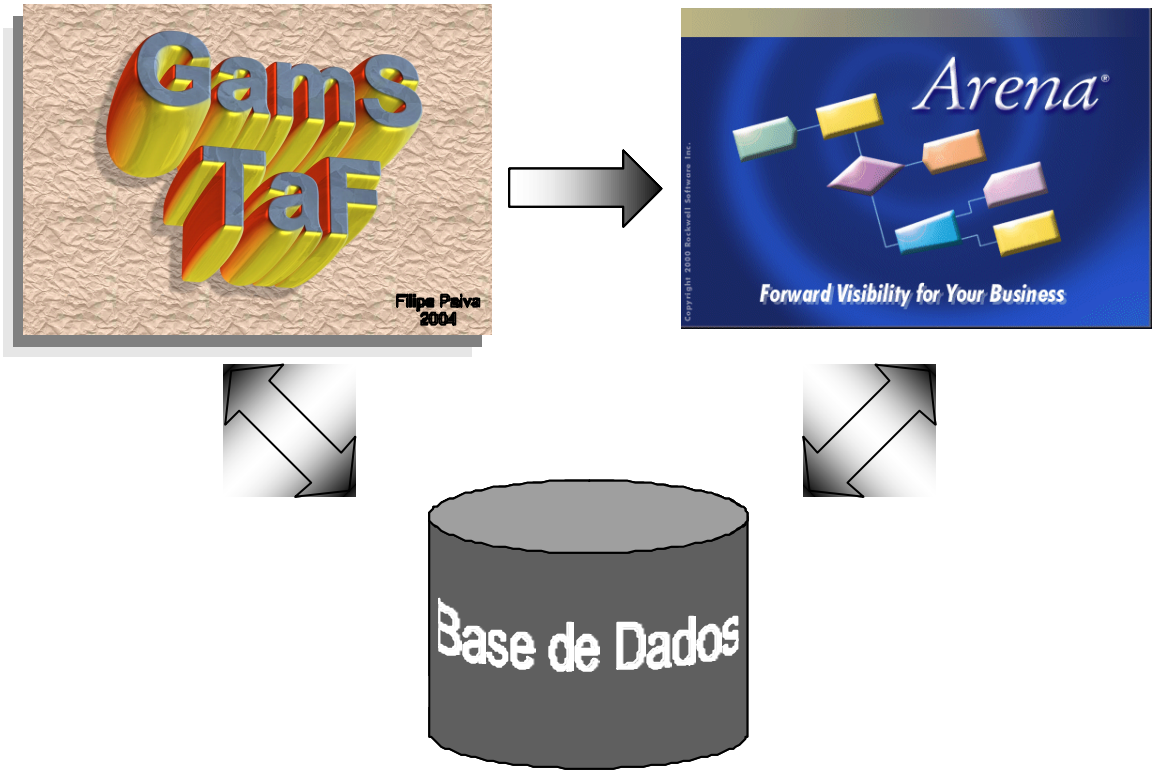


Figura 4.4 – Interligação entre os componentes existentes

O *Modelo Base* surge devido à impossibilidade de passar texto (no nosso caso seria código de programação) para um documento compatível com o simulador.

A função deste *Modelo Base*, além de garantir a implementação das estratégias seleccionadas é controlar as máquinas, os operadores de máquina e a equipa de manutenção.

A informação é exportada para este *Modelo Base*, através da GAMSTAF, auxiliada por um *template* específico. Este *template* faz a representação do sistema real, contendo as

principais características e especificidades do sistema real. No *template* tem-se definidos os módulos que representam o funcionamento da máquina e do operador de máquina.

O *template* foi elaborado em linguagem de simulação *Arena*®, o código contido no *Modelo Base* foi elaborado em VBA e a ferramenta informática foi construída em linguagem de programação Visual Basic®. Como já foi explicado no ponto 4.1 deste relatório, o simulador permite uma interligação directa com o GAMSTAF.

A base de dados contém a informação gerada pelo utilizador no GAMSTAF que é necessário transferir para o *Arena*®, tais como o número de linhas, o número de máquinas de cada linha, o número de produtos e suas características. A informação gerada pelo simulador, durante a simulação, necessária para posterior análise e estudos de resultados através do GAMSTAF, também é registada na base de dados.

A relação entre os elementos que compõem o sistema está descrita na Figura 4.4.

4.3. O modelo em *Arena*®

Os elementos necessários à geração do modelo em *Arena*®, com as definições do sistema e com a aplicação das estratégias, implica a utilização do *template* e do *Modelo base.*

O *template* é constituído por dois módulos específicos, o módulo *Máquina* e o módulo *Entrada Operador*. O *template* é acoplado ao *Arena*®.

A elaboração de um módulo não é mais do que o agrupamento de vários módulos disponíveis no *Arena*®. Cada módulo existente no simulador é passível de ser programado com determinados objectivos específicos. O conjunto destes módulos devidamente programados resulta num módulo novo, com uma função específica. O *template* é elaborado em linguagem de simulação *Arena*® de forma a reflectir o comportamento das máquinas e do operador de máquina.

O *Modelo Base* apenas contém código de programação em Visual Basic®. O objectivo deste código é controlar os recursos e entidades do sistema durante a simulação, garantindo a aplicação das estratégias seleccionadas pelo utilizador.

De seguida serão descritos os aspectos mais relevantes dos elementos construídos no *Arena*®: o *Modelo base* e o *template* constituído pelos dois módulos.

O Modelo Base

No que se refere à construção do modelo no simulador *Arena*®, resta referir como é feito o controlo dos modelos desenvolvidos nos pontos 3.1 e 3.2.

Estes critérios são garantidos através de um programa escrito em VBA, dentro do

simulador. Com a actual versão do *Arena*®, não é possível criar um modelo novo e copiar para dentro dele um texto de código em VBA. Esta situação obriga-nos a transportar esse código, sempre que se gera um novo cenário de simulação.

Foi então criado um modelo que se denomina por *Modelo Base*, que apenas contém código em VBA. Quando se faz a exportação de um novo cenário para o *Modelo Base*, este já contém todo o código que controlará o modelo, como terá também as respectivas características, seleccionadas pelo utilizador, para o novo ambiente que se quer simular. O processo de exportação fará a transformação referida na Figura 4.5.

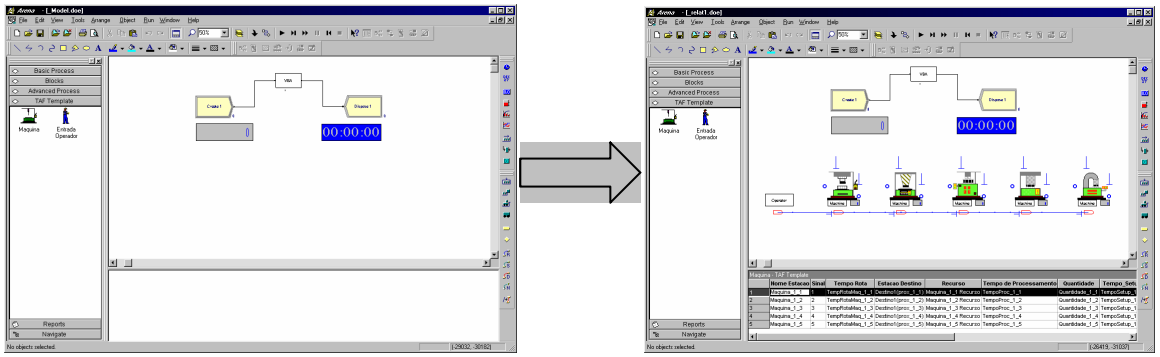


Figura 4.5 – Transformação do *Modelo Base* no cenário de simulação

Nesta exportação pode-se observar um cenário de teste com apenas uma linha, e respectivo operador contendo cinco máquinas, todas com imagens diferentes, com características distintas.

O processo de exportação é uma opção incluída no GAMSTAF e permite facilitar ao utilizador a tarefa de gerar automaticamente novos modelos de simulação com diferentes quantidades de linhas, máquinas e operadores. Também, permite a alteração das características ou propriedades associadas a cada um destes elementos.

A ferramenta GAMSTAF tem outras funções para além da exportação de modelos de simulação. No ponto seguinte serão explicadas as funções do GAMSTAF e a sua estrutura de funcionamento. O procedimento e biblioteca de exportação de modelos do *Arena*®, usada no GAMSTAF é baseada em [VIEI02].

O Módulo *Máquina*

O módulo *Máquina* representa o comportamento de uma máquina de produção de meias. Este módulo tem quatro estados possíveis (Trabalhar, Parada, Avariada, Manutenção).

A Figura 4.6 mostra o aspecto final da construção lógica do módulo *Máquina*. Para se entender melhor a Figura 4.6 passa-se a descrever todos os módulos e suas principais características:

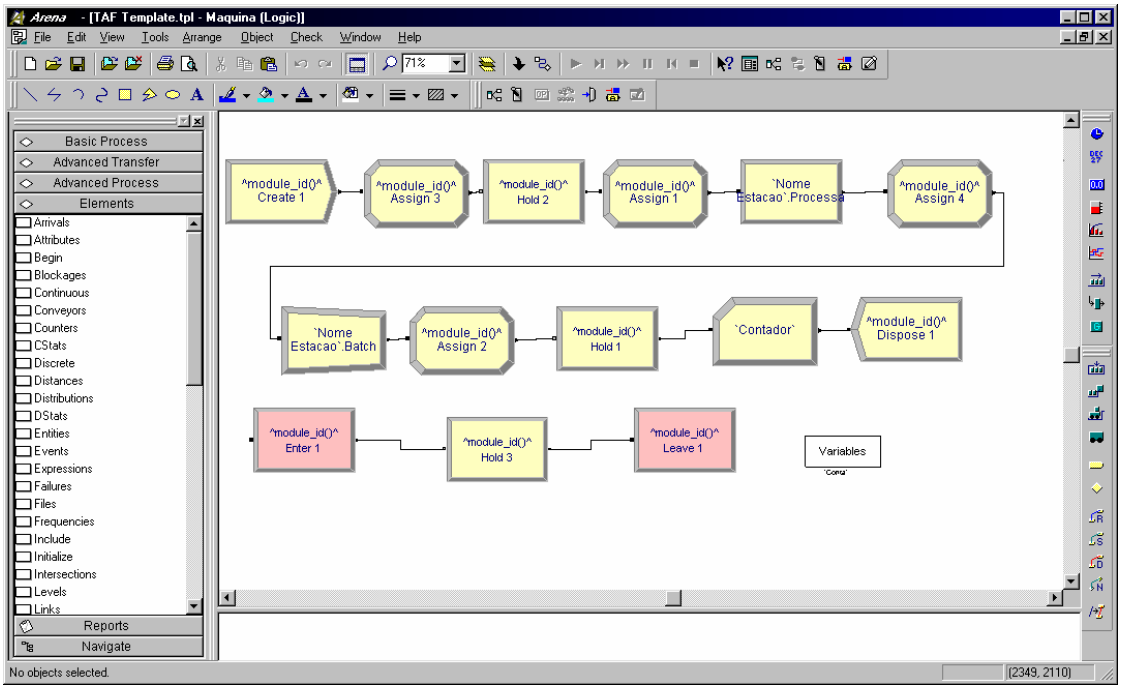


Figura 4.6 – Módulo *Máquina*

Módulo Create 1: A função deste módulo é criar as entidades fio.

Módulo Assign 3: A missão deste módulo é permitir alterar a imagem da entidade fio.

Módulo Hold 2: O módulo *Hold* pode ser aplicado para diversas finalidades. Neste caso o objectivo é controlar a entrada de fio na máquina. A entrada de novas quantidades de fio está sujeita a várias condições: a quantidade de pares a produzir, se o lote produzido na totalidade não foi retirado pelo operador e a máquina não estiver no estado de *Parada*. Se estas condições não se verificarem o fio ficará retido neste módulo.

Módulo Assign 1: Este módulo funciona como contador do número de pares de meias (produtos) que se encontram em produção, ou terminadas para formar o lote.

Módulo Process: Representa a área de produção da máquina. Neste módulo define-se as características do recurso máquina, a capacidade e o tempo de processamento.

Módulo Assign 4: Altera a imagem do produto. Pretende representar a alteração do fio (matéria-prima) em par de meias (produto final).

Módulo Batch: Garante o acumular de produtos até que se atinja a quantidade do lote. Este módulo retém as entidades até essa condição ser verdadeira, transformando as entidades (pares de meias) numa única (lote).

Módulo Assign 2: Modifica a imagem do lote. Pretende representar visualmente a produção de um lote. Tem ainda a função de colocar a zero a variável que controla a quantidade produzida pela máquina.

Módulo Hold 1: Neste instante o lote está completo, e a máquina parada aguardando que o operador o retire da máquina para que esta possa prosseguir com a produção. O lote aguardará neste módulo até que o operador disponibilize a máquina, retirando o lote.

Módulo Contador: É um módulo do tipo *Record* e, tem por objectivo contar os lotes produzidos pela máquina.

Módulo Dispose: Este módulo retira o lote da máquina e do ambiente de simulação. Quando o lote é contabilizado e abandona a máquina deixa de ter qualquer serventia.

Módulo Hold 3: Permite às entidades que percorrem este módulo enviar um sinal para o modelo. O envio de sinal é o método usado para notificar que um lote foi retirado de uma determinada máquina.

Módulo Dispose 1: Faz a ligação física entre este módulo e outras zonas do modelo.

Módulo Variables: Permite a definição da variável conta, que tem o objectivo de contar o número de lotes produzidos pelas máquinas.

Além destes módulos, que estão visíveis na Figura 4.6, ainda existe um outro, o **Módulo Failure**, para definir a probabilidade de ocorrência do estado de avaria na máquina. Numa variável define-se a quantidade que a máquina produz sem avaria, numa outra define-se quando podem ocorrer as avarias. As variáveis podem ser usadas em simultâneo e permitem a inclusão de funções estatísticas.

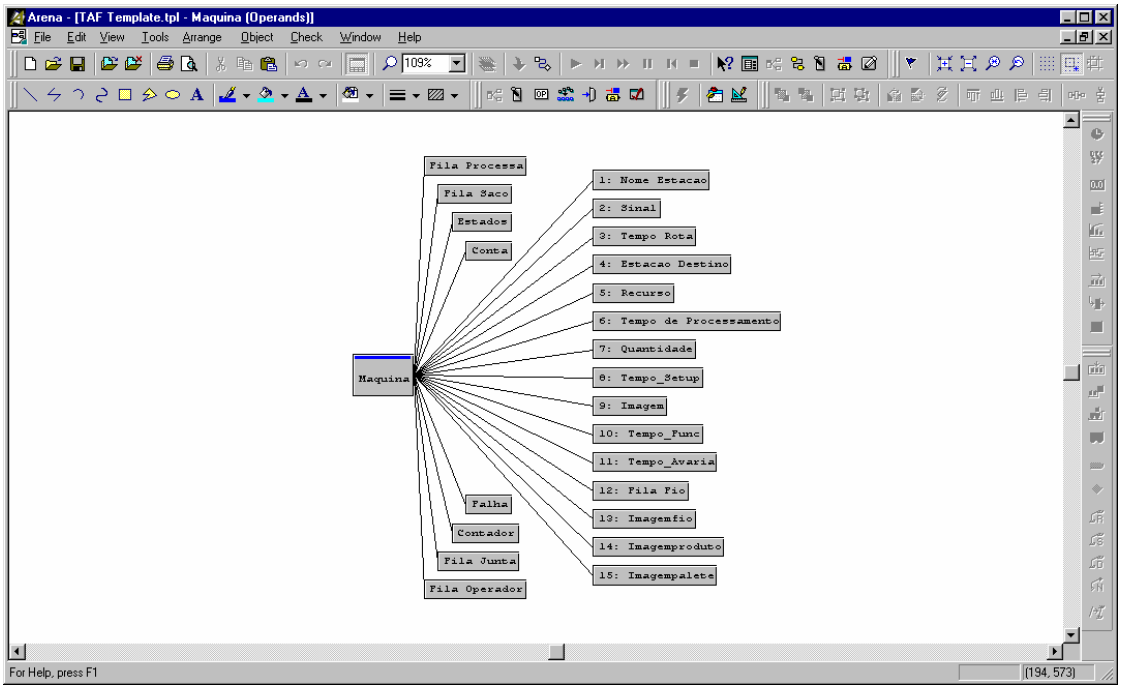


Figura 4.7 – Parâmetros do Módulo *Máquina*

Para possibilitar um controlo mais apertado sobre a actividade da máquina foram acopladas a este recurso alguns parâmetros (ver Figura 4.7).

A Figura 4.7 mostra os parâmetros que além de serem necessárias ao modelo, podem ser parametrizáveis pelo utilizador.

Algumas variáveis são de utilização exclusiva do programa (criado em VBA) existente dentro do simulador, outras são definidas pelo utilizador.

Os valores das variáveis definidas pelo utilizador aparecem na GAMSTAF e são transferidas para o modelo quando este é exportado, criando um novo cenário de simulação.

O Módulo *Entrada Operador*

O módulo *Entrada Operador* foi elaborado para representar o comportamento de um operador de máquina de produção de meias.

No nosso projecto vai-se necessitar de um operador por cada linha que se pretenda e o operador só trabalhar nas máquinas que pertencem à mesma linha a que ele está associado.

A Figura 4.8 mostra os componentes lógicos do *Arena*® usados para implementar a funcionalidade do operador de máquina. Para melhor compreensão, serão descritos os módulos com uma breve descrição da sua função.

Módulo Create: A finalidade deste módulo é criar as entidades do tipo operador de máquina.

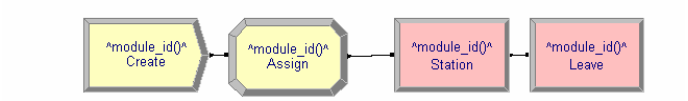


Figura 4.8 – Módulo *Entrada Operador*

Módulo Assign: Estabelece os atributos referentes às entidades operador de máquina. Estes atributos são as diferentes características que a entidade pode assumir, como por exemplo a imagem que irá ter durante a simulação.

Módulo Station: Este módulo tem a função de estabelecer o conjunto de máquinas que estão atribuídas à entidade operador de máquina.

Módulo Leave: Faz a ligação física entre este módulo e outras zonas do modelo.

A deslocação do operador da máquina actual para a máquina seguinte é totalmente controlada pelo utilizador, podendo este escolher um critério de vários existentes, criados no GAMSTAF com esse propósito. O controlo e aplicação dos critérios de deslocação são

garantidos pelo código existente no *Modelo Base*

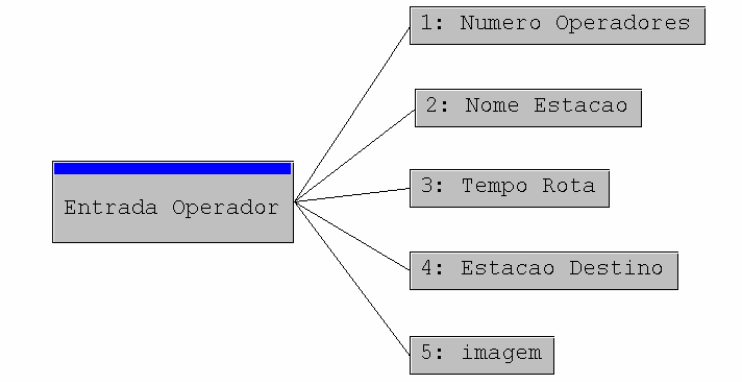


Figura 4.9 – Parâmetros do Módulo *Entrada Operador*

As principais funções dos parâmetros do módulo *Entrada Operador* (Figura 4.9) são:

Numero Operadores: Estabelece o número de operadores disponíveis para a linha de produção.

Nome Estacao: permite distinguir as diferentes *Entrada Operador* existentes no modelo. A designação dos módulos será por defeito *Entrada Operador #*, sendo # o número da linha.

Tempo Rota: Estabelece o tempo em segundos que o operador demora para executar a tarefa de retirar o saco das máquinas e colocar a máquina operacional.

Estacao Destino: Define a máquina seguinte para esta entidade, isto é, a próxima máquina a ser visitada pelo operador.

Imagem: Aplica a imagem previamente escolhida para o operador.

4.4. A Ferramenta GAMSTAF

A construção de uma ferramenta que fosse de fácil manuseamento por parte de utilizadores com conhecimentos superficiais de informática foi um desafio colocado à partida. Após diversas tentativas de simplificação decidiu-se organizar a ferramenta por funcionalidades e/ou áreas de actividade. Utiliza-se um componente existente no compilador de Visual Basic®, que nos permitiu adaptar as diversas funcionalidades da ferramenta por separadores.

Para melhor se compreender a estrutura desta ferramenta vai-se dividi-la em três grupos:

1. Parametrização pré-exportação – introdução da informação necessária para criar o cenário de simulação

2. Monitorização da simulação – seguimento, em tempo real, da simulação
3. Análise de resultados – estudos, à posteriori, dos resultados obtidos durante a simulação

O primeiro grupo permite escolher o *Modelo Base* e inserir todas as parametrizações necessárias com a correcta definição do modelo que se pretende simular. Deste grupo fazem parte os primeiros seis separadores horizontais (desde a Figura 4.10 até à Figura 4.18).

O segundo grupo, quando a simulação está em funcionamento, permite observar a alteração das variáveis, expressões, entidades, filas de espera e recursos, como resultado do decorrer da simulação. Mais ainda, pode-se observar o fluxo da matéria-prima e as transacções dos armazéns, bem como acompanhar a produção das encomendas. Pode-se incluir neste grupo desde o sétimo ao oitavo separador (da Figura 4.19 até à Figura 4.20).

O terceiro grupo faz a análise e apresentação dos resultados pós simulação. Nos separadores nove e dez pode-se determinar quanto e quando é necessária a matéria-prima, determinado através de um dos métodos de aquisição de materiais disponíveis (Figura 4.21). A análise e apresentação dos principais índices de desempenho do modelo estão disponíveis no último separador (Figura 4.22)

De seguida, apresentam-se os separadores que constituem a ferramenta GAMSTAF, em que, além de um breve resumo contendo o seu principal objectivo, serão mencionadas, quando considerado relevante, outras funções incluídas no separador em questão.

1. Load Model

Funcionalidade:

Permitir ao utilizador seleccionar o modelo sobre o qual pretende criar um novo cenário para simular no *Arena*®.

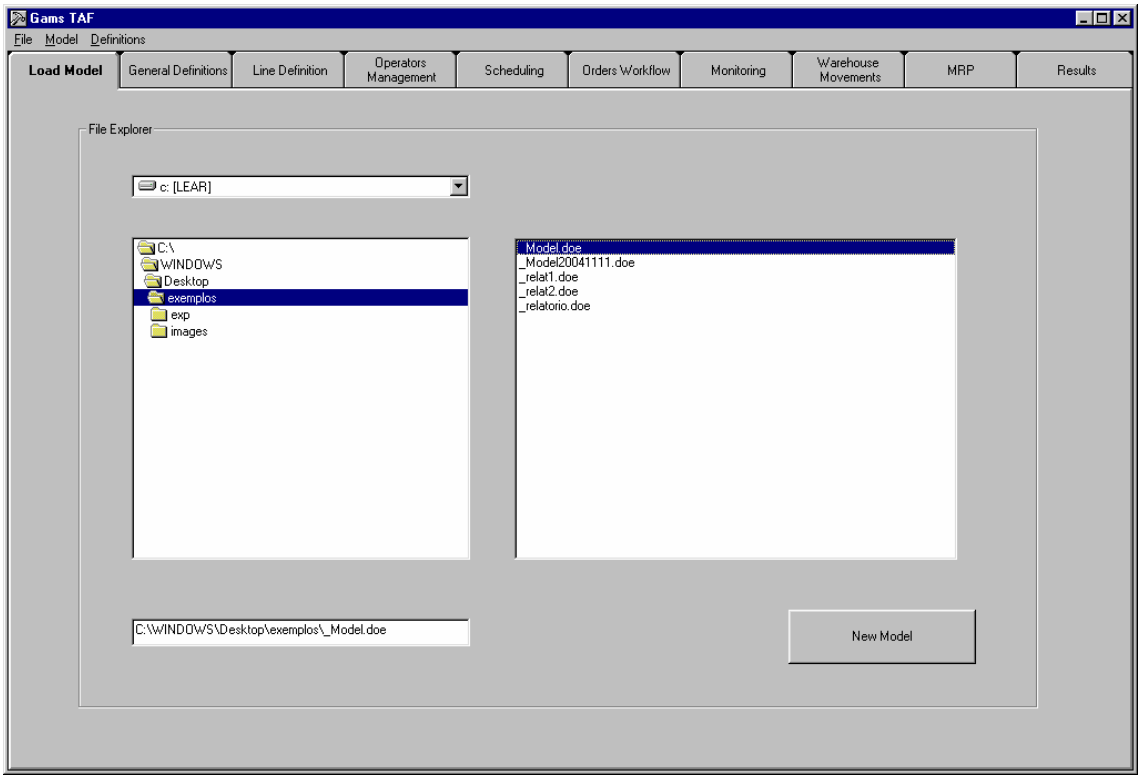


Figura 4.10 – Separador do GAMSTAF: *Load Model*

Esta opção pressupõe que o utilizador pretende sobrepor novas escolhas e definições num *Modelo Base.*

Observando para esta Figura, facilmente nos se apercebe que estão disponibilizados os elementos essenciais para se poder seleccionar os *Modelos Base* existentes no computador, permitindo também uma organização personalizada dos modelos criados por parte dos utilizadores.

2. General Definitions

Para agrupar todas as parametrizações, nomeadamente, dos produtos, dos operadores e das máquinas, inclui-se neste separador, um outro com quatro separadores verticais.

2.1 Product Structures

Funcionalidade:

Permitir definir, para os produtos, a estrutura da sua composição para a produção de uma unidade.

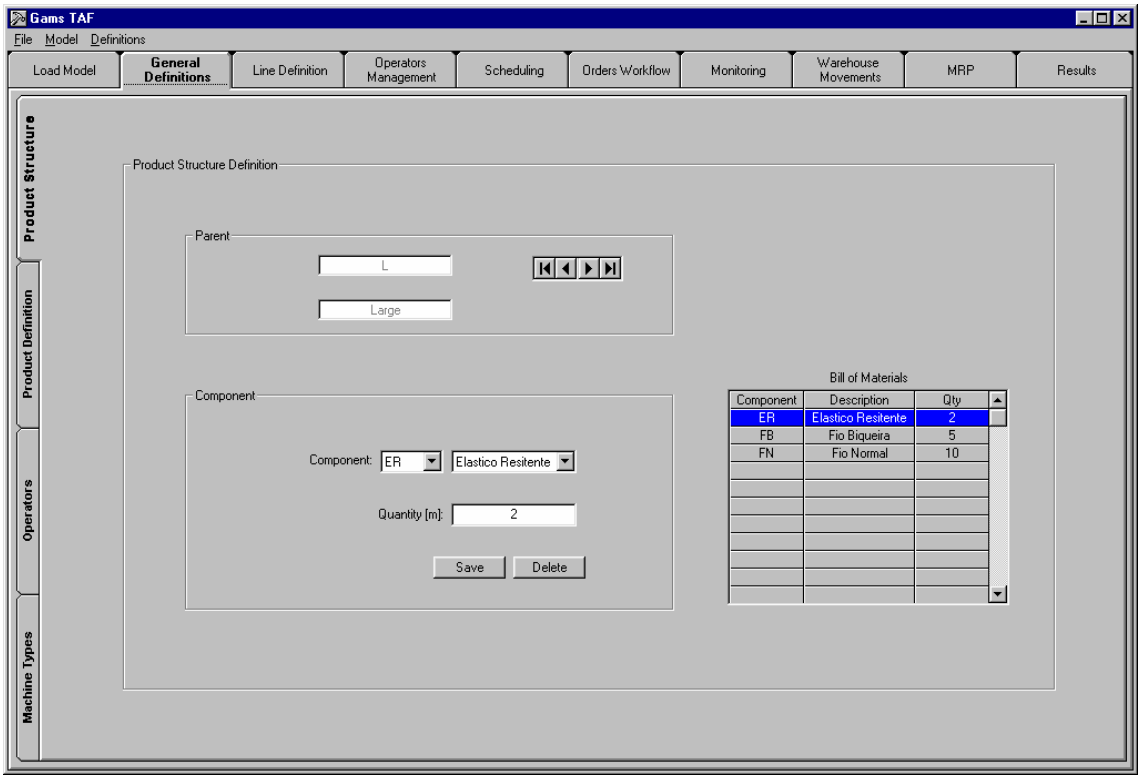


Figura 4.11 – Separador do GAMSTAF: *Product Structures*

Mais conhecido por BOM (*Bill of Materials*) ou Lista de Materiais, representa as matérias-primas necessárias para a execução de um determinado produto.

No exemplo tem-se a definição da estrutura do produto *Large* que está composto por 2 unidades de *Elástico Resistente*, 5 unidades de *Fio Biqueira* e 10 unidades de *Fio Normal.*

Esta informação será utilizada, durante o funcionamento da simulação. No início da execução de um produto, o seu BOM será retirado do armazém para a máquina que o irá consumir, originando um produto do tipo *Large.*

2.2 Product Definition

Funcionalidade:

Definição das características específicas dos produtos.

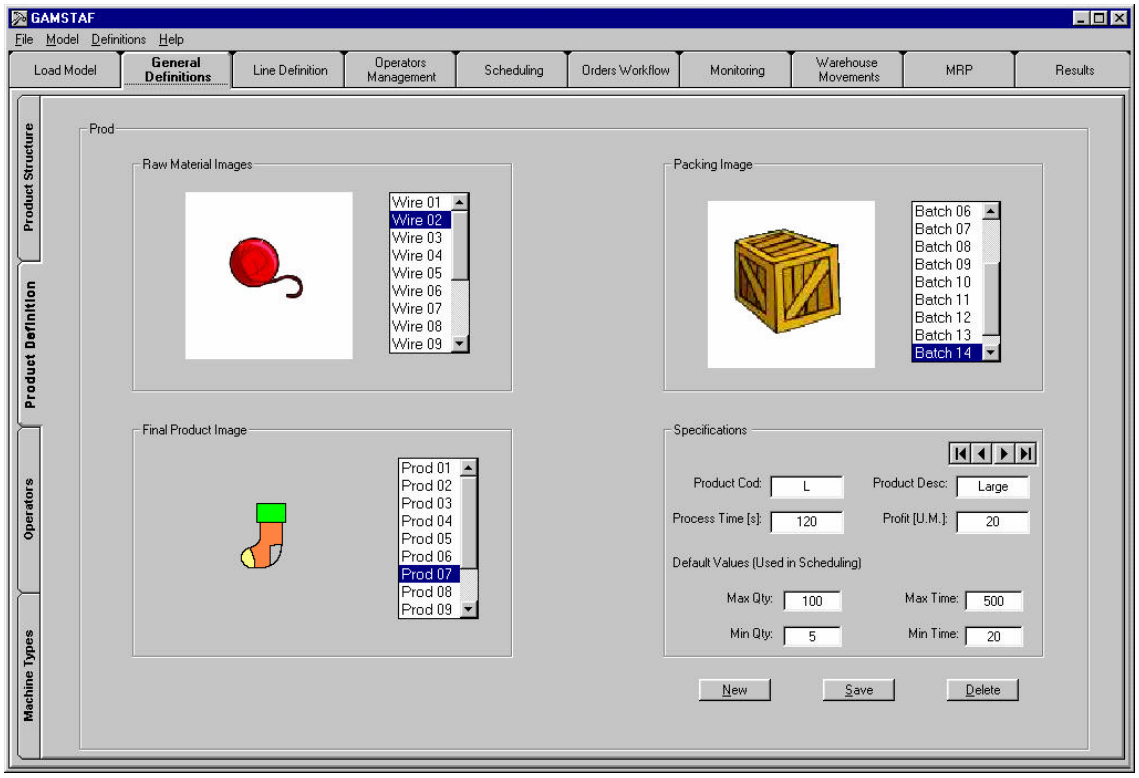


Figura 4.12 – Separador do GAMSTAF: *Product Definition*

A possibilidade de escolher imagens diferentes para produtos diferentes implicará uma distinção imediata dos produtos que estão a ser produzidos nas máquinas. Este efeito só é visível quando a simulação está em funcionamento.

Outras características tidas em conta foram o tempo de processamento e o lucro do produto. O tempo de processamento é a duração, em segundos, que uma unidade de produto necessita para ser produzida por uma máquina. O lucro é expresso em unidades monetárias.

Pode-se definir por defeito, os valores dos limites de quantidades e de tempo para produzir cada produto. Estes valores são usados pelo módulo de escalonamento que será referido no ponto 5 (Figura 4.17).

Todas as imagens incluídas nesta ferramenta foram previamente adicionadas a uma biblioteca compatível com o simulador. Porém, outras imagens poderão ser usadas desde que adicionadas a essa biblioteca.

2.3 Operators

Funcionalidade:

Definição das características dos operadores das máquinas.

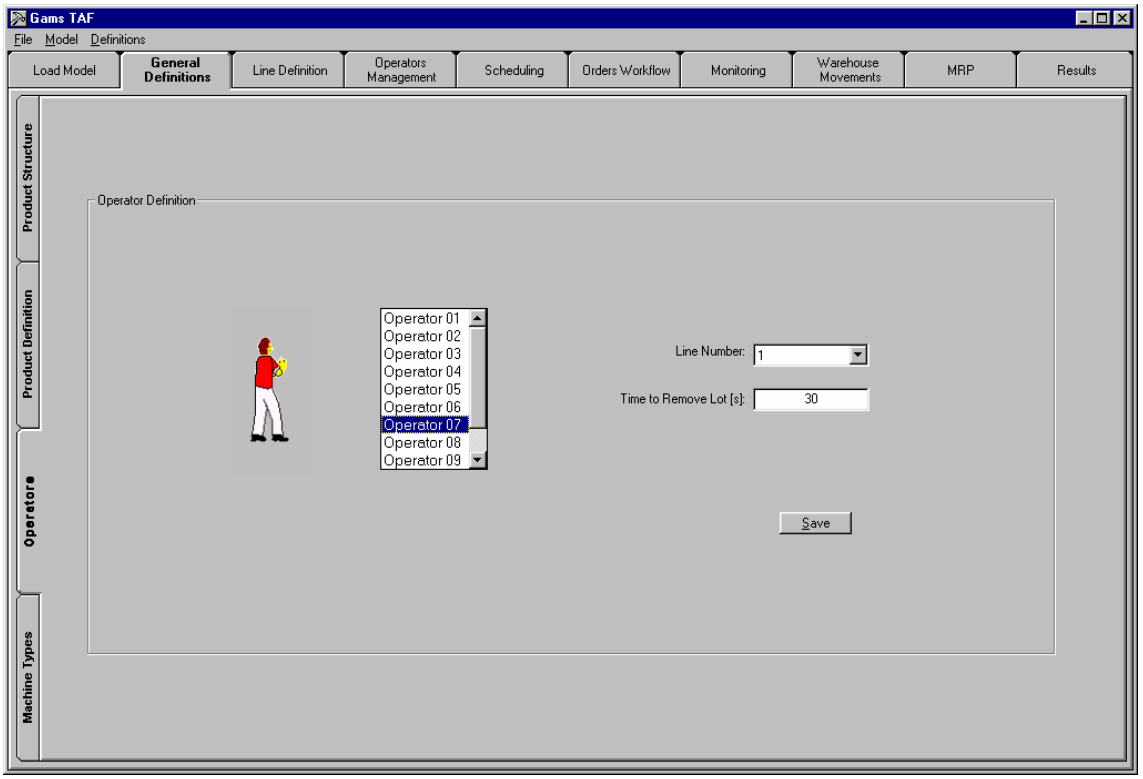


Figura 4.13 – Separador do GAMSTAF: *Operators*

O objectivo deste formulário é dar a possibilidade ao utilizador de distinguir os diversos operadores.

Para além de atribuir diferentes imagens aos operadores de máquina, é possível personalizar o seu desempenho. O tempo que cada operador demora a executar a tarefa nas máquinas é parametrizável e definido em segundos.

A escolha do operador é feita por linha, em que cada operador faz o controlo total de uma única linha de produção, independentemente do número e tipo de máquinas que pertençam à linha.

2.4 Machine Types

Funcionalidade:

Definição completa das especificações relativas às máquinas.

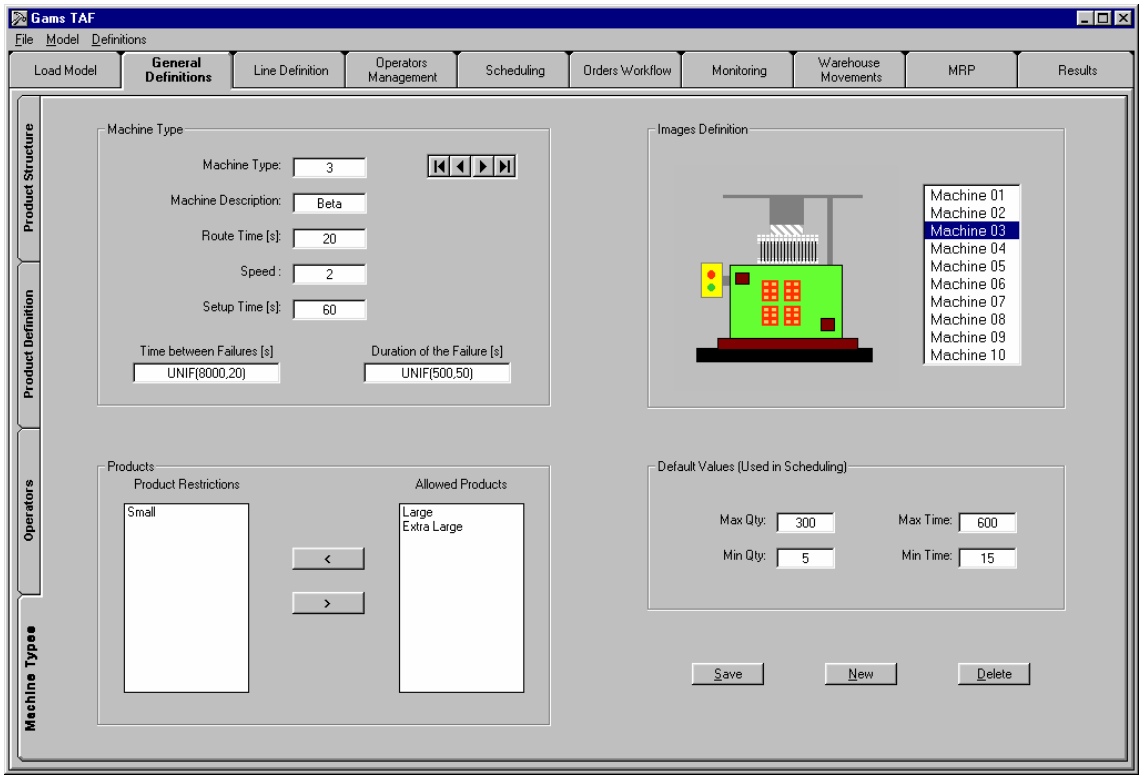


Figura 4.14 – Separador do GAMSTAF: *Machine Types*

Para cada tipo de máquina, além da imagem, pode-se definir: as restrições/permissões aos produtos, a velocidade da máquina, o tempo entre falhas, a duração dessa avaria e os valores por defeito dos limites de quantidades e de tempo que devem produzir. Estes valores são usados pelo módulo de escalonamento que será referido no ponto 5 (Figura 4.17).

O factor velocidade (*Speed*) afecta o tempo de processamento de um produto. Neste caso com uma velocidade 2, por exemplo, para uma unidade do produto *Large* com 120 segundos de tempo processamento é processado por esta máquina em 60 segundos.

O *Setup* da máquina é activado quando a máquina altera o tipo de produto, por exemplo, quando passa do produto *Large* para o *Small*. Neste caso, a máquina começa a trabalhar decorrido o tempo de *Setup* mais o tempo da tarefa do operador.

Route Time, expresso em segundos, pretende reflectir o tempo que o operador demora a atravessar a máquina. Por exemplo, no início, se o operador se deslocar para a primeira máquina e essa máquina for do tipo 3, então demorará 20 segundos a lá chegar.

3. Line Definition

Funcionalidade:

Imputar a cada linha de produção a quantidade e o tipo de máquinas que se vai desejar no modelo a simular.

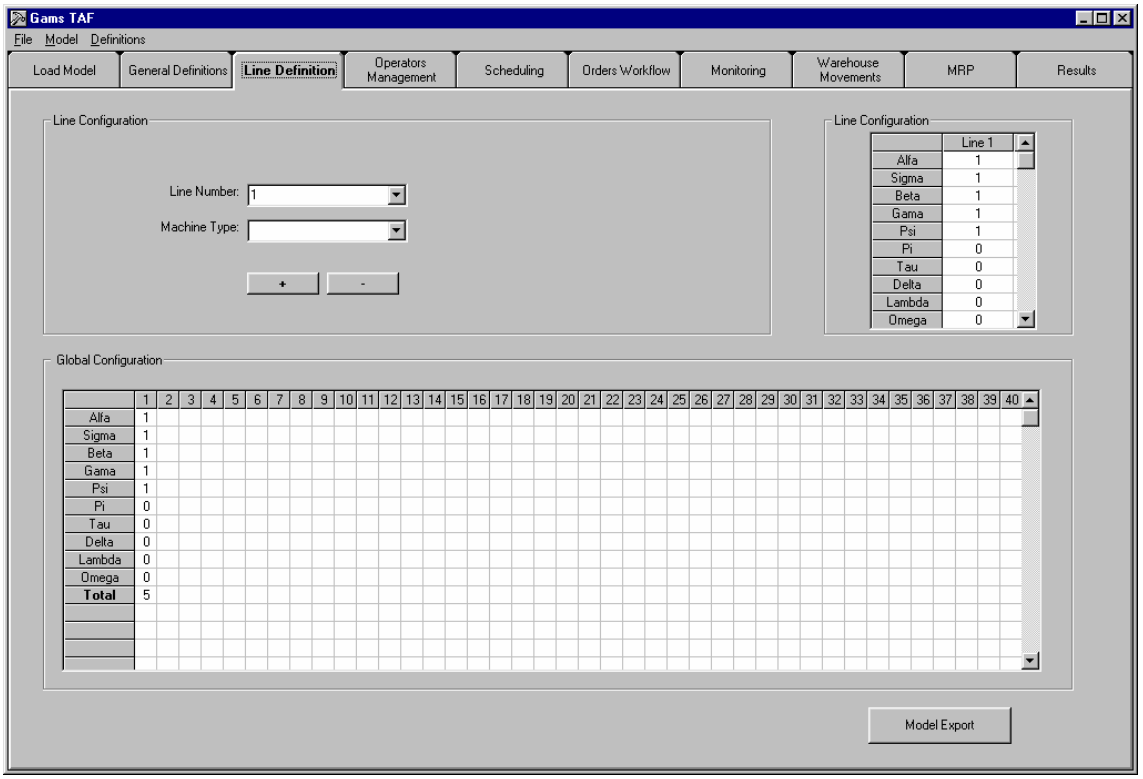


Figura 4.15 – Separador do GAMSTAF*: Line Definition*

Este formulário auxilia o utilizador a elaborar a composição do sistema produtivo que pretende simular.

Pode-se seleccionar a quantidade de linhas e respectivo número. A cada linha pode-se associar a quantidade do tipo de máquinas que se desejar.

As máquinas aparecerão no modelo de simulação pela ordem que foram seleccionadas neste formulário. Assim, se se escolher primeiro uma máquina tipo 4 e depois uma máquina tipo 2 para uma determinada linha, no modelo de simulação aparecerá primeiro a máquina do tipo 4 e depois a do tipo 2.

4. Operators Management

Funcionalidade:

Permitir escolher as estratégias para os operadores das máquinas e para a equipa de manutenção.

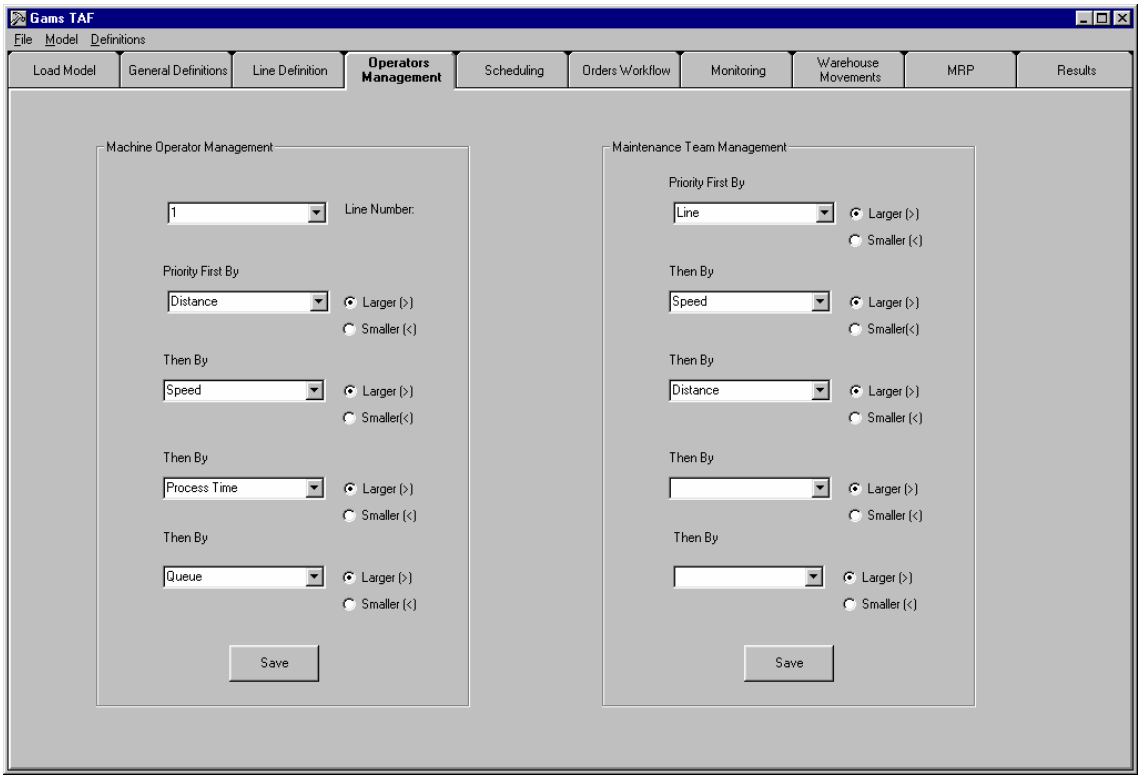


Figura 4.16 – Separador do GAMSTAF: *Operators Management*

Define os critérios aplicados nos procedimentos descritos nas Figuras 3.5 e 3.3 respectivamente da esquerda para a direita.

Pode-se definir, no caso dos operadores de máquinas, quatro níveis de critério para determinar qual será a próxima máquina que o operador irá operar. Estes critérios só são usados se no momento em que o operador fica livre existir mais do que uma máquina no estado *Parada*.

Para cada critério acrescentou-se a possibilidade de ordenação crescente ou decrescente. Por exemplo, se sedefinir como primeiro critério a distancia (*Distance*) pode-se escolher se se quer a que está mais perto ou mais longe, relativamente à posição actual do operador.

Relativamente à equipa de manutenção, acrescentou-se, comparativamente ao operador, o critério *linha.* O método de aplicação é igual ao do operador de máquina.

5. Scheduling

Funcionalidade:

Calcular e distribuir o escalonamento da produção pelas máquinas.

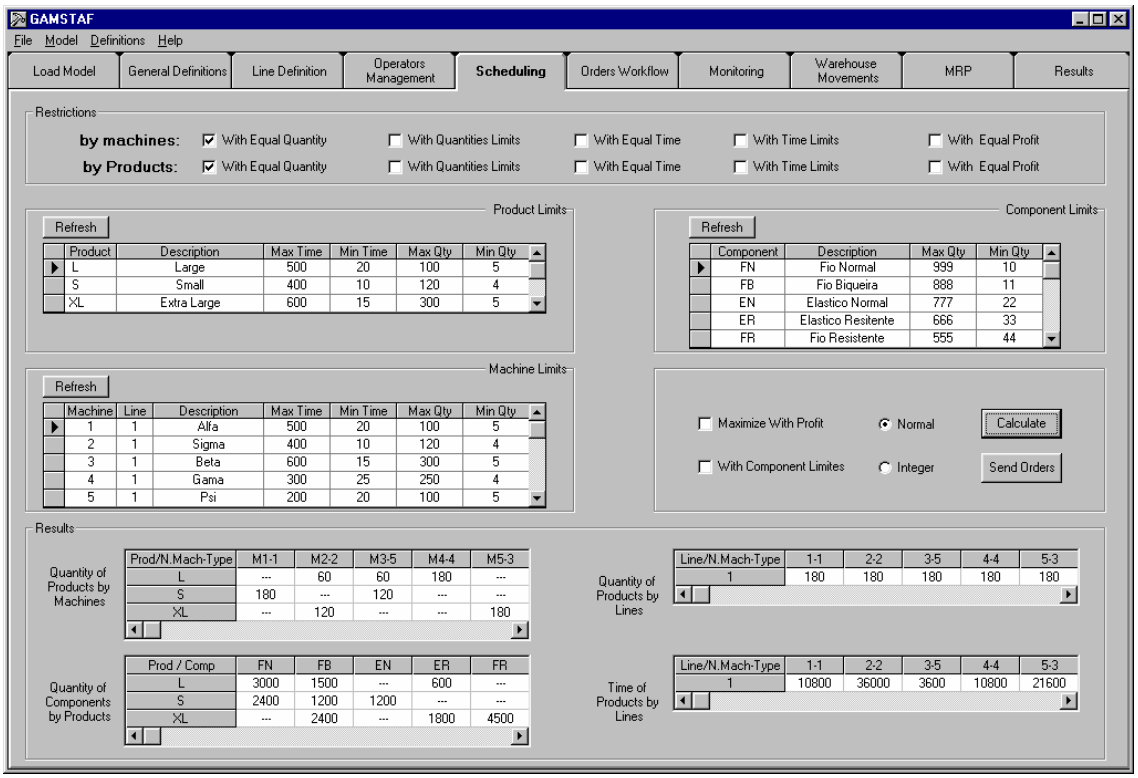


Figura 4.17 – Separador do GAMSTAF: *Scheduling*

Permite escolher a estratégia de distribuição das encomendas pelas máquinas conforme descrito no ponto 3.4 deste relatório. As estratégias podem ser aplicadas apenas às máquinas, apenas aos produtos ou combinadas.

No exemplo, todas as máquinas têm quantidade iguais de produtos, e todos os produtos são produzidos em quantidades iguais (as máquinas produzem 180, e os produtos 180).

As tabelas apresentadas retratam os valores por defeito definidos nos formulários anteriores. Estas tabelas permitem a sua alteração, bastando para isso colocar o cursor em cima do valor a alterar e escrever o novo limite.

No caso da solução não assumir valores inteiros, tem-se a opção *Integer* que resolve o problema através de programação inteira. Esta opção está implementada através de uma livraria (*lpsolve.dll Versão 2.1 - Copyright © 1999, Free Software Foundation, Inc.).*

No espaço reservado a *Results*, observa-se o resultado da aplicação da estratégia escolhida. Para terminar este processo pressiona-se *Send Orders* e a distribuição é executada.

6. Orders Workflow

Funcionalidade:

Inserir encomendas e declarar a produção.

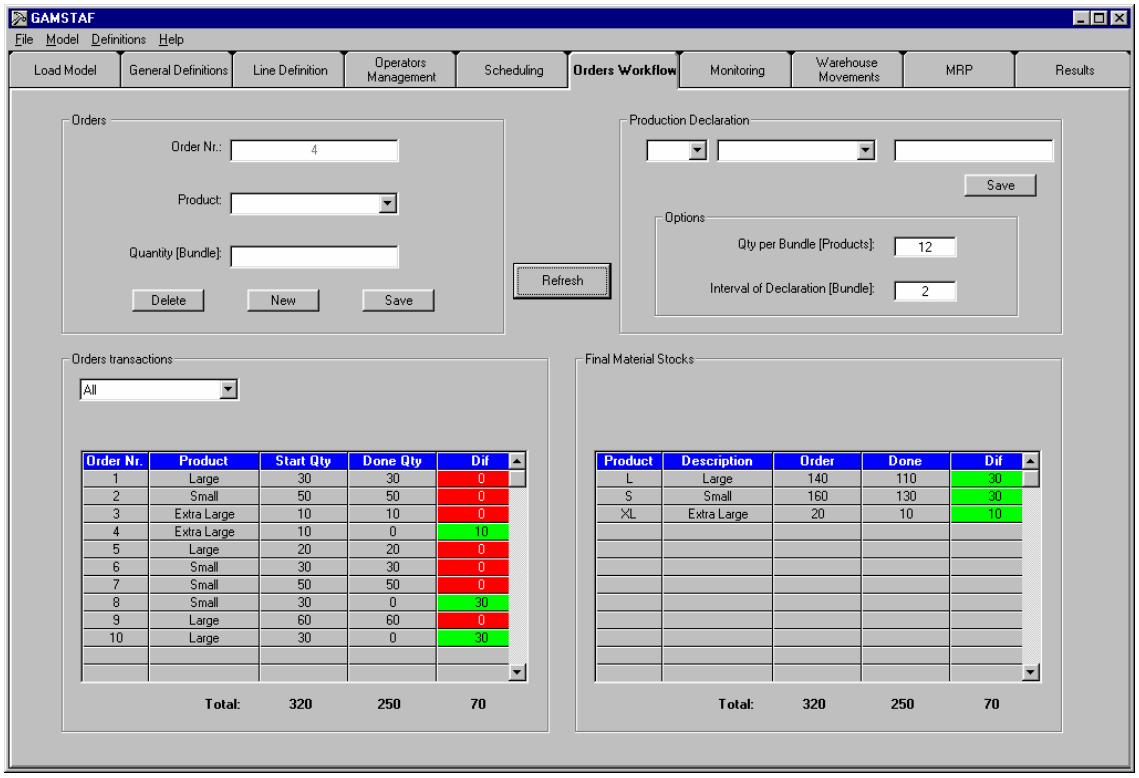


Figura 4.18 – Separador do GAMSTAF: *Orders Workflow*

Neste separador do programa dá-se entrada das encomendas. As quantidades iniciais (antes da simulação) são as que entram no cálculo das estratégias de escalonamento.

Defini-se aqui os produtos e respectivas quantidades que se pretende que o modelo produza. A simulação termina quando forem produzidas todas as quantidades aqui definidas.

A declaração de produção aparece neste menu, mas é usada de forma automática pelo *Modelo Base*, que executa a declaração assim que o operador retira o saco da máquina.

Pode-se parametrizar o tamanho do lote (quantidade de produtos) e a quantidade de lotes que a máquina produz até necessitar da presença do operador de máquina. Esta parametrização influenciará obrigatoriamente a quantidade de produtos.

Pressionado o botão *Refresh* pode-se acompanhar, em tempo real, o desenrolar da simulação em termos de quantidades de produtos declarados e consequentemente, a progressão da realização das encomendas.

7. Monitoring

Funcionalidade:

Executa a monitorização da simulação, mais propriamente do modelo.

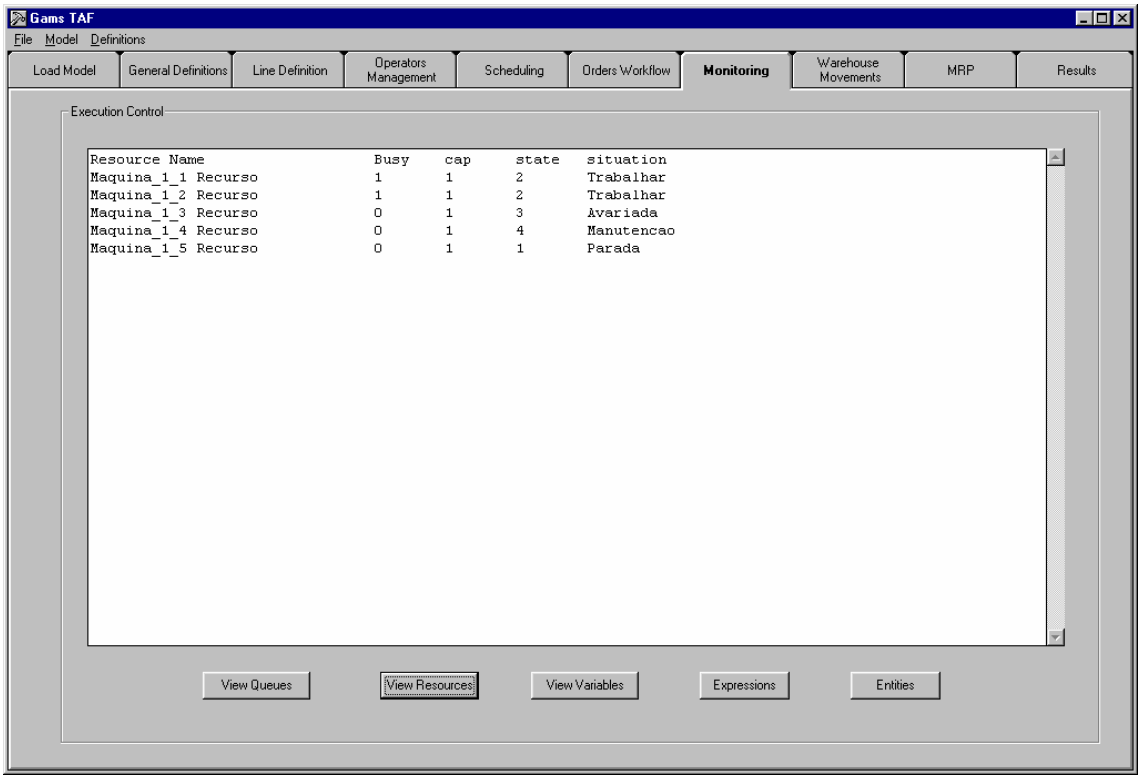


Figura 4.19 – Separador do GAMSTAF: *Monitoring*

È usado exclusivamente quando a simulação está em funcionamento. Permite ao utilizador acompanhar com mais pormenor as alterações da simulação ocorridas no modelo de simulação.

Dependendo da opção seleccionada pode-se obter informação sobre o estado dos recursos, das variáveis, das filas de espera, das expressões e das entidades usadas no modelo.

È a informação disponível pelo simulador *Arena*®, transferida para esta ferramenta e apresentada de uma forma mais organizada e ordenada.

No exemplo, observa-se a monitorização dos recursos existentes no modelo que está em funcionamento num determinado instante da simulação - cenário com cinco máquinas, todas de capacidade 1, duas estão operacionais, uma está avariada aguardando a equipa de manutenção, outra está a ser assistida pela equipa de manutenção, a última terminou o lote e aguarda pelo operador de máquina.

8. Warehouse Movements

Funcionalidade:

Seguimento das transacções efectuadas nos armazéns.

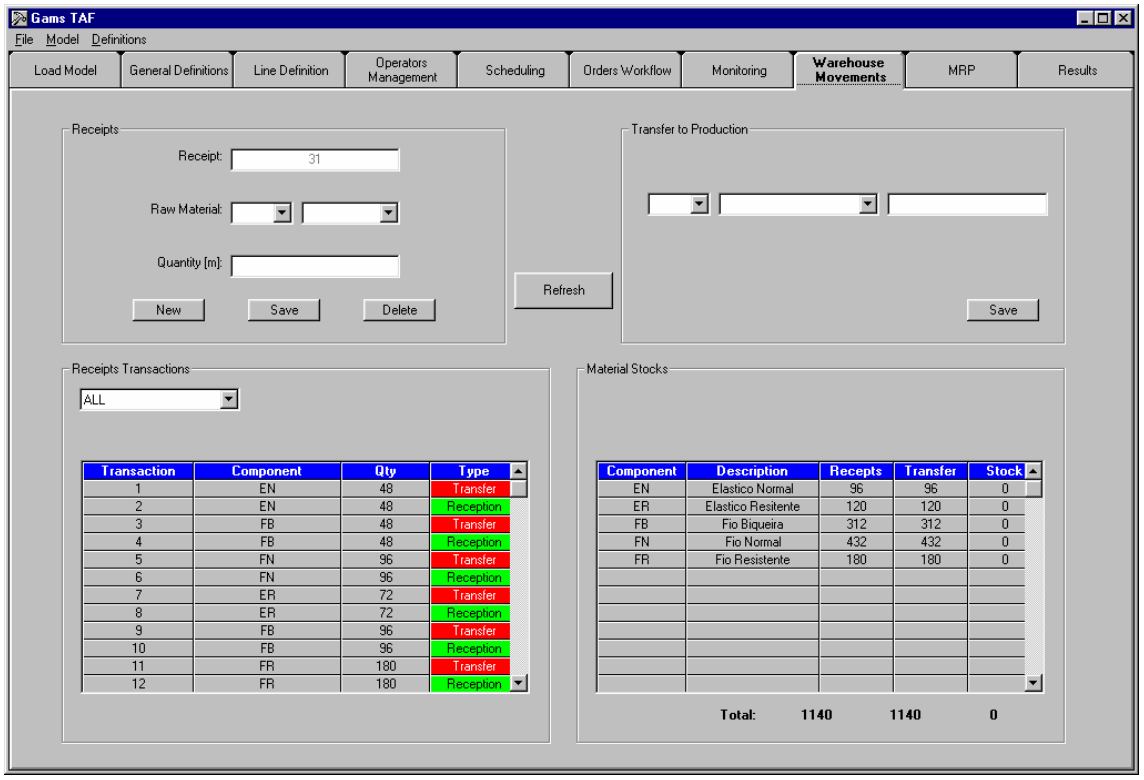


Figura 4.20 – Separador do GAMSTAF: *Warehouse Movements*

Pretende-se com este menu fazer o seguimento do consumo da matéria-prima pelo modelo de simulação.

Os materiais entram através do controlo de recepção e são transferidos para a área de produção, onde são consumidos pelas máquinas.

As transacções de matéria-prima ocorrem sempre que é atribuída nova ordem de encomenda às máquinas e os respectivos BOM´s são descontados dos armazéns.

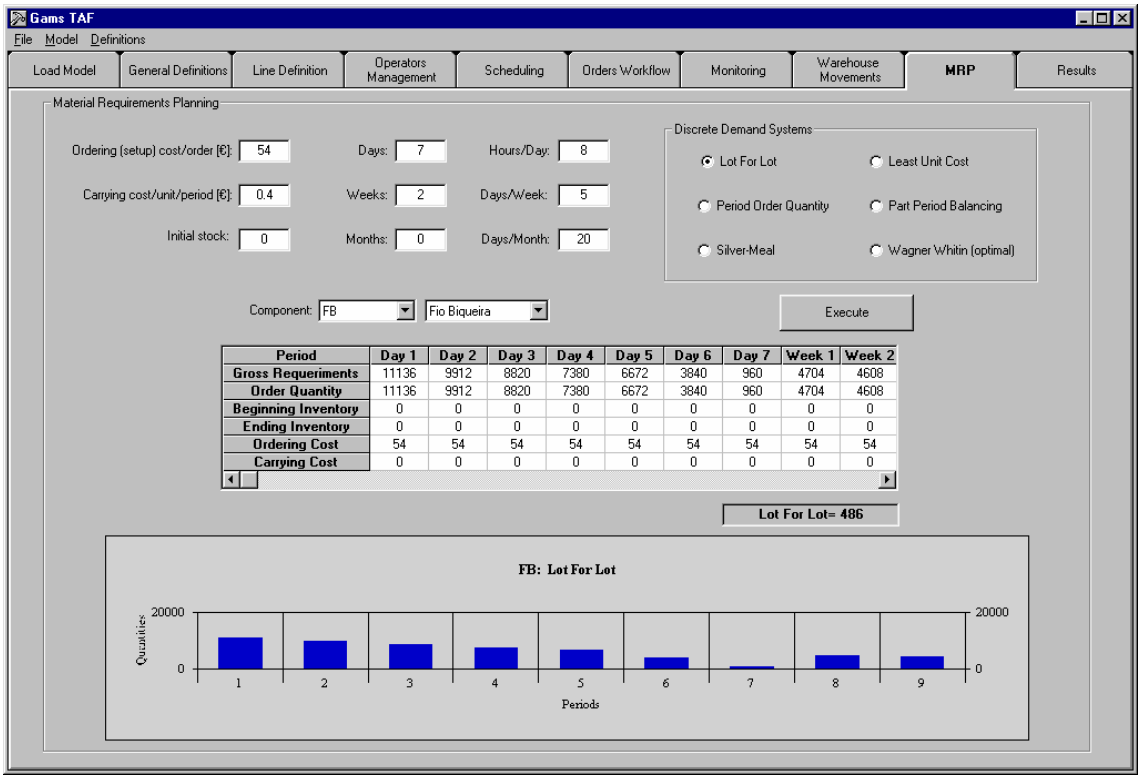
A informação das quantidades juntamente com a data em que a matéria-prima é consumida, durante a simulação, vai ser usada para definir as necessidades de compra.

O registo da informação aqui produzida pode ter outras aplicações. Pode-se inferir, por exemplo, sobre a dimensão do armazém e o número de funcionários necessários para a sua operacionalidade decorrer com a normalidade desejada.

9. MRP

Funcionalidade:

Determinar o melhor método de aquisição para a matéria-prima.



Period	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Week 1	Week 2
Gross Requeriments	11136	9912	8820	7380	6672	3840	960	4704	4608
Order Quantity	11136	9912	8820	7380	6672	3840	960	4704	4608
Beginning Inventory	0	0	0	0	0	0	0	0	0
Ending Inventory	0	0	0	0	0	0	0	0	0
Ordering Cost	54	54	54	54	54	54	54	54	54
Carrying Cost	0	0	0	0	0	0	0	0	0

Figura 4.21 – Separador do GAMSTAF: *MRP*

O principal objectivo é possibilitar, ao utilizador, alternativas para aquisição da matéria-prima. Este estudo é executado posteriormente à simulação, quando são conhecidos os valores totais da matéria-prima consumida e as datas em que foi necessária.

As alternativas implementadas neste menu resumem-se a seis dos mais conhecidos métodos determinísticos para aquisição de matéria-prima (descritos em 3.3).

A parametrização implementada é a usual para problemas deste género e a necessária para a correcta aplicação dos métodos. Toda a parametrização pode ser personalizada, e todos os parâmetros estão acessíveis para o utilizador.

Os períodos representados na grelha e no gráfico - horizonte de cálculo, representam a soma dos dias, das semanas e dos meses. Este horizonte descreve o intervalo de tempo de simulação que está a ser alvo de estudo por parte do utilizador.

10. Results

Funcionalidade:

Apresentação dos principais índices de desempenho

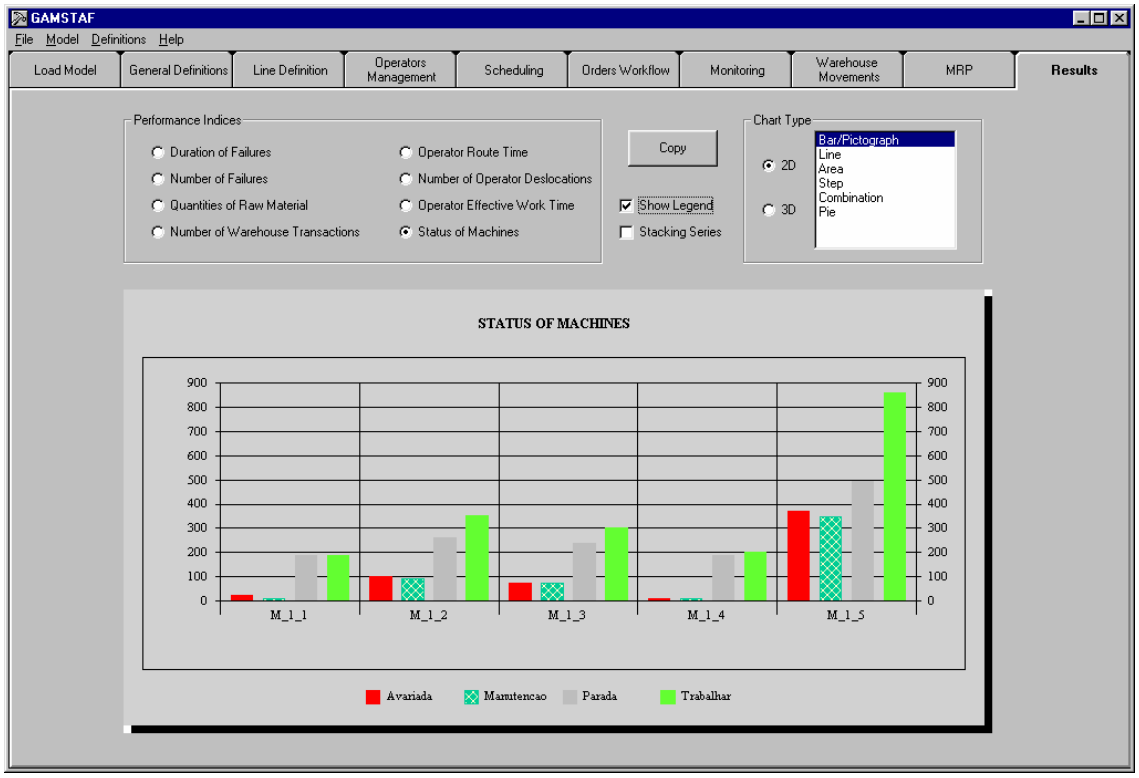


Figura 4.22 – Separador do GAMSTAF: *Results*

Para melhor percepção e análise dos resultados, os principais índices de desempenho do sistema são apresentados neste separador, sob a forma de gráficos.

Os índices de desempenho foram elaborados a pensar em melhorar a performance do sistema em simulações subsequentes.

Os índices considerados mais relevantes, são: duração das falhas, número de falhas, quantidade de matéria-prima, número de transacções no armazém, tempo do operador gasto nas deslocações, número de deslocações do operador, tempo que o operador passou a trabalhar nas máquinas, tempo de permanência das máquinas nos diferentes estados (Avariada, Manutenção, Parada, Trabalhar).

Prevendo a possibilidade de copiar o gráfico para outra aplicação, para elaborar relatórios, criou-se a funcionalidade associada ao botão *Copy* que quando pressionado o gráfico fica memorizado, bastando fazer *Paste* na aplicação onde se pretenda inserir.

4.5 O Sistema proposto: Funcionamento, Relação e Informação

Para dissipar qualquer dúvida que ainda possa restar, ao nível do funcionamento do sistema proposto, considerou-se importante descrever o fluxo de informação existente.

Através da Figura 4.23, pode-se facilmente distinguir as três fases que integram o sistema (Parametrização, Monitorização e Análise de resultados)

O processo inicia-se com a introdução das definições para o novo projecto, que interage com a base de dados, para consulta dos valores existentes e arquivo das alterações inseridas pelo utilizador.

Quando o novo cenário estiver idealizado no GAMSTAF, incluindo a selecção do *Modelo Base*, activa-se o procedimento de exportação.

As opções seleccionadas no GAMSTAF, com o recurso à base de dados, são aplicadas no *Modelo Base* (Figura 4.5), que passa então a ser o novo cenário de simulação.

Com o processo de geração do modelo terminado, resta apenas executar a simulação. Quando a simulação está a decorrer, através do GAMSTAF pode-se observar quer as alterações geradas pelo modelo quer as mudanças por este provocado no sistema.

A monitorização do modelo inclui informação sobre as variáveis, expressões, recursos, entidades e filas de espera (Figura 4.19). Pode-se também acompanhar as alterações das ordens de encomenda (Figura 4.18) e as transacções de materiais dos armazéns devido às declarações de produtos executadas pelas máquinas (Figura 4.20).

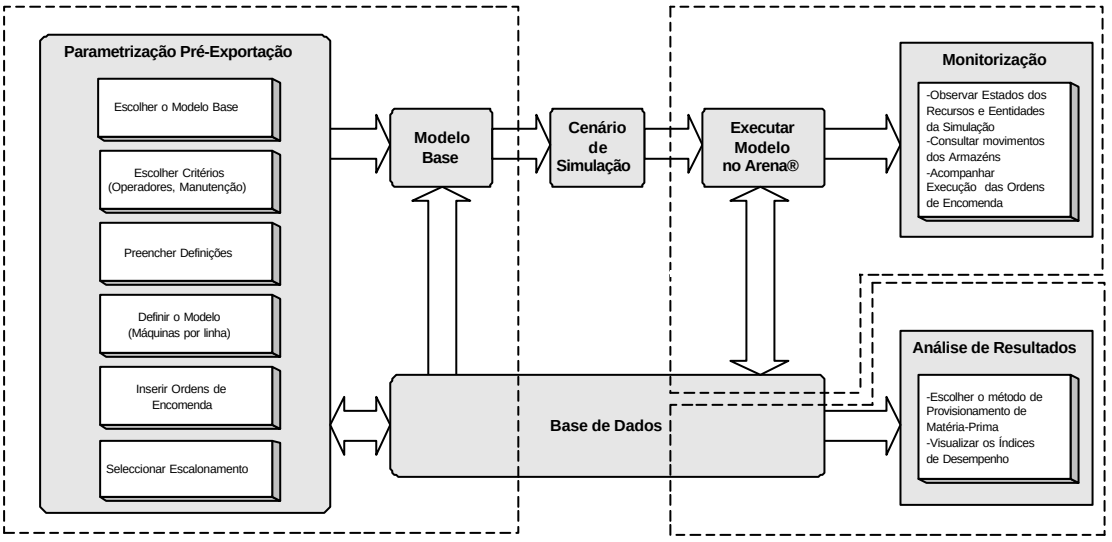


Figura 4.23 – Fluxo de informação do sistema proposto

Finalmente, a análise de resultados, possibilita efectuar um estudo, posterior à simulação, sobre o comportamento do sistema.

Baseia-se na informação contida na base de dados, com registos que foram guardados durante a execução da simulação.

A informação disponível na base de dados contém, nomeadamente:

- as ocorrências efectuadas pela equipa de manutenção e tempo dispendido na reparação da avaria;
- as deslocações do operador às máquinas e respectivos tempos gastos quer na máquina quer nas deslocações;
- as alterações de estado das máquinas (Trabalhar, Parada, Avariada, Manutenção), e a permanência das máquinas em cada um deles;
- as quantidades de produtos declarados e transacções de matéria-prima efectuadas nos armazéns.

Sobre esta informação são apresentados alguns índices, considerados mais importantes, que podem ser observados no separador *Results* (Figura 4.21).

Toda a informação está disponibilizada em base de dados, que o utilizador poderá consultar, fazer outros tratamentos da informação, se preferir, fora do sistema proposto, usar a informação noutras ferramentas, para definir novos índices de desempenho ou simplesmente para apresentações detalhadas em forma de tabelas.

4.6 Conclusão

Neste capítulo é feita a descrição do SAD e abordado o tema central desta dissertação, a geração automática de modelos de simulação. São descritos os módulo que compõem a *template* que quando incluída no simulador *Arena*® possibilita-o de os usar como se fizessem parte do próprio simulador. Esta característica permite ao *Arena*® utiliza-los num mesmo cenário, as vezes que interessar.

A ferramenta informática que ajuda o utilizador a definir as linhas de produção, o número e tipo de máquinas em cada linha, o operador e todas as parametrizações necessárias para o normal funcionamento do sistema produtivo também é descrita.

Relativamente à SAD, descreve-se a relação entre o *Modelo Base*, a ferramenta informática e o registo de dados e é justificada a existência quer do *Modelo Base* quer da ferramenta informática bem como do registo de dados.

Capítulo 5

Conclusão

5.1 Trabalho Desenvolvido

A implementação deste sistema de apoio à decisão teve a finalidade de proporcionar ao utilizador final, interactividade, flexibilidade e controlo da parametrização dos factores que podem influenciar os principais índices de desempenho.

O sistema tem a capacidade de gerar modelos automaticamente, possui uma forte componente visual, possibilitando ao utilizador uma melhor compreensão e validação do modelo. Foi construído para ser manuseado por utilizadores sem grande preparação técnica nas áreas de programação ou de simulação.

A ferramenta está dotada de diversas estratégias de controlo, ao nível da gestão dos materiais, do controlo do operador de máquina, do controlo da equipa de manutenção e do

escalonamento da produção. Foram implementadas com o propósito de facilitar ao utilizador diversas alternativas para avaliar o impacto na produção.

As estratégias de base foram elaboradas para terem um carácter genérico e flexível do ponto de vista da utilização. Se às estratégias existentes se fizer combinações entre elas, pois pode-se associar mais do que uma, resultará um enorme conjunto de especificações diferentes.

Para facilitar o estudo dos resultados e análise final dos modelos de teste resultantes da simulação, implementou-se, sob a forma de gráficos, um conjunto de índices de desempenho abrangendo os principais factores intervenientes no sistema produtivo.

5.2 Contributos e Avaliação do Trabalho

Um trabalho desta índole tornou evidente a importância da utilização da técnica de simulação para estudar e racionalizar conjuntos de especificações e estratégias em linhas de produção para a indústria.

Neste trabalho, a simulação revela a importância desta técnica para a resolução de problemas em que o controlo, a parametrização do sistema, a flexibilidade de políticas e definições dos recursos são pressupostos básicos para uma correcta avaliação do modelo em análise.

A facilidade de implementar estratégias que, além de controlar os recursos e entidades do sistema, podem ser dinâmicas ao ponto de dependerem das suas características dos recursos e entidades para determinar a acção seguinte.

A aplicação deste sistema a um caso demonstrou ser consistente e útil, facilitando todo o trabalho de definição de tarefas dos operadores e escalonamento da produção, bem como a escolha das máquinas mais adequadas à execução de determinadas encomendas.

Um contributo desta dissertação é ajudar a entender como se pode utilizar a construção de modelos de simulação para analisar fenómenos, problemas e tomadas de decisões sobre eles, isto é, evidenciar o papel da simulação nos processos de tomada de decisão e em especial nos sistemas informáticos de ajuda à tomada de decisões.

Pretende-se deixar bem patente como a simulação permite aproximar a análise e avaliação do rendimento de sistemas antes que sejam construídos, convertendo-se, assim, numa ferramenta chave de concepção de soluções de engenharia, ou para avaliar *à priori* o impacto das alterações propostas em sistemas existentes. Em ambos os casos o estudo da simulação realiza-se antes da construção do novo sistema ou da modificação do antigo, ajudando assim a eliminar ou a reduzir o risco de gargalos não previstos, infra ou extra utilização de recursos, etc.

5.3 Justificação e Ineditismo

O mercado está cada dia mais exigente e a informação é um factor que pode representar um importante diferencial competitivo entre as empresas.

A informação é justamente a alavanca principal para que o gestor possa tomar as decisões mais acertadas quanto aos seus objectivos.

O problema proposto nesta dissertação é relevante pois aborda um caso da indústria, onde as margens de lucro são muito apertadas, as quantidades produzidas são enormes e pequenas melhorias no sistema produtivo podem representar lucros bastante significativos.

O modelo proposto integra os principais conceitos de logística e produção na técnica de simulação. Através da aplicação de diversas estratégias para outras tantas áreas, é possível comparar alternativas, adoptando-se políticas de operação bem mais fundamentadas, obtendo-se um sólido conhecimento dos seus riscos e benefícios.

O ineditismo deste trabalho está caracterizado na proposta inovadora de sistematizar e integrar automaticamente numa perspectiva temporal, os principais fluxos de materiais e factores intervenientes na produção, não necessitando, sempre que se desejar efectuar novas simulações com parametrizações diferentes, redesenhar o modelo.

Para as áreas consideradas relevantes do ponto de vista científico, foram elaboradas diversas estratégias originais, possibilitando várias combinações entre elas proporcionando ao utilizador uma enorme quantidade de alternativas de controlo.

5.4 Limitações do Estudo

O presente estudo visa a construção de um SAD com características específicas para ajudar na tomada de decisão.

Devido aos objectivos do trabalho, a questões metodológicas e ao tempo empregado, entre outros motivos, o modelo foi construído com base na realidade de uma única empresa.

Desta forma, a ferramenta proposta neste estudo destina-se a dar resposta a problemas com as mesmas características das linhas de produção de meias existentes no estudo.

Porém, a aplicação do modelo proposto pode ser mais genérica, desde que seja validado como sendo uma representação fidedigna do sistema em questão.

A impossibilidade de se conceber, através do modelo construído, um procedimento específico cuja aplicação possa garantir os resultados mais favoráveis no que refere ao desempenho da produção.

Destaca-se como factor de limitação deste SAD o facto de não contemplar a escolha da melhor política para o estudo de um caso, devido ao número de variáveis envolvidas e consequentemente o número de iterações necessárias.

5.5 Perspectivas para Trabalhos Futuros

Para trabalhos futuros e baseado em projectos deste tipo, sugere-se solucionar a questão da definição da melhor política, ou do conjunto de especificações que resultaria num melhor resultado final para qualquer modelo.

Outra sugestão, não menos interessante, consiste na implementação e controlo de um sistema em tempo real, dinâmico com capacidades inteligentes para a redefinição das estratégias, do escalonamento, do operador de máquina e da equipa de manutenção.

Anexo A

GAMSTAF

```
Dim dist(), qtfila(), veloc(), tproc(), criterio(4, 1), TempoRota()
Dim valorgrafico(1) As Integer

Private Sub bom_grid_Click()
aux = bom_grid.Row
If bom_grid.Row > 0 And bom_grid.Row <= CInt(linha_bom_max.Text) Then
  bom_grid.Row = CInt(linha_bom_actual.Text)
  For i = 0 To 2
    bom_grid.Col = i
    bom_grid.CellBackColor = vbGray
    bom_grid.CellForeColor = vbBlack
  Next
  bom_grid.Row = CInt(aux)
  For i = 0 To 2
    bom_grid.Col = i
    bom_grid.CellBackColor = vbBlue
    bom_grid.CellForeColor = vbWhite
  Next
  Call posiciona_bom(bom_grid.Row)
  bom_grid.Col = 3
  linha_bom_actual.Text = CStr(aux)
End If
End Sub

Private Sub Calendar1_Click()
  Txt_calendar1.Text = Calendar1.Value
  Txt_calendar2.Text = Calendar1.Month
  MonthView1.Value = Calendar1.Value
  Txt_calendar3.Text = MonthView1.DayOfWeek
End Sub

Private Sub Check_res_grafico_legend_Click()
  Call Opt_tipo_grf_Click(Check_res_grafico_legend.Value)
End Sub

Private Sub Check_res_grafico_stack_Click()
  Call Opt_tipo_grf_Click(Check_res_grafico_stack.Value)
End Sub

Private Sub cmd_adiciona_maquina_a_linha_Click()
Dim tipo_maquina As String
Dim linha As String

tipo_maquina = Combo_machine_cod.List(Combo_machine_cod.ListIndex)
linha = Combo1.List(Combo1.ListIndex)

If CInt(Val(tipo_maquina)) <= 0 Then
  MsgBox "ERROR: Machine Not Selected ...", vbExclamation
  Exit Sub
End If

If CInt(txt_imagem_op.Text) <= 0 Then
  MsgBox "ERROR: Image Not Selected ...", vbExclamation
  Exit Sub
End If
If devolve_count_maquinas_linha(linha) <= 60 Then
  With Data2.Recordset
    .AddNew
    !Line = linha
    ![Machine_cod] = tipo_maquina
    .Update
  End With
  desenha_global_uma (CInt(linha))
  For i = 1 To grid.rows
    grid.Col = 2
    grid.Row = i
    If CInt(grid.Text) = tipo_maquina Then
      grid.Col = 1
      grid.Text = CStr(CInt(grid.Text) + 1)
      Exit For
    End If
  Next i
  grid.Col = 1
  grid.Row = grid.rows - 14
  grid.Text = CStr(CInt(grid.Text) + 1)
 Else
    MsgBox "The maximum Number of machines for line is 60", 16
End If

Call grid_prod_refresh_Click
Call grid_mach_refresh_Click
Call grid_comp_refresh_Click

End Sub

Private Sub cmd_delete_maquina_a_linha_Click()
'this may produce an error if you delete the last
'record or the only record in the recordset
On Error Resume Next
Dim linha As String
Dim tipo_maquina As String
Dim sql As String

tipo_maquina = Combo_machine_cod.List(Combo_machine_cod.ListIndex)
linha = Combo1.List(Combo1.ListIndex)
sql = "line=" & linha & " and machine_cod=" & tipo_maquina & ""
If CInt(Val(tipo_maquina)) <= 0 Then
  MsgBox "ERROR: Machine Not Selected ...", vbExclamation
  Exit Sub
End If

If devolve_count_maquinas(tipo_maquina, linha) = 1 Then
  grid_global.Col = linha
  For j = 1 To grid.rows - 14
    grid_global.Row = j
    grid_global.Text = "0"
```

```
  Next j
  For i = j To grid_global.rows
    grid_global.Col = linha
    grid_global.Row = i
    grid_global.Text = ""
  Next i
End If
If devolve_count_maquinas(tipo_maquina, linha) > 0 Then
  Data2.Recordset.FindFirst sql
  Data2.Recordset.Delete
  Data2.Recordset.MovePrevious
  Data2.Refresh
  desenha_global_uma (CInt(Combo1.List(Combo1.ListIndex)))
  For i = 1 To grid.rows
    grid.Col = 2
    grid.Row = i
    If CInt(grid.Text) = tipo_maquina Then
      grid.Col = 1
      grid.Text = CStr(CInt(grid.Text) - 1)
      Exit For
    End If
  Next i
  grid.Col = 1
  grid.Row = grid.rows - 14
  grid.Text = CStr(CInt(grid.Text) - 1)
End If

Call grid_prod_refresh_Click
Call grid_mach_refresh_Click
Call grid_comp_refresh_Click

End Sub

Private Sub cmd_up2_Click()
Data2.UpdateRecord
Data2.Recordset.Bookmark = Data2.Recordset.LastModified
End Sub

Private Sub Cmd_envia_orders_Click()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_manufacturing", dbOpenTable)
Call delete_CTRL_planned_orders
qtd_ordem = CLng(qty_declaration.Text)
qtd_lote = CLng(txt_qtd_batch.Text)
num_lotes = 1
lot = 0
rs.MoveFirst
While Not rs.EOF
  With rs
    rs.Edit
    maquina = rs.Fields("machine")
    produto = rs.Fields("product")
    quant = rs.Fields("qty")
    qt = 0
    lotes = Int((quant * qtd_lote) / (qtd_lote * qtd_ordem))
    lot = lot + 1

    qtd = qtd_lote * qtd_ordem

    For i = 1 To lotes
      Call   escreve_CTRL_planned_orders(CStr(maquina),   CLng(produto),   CLng(qtd),   CLng(num_lotes),
CLng(lot))
      qt = qtd + qt
      num_lotes = num_lotes + 1
    Next i
    If qt < quant * qtd_lote Then
'        Call escreve_CTRL_planned_orders(CStr(maquina), CLng(produto), CLng(quant * qtd_lote  - qt),
CLng(num_lotes), CLng(lot))
      Call   escreve_CTRL_planned_orders(CStr(maquina),   CLng(produto),   CLng(qtd),   CLng(num_lotes),
CLng(lot))
      num_lotes = num_lotes + 1
    End If
    End With
  rs.MoveNext
Wend
End Sub

Private Sub Combo_bom_component_cod_Click()
Combo_bom_component_desc.ListIndex = Combo_bom_component_cod.ListIndex
txt_component_qty.Text = devolve_qty_bom(CStr(Prod_cod_bom.Text), CStr(Combo_bom_component_cod.Text))
End Sub

Private Sub Combo_bom_component_desc_Click()
Combo_bom_component_cod.ListIndex = Combo_bom_component_desc.ListIndex
End Sub

Private Sub Combo_dec_prod_cod_Click()
Combo_dec_prod_desc.ListIndex = Combo_dec_prod_cod.ListIndex
txt_component_qty.Text                  =                  CLng(devolve_qty_bom(CStr(Prod_cod_bom.Text),
CStr(Combo_bom_component_cod.Text)))
End Sub

Private Sub Combo_dec_prod_desc_Click()
Combo_dec_prod_cod.ListIndex = Combo_dec_prod_desc.ListIndex
End Sub

Private Sub Combo_machine_type_Click()
Combo_machine_cod.ListIndex = Combo_machine_type.ListIndex
End Sub

Private Sub Combo_machine_type_KeyPress(KeyAscii As Integer)
Combo_machine_cod.ListIndex = Combo_machine_type.ListIndex
End Sub
Sub mensagem_menu(mostra As Boolean)
If mostra Then
form_msg_calculo.Show
```

```
form_msg_calculo.Left = (Screen.Width - form_msg_calculo.Width) / 2
form_msg_calculo.Top = (Screen.Height - form_msg_calculo.Height) / 2
Else
form_msg_calculo.Hide
End If
form_msg_calculo.Refresh
End Sub

Private Sub cmd_calcula_distribuicao_Click()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_manufacturing", dbOpenTable)

Dim maquinas_tipos(), produtos_nome(), r(), mat(), tot_mat(), maqlinha(), max_comp()
Dim timeprocess(), profit(), bi(), maxmaq(), tempo_max(), linhasactivas(), max_prod()
Dim resposta As Boolean
Dim objectivo As Double
Dim texto As String
Dim opcao As Integer

Call mensagem_menu(True)

Frame(40).Visible = True
num_maquinas = devolve_num_maquinas()
num_materiais = devolve_num_materiais_usados()
ReDim profit(num_produtos)
produtos_nome = devolve_produtos_usados
num_produtos = UBound(produtos_nome)

tempo_max = devolve_limites_tempo(CLng(num_maquinas))
maxmaq = devolve_limites_quantidades(CLng(num_maquinas))
max_comp = devolve_limites_materiais(CLng(num_materiais))
tempo_prod = devolve_limites_tempo_produtos(CLng(num_produtos))
max_prod = devolve_limites_quantidades_produtos(CLng(num_produtos))

ReDim maquinas_tipos(num_produtos, num_maquinas)
ReDim r(num_produtos, num_maquinas)
ReDim materiais_tipos(num_materiais)
ReDim timeprocess(num_produtos, num_maquinas)
ReDim profit(num_produtos)
ReDim bi(num_produtos)
ReDim mat(num_produtos, num_materiais)
ReDim tot_mat(num_materiais)

Label(59).Visible = True ' calculating
maquinas_tipos = devolve_tipo_maquinas(CLng(num_maquinas))
materiais_tipos = devolve_tipo_materiais_usados()

For i = 1 To num_produtos
  For j = 1 To num_maquinas
    resposta = permite_maquina_produto(CStr(maquinas_tipos(j)), CStr(produtos_nome(i)))
    If resposta Then r(i, j) = 1 Else r(i, j) = 0
    profit(i) = CLng(devolve_profit_produto(CStr(produtos_nome(i))))
    timeprocess(i,       j)       =       CLng(devolve_tempo_processo(CStr(produtos_nome(i))))       /
CLng(devolve_velocidade_maquina(CStr(maquinas_tipos(j))))
  Next j
  bi(i) = devolve_orders_produto(CStr(produtos_nome(i))) - devolve_stock_produto(CStr(produtos_nome(i)))
  For k = 1 To num_materiais
      mat(i, k) = devolve_qty_material(CStr(materiais_tipos(k)), CStr(produtos_nome(i)))
  Next k
Next i
For k = 1 To num_materiais
  tot_mat(k) = devolve_stock(CStr(materiais_tipos(k)))
Next k

grid_mat.Visible = False
grid_result.Visible = False
grid_maquinas.Visible = False
grid_ocupacao.Visible = False

Label(66).Visible = False
Label(67).Visible = False
Label(68).Visible = False
Label(69).Visible = False

Call desenha_grelha_result
Call delete_table_CTRL_manufacturing
Call faz_matriz(CLng(num_produtos), CLng(num_maquinas), r, bi, timeprocess, tempo_max, maxmaq, profit,
max_comp, max_prod, tempo_prod)
Label(59).Visible = False            ' calculating
If Not existe_solucao Then Exit Sub  'nao existe solucoes ....
'determina as maquinas por linhas usadas
linhas = devolve_count_linhas()
ReDim linhasactivas(linhas)
ReDim maqlinha(linhas)
fim = 0
For i = 1 To 40
  qt = devolve_count_maquinas_linha(CLng(i))
  If qt > 0 Then
    linhasactivas(fim + 1) = i
    maqlinha(fim + 1) = qt
    fim = fim + 1
   End If
  If fim = linhas Then Exit For
Next i
For i = 1 To num_produtos
For j = 1 To num_maquinas
  grid_result.Row = i
  grid_result.Col = j
  X = CDbl(Val(grid_result.Text))
  texto = grid_result.Text
  If X > 0 Then
    v = Len(texto)
    For t = 1 To v
    KK = Mid$(texto, t, 1)
      If (Mid$(texto, t, 1) = ",") Then Mid$(texto, t, 1) = "."
    Next t
```

```
    End If
     X = CDbl(Val(texto))
     mat(i, 0) = mat(i, 0) + X
    Next j
Next i
Call desenha_grelha_materiais
For i = 1 To num_produtos
  grid_mat.Row = i
  grid_mat.Col = 0
  produto = grid_mat.Text
  For j = 1 To num_materiais
    grid_mat.Col = j
    grid_mat.Row = 0
    grid_mat.CellAlignment = flexAlignCenterCenter
    If pertence_bom(CStr(produto), CStr(grid_mat.Text)) Then
      grid_mat.Row = i
      grid_mat.CellAlignment = flexAlignCenterCenter
      grid_mat.Text = CDbl(mat(i, j) * mat(i, 0))
      Else
       grid_mat.Row = i
       grid_mat.CellAlignment = flexAlignCenterCenter
       grid_mat.Text = "---"
    End If
  Next j
 Next i
grid_mat.Visible = True
Call desenha_grelha_maquinas
grid_maquinas.Visible = True
Call desenha_grelha_ocupacao

rs.MoveFirst
While Not rs.EOF
  With rs
    rs.Edit
    i = CLng(rs.Fields("line_prod"))
    j = CLng(rs.Fields("number"))
    c = CLng(rs.Fields("product"))
    grid_ocupacao.Row = i
    grid_ocupacao.Col = j
    grid_ocupacao.CellAlignment = flexAlignCenterCenter
    grid_ocupacao.Text = CDbl(rs.Fields("qty")) * timeprocess(c, j) + CDbl(Val(grid_ocupacao.Text))
grid_ocupacao.Visible = True
End With
rs.MoveNext
Wend
grid_ocupacao.Visible = True
Call mensagem_menu(False)
End Sub

Private Sub Combo_mrp_component_cod_Click()
Combo_mrp_component_desc.ListIndex = Combo_mrp_component_cod.ListIndex
End Sub

Private Sub Combo_mrp_component_desc_Click()
Combo_mrp_component_cod.ListIndex = Combo_mrp_component_desc.ListIndex
End Sub

Private Sub delete_component_Click()
On Error Resume Next
sql = "SELECT bom_cod, component, qty "
sql = sql & "From DEF_bom "
sql = sql & "WHERE DEF_bom.bom_cod='" & Prod_cod_bom.Text & " ' and DEF_bom.component = '" &
Combo_bom_component_cod.Text & "'"
With DataBD
      .RecordSource = sql
      .Refresh
End With
If DataBD.Recordset.RecordCount = 1 Then
  DataBD.Recordset.Delete
End If
DataBD.Recordset.Close
Call desenha_grelha_BOM(CStr(Prod_cod_bom.Text))
End Sub

Private Sub esconde_grelhas_Click()
Frame(40).Visible = False
Frame(44).Visible = False
Frame(45).Visible = False
Frame(46).Visible = False
DBGrid_limites.Visible = False
DBGrid_produtos.Visible = False
DBGrid_componentes.Visible = False
DBGrid_limites.Visible = False
DBGrid_produtos.Visible = False
DBGrid_componentes.Visible = False
grid_mach_refresh.Visible = False
grid_prod_refresh.Visible = False
grid_comp_refresh.Visible = False
End Sub

Private Sub grid_comp_refresh_Click()
Call actualiza_TEMP_componentes
DBGrid_componentes.Visible = True
End Sub

Private Sub grid_mach_refresh_Click()
Call actualiza_TEMP_maquinas
DBGrid_limites.Visible = True
End Sub

Private Sub grid_prod_refresh_Click()
Call actualiza_TEMP_produtos
DBGrid_produtos.Visible = True
End Sub

Private Sub List_res_grafico_Click()
```

```
 If Opt_tipo_grf(0) = True Then
  txt_res_grafico2d.Text = List_res_grafico.ListIndex
  opcao = List_res_grafico.ListIndex
 Else
  txt_res_grafico3d.Text = List_res_grafico.ListIndex
  opcao = List_res_grafico.ListIndex + 7
End If

  Select Case opcao
    Case 0: txt_res_grafico.Text = 1 '
    Case 1: txt_res_grafico.Text = 3
    Case 2: txt_res_grafico.Text = 5 '
    Case 3: txt_res_grafico.Text = 7 '
    Case 4: txt_res_grafico.Text = 9 '
    Case 5: txt_res_grafico.Text = 14 '
    Case 6: txt_res_grafico.Text = 16 '
    Case 7: txt_res_grafico.Text = 0 '
    Case 8: txt_res_grafico.Text = 2
    Case 9: txt_res_grafico.Text = 4
    Case 10: txt_res_grafico.Text = 6 '
    Case 11: txt_res_grafico.Text = 8 '
End Select

 Call cmdGraphics_Click
End Sub
Private Sub List1_Click()
  txt = Trim(str(List1.ListIndex + 1))
  If List1.ListIndex + 1 < 10 Then txt = "0" + Trim(str(List1.ListIndex + 1))
  ficheiro = "wire" + txt + ".bmp"
  directoria = CurDir
  ficheiro = directoria & "\images\" + ficheiro
  Set Imagem1.Picture = LoadPicture(ficheiro)
  txt_prod_image.Text = List1.ListIndex + 1
End Sub

Private Sub List2_Click()
  txt = Trim(str(List2.ListIndex + 1))
  If List2.ListIndex + 1 < 10 Then txt = "0" + Trim(str(List2.ListIndex + 1))
  ficheiro = "opr" + txt + ".bmp"
  directoria = CurDir
  ficheiro = directoria & "\images\" + ficheiro
'  ficheiro = "d:\master\images\" + ficheiro
  Set Imagem2.Picture = LoadPicture(ficheiro)
  txt_imagem_op.Text = List2.ListIndex + 1
End Sub

Private Sub List3_Click()
  txt = Trim(str(List3.ListIndex + 1))
  If List3.ListIndex + 1 < 10 Then txt = "0" + Trim(str(List3.ListIndex + 1))
  ficheiro = "Prod" + txt + ".bmp"
  'ficheiro = "d:\master\images\" + ficheiro
  directoria = CurDir
  ficheiro = directoria & "\images\" + ficheiro
  Set Imagem3.Picture = LoadPicture(ficheiro)
  txt_final_image.Text = List3.ListIndex + 1
End Sub

Private Sub List4_Click()
  txt = Trim(str(List4.ListIndex + 1))
  If List4.ListIndex + 1 < 10 Then txt = "0" + Trim(str(List4.ListIndex + 1))
  ficheiro = "maq" + txt + ".bmp"
  directoria = CurDir
  ficheiro = directoria & "\images\" + ficheiro
'  ficheiro = "d:\master\images\" + ficheiro
  Set Imagem4.Picture = LoadPicture(ficheiro)
  txt_machine_image.Text = List4.ListIndex + 1
End Sub

Private Sub List5_Click()
  txt = Trim(str(List5.ListIndex + 1))
  If List5.ListIndex + 1 < 10 Then txt = "0" + Trim(str(List5.ListIndex + 1))
  ficheiro = "palete" + txt + ".bmp"
'  ficheiro = "d:\master\images\" + ficheiro
  directoria = CurDir
  ficheiro = directoria & "\images\" + ficheiro
  Set Imagem5.Picture = LoadPicture(ficheiro)
  txt_batch_image.Text = List5.ListIndex + 1
End Sub

Private Sub List6_Click()
 ' txt = Trim(str(List6.ListIndex + 1))
 ' If List6.ListIndex + 1 < 10 Then txt = "0" + Trim(str(List6.ListIndex + 1))
 ' ficheiro = "man" + txt + ".bmp"
 ' ficheiro = "d:\master\images\" + ficheiro
 ' Set Imagem6.Picture = LoadPicture(ficheiro)
 'txt_manutencao_Image = List6.ListIndex + 1
End Sub

Private Sub mnabout_Click()
Form_About.Left = (Screen.Width - Form_About.Width) / 2
Form_About.Top = (Screen.Height - Form_About.Height) / 2
Form_About.Show
End Sub

Private Sub mnauthor_Click()
Form_Author.Left = (Screen.Width - Form_Author.Width) / 2
Form_Author.Top = (Screen.Height - Form_Author.Height) / 2
Form_Author.Show
End Sub

Private Sub mnexport_Click()
Call Command_model_export_Click
End Sub

Private Sub mnload_Click()
```

```
SSTab1.Tab = 0
End Sub

Private Sub mnmachines_Click()
SSTab1.Tab = 1
SSTab2.Tab = 0
End Sub

Private Sub mnNew_Click()
Text3.Text = "New Model"
Model_FileName = ""
End Sub

Private Sub mnOperators_Click()
SSTab1.Tab = 1
SSTab2.Tab = 1
End Sub

Private Sub mnProduct_Struture_Click()
SSTab1.Tab = 1
SSTab2.Tab = 3
End Sub

Private Sub mnProducts_definition_Click()
SSTab1.Tab = 1
SSTab2.Tab = 2
End Sub

Private Sub mnquit_Click()
 Quit
End Sub

Private Sub Opt_tipo_grf_Click(Index As Integer)
List_res_grafico.Clear
 If Opt_tipo_grf(0) = True Then
  List_res_grafico.TabIndex = 5
  List_res_grafico.List(0) = "Bar/Pictograph"
  List_res_grafico.List(1) = "Line"
  List_res_grafico.List(2) = "Area"
  List_res_grafico.List(3) = "Step"
  List_res_grafico.List(4) = "Combination"
  List_res_grafico.List(5) = "Pie"
  'List_res_grafico.List(6) = "XY (Scatter)"
   List_res_grafico.ListIndex = txt_res_grafico2d.Text
 Else
  List_res_grafico.TabIndex = 4
  List_res_grafico.List(0) = "Bar (Column)"
  List_res_grafico.List(1) = "Line (Tape"
  List_res_grafico.List(2) = "Area"
  List_res_grafico.List(3) = "Step"
  List_res_grafico.List(4) = "Combination"
  List_res_grafico.ListIndex = txt_res_grafico3d.Text
End If
  Call List_res_grafico_Click
End Sub

Private Sub Optionres_Click(Index As Integer)

If valorgrafico(1) = 6 Then 'And Index <> 6 Then
' restaura
'res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 255, 0, 0      ' vermelho
res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 217, 177, 42  ' amarelado
res_grafico.Plot.SeriesCollection(2).DataPoints(-1).Brush.FillColor.Set 100, 255, 50    ' verde
res_grafico.Plot.SeriesCollection(3).DataPoints(-1).Brush.FillColor.Set 0, 0, 255  ' azul
res_grafico.Plot.SeriesCollection(2).DataPoints(-1).Brush.Style = VtBrushStyleSolid
res_grafico.Plot.SeriesCollection(4).DataPoints(-1).Brush.FillColor.Set 255, 255, 0      'amarelo
'res_grafico.Plot.SeriesCollection(5).DataPoints(-1).Brush.FillColor.Set 255, 0, 255      'rosa
'Else
'res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 214, 177, 42  ' amarelado

End If

valorgrafico(1) = Index
Call cmdGraphics_Click
End Sub

Private Sub prod_cod_aux_Change()
prod_cod.Text = Trim(prod_cod_aux)
End Sub

Private Sub qty_declaration_Change()
 If CLng(qty_declaration.Text) > CLng(txt_qtd_batch.Text) Then
    MsgBox "ERROR: The Interval of Declaration Must Be Less than Qty per Bundle. ...", vbExclamation
    qty_declaration.Text = 1
 End If
 Call txt_qtd_batch_Change
End Sub

Private Sub refresca_man_management_Click()
On Error Resume Next
sql = "SELECT * "
sql = sql & "From DEF_man_table "

With DataBD
      .RecordSource = sql
      .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
  If DataBD.Recordset.Fields("first") <> "" Then
    Combo_man_management1.Text = DataBD.Recordset.Fields("first")
    If DataBD.Recordset.Fields("first_v") = 0 Then Option9.Value = True Else Option10.Value = True
     Else
      Combo_man_management1.Text = ""
      Option14.Value = True
  End If
  If DataBD.Recordset.Fields("second") <> "" Then
```

```
    Combo_man_management2.Text = DataBD.Recordset.Fields("second")
    If DataBD.Recordset.Fields("second_v") = 0 Then Option11.Value = True Else Option12.Value = True
     Else
      Combo_man_management2.Text = ""
      Option14.Value = True
  End If
  If DataBD.Recordset.Fields("third") <> "" Then
    Combo_man_management3.Text = DataBD.Recordset.Fields("third")
    If DataBD.Recordset.Fields("third_v") = 0 Then Option13.Value = True Else Option14.Value = True
     Else
      Combo_man_management3.Text = ""
      Option14.Value = True
  End If
  If DataBD.Recordset.Fields("fourth") <> "" Then
    Combo_man_management4.Text = DataBD.Recordset.Fields("fourth")
    If DataBD.Recordset.Fields("fourth_v") = 0 Then Option15.Value = True Else Option16.Value = True
     Else
      Combo_man_management4.Text = ""
      Option14.Value = True
  End If
  If DataBD.Recordset.Fields("fifth") <> "" Then
    Combo_man_management5.Text = DataBD.Recordset.Fields("fifth")
    If DataBD.Recordset.Fields("fifth_v") = 0 Then Option18.Value = True Else Option17.Value = True
     Else
      Combo_man_management5.Text = ""
      Option14.Value = True
  End If
End If
End Sub

Private Sub refresca_ordens_Click()
Call desenha_grelha_produto_stock
Call desenha_grelha_ordens
End Sub

Private Sub refresca_warehouse_Click()
Call desenha_grelha_receipts
Call desenha_grelha_stock
End Sub

Private Sub save_component_Click()
On Error Resume Next
sql = "SELECT bom_cod, component, qty "
sql = sql & "From DEF_bom "
sql = sql & "WHERE DEF_bom.bom_cod='" & Prod_cod_bom.Text & " ' and DEF_bom.component = '" &
Combo_bom_component_cod.Text & "'"
With DataBD
      .RecordSource = sql
      .Refresh
End With
If DataBD.Recordset.RecordCount <= 0 Then
  DataBD.Recordset.AddNew
  DataBD.Recordset.Fields("bom_cod") = CStr(Prod_cod_bom.Text)
  DataBD.Recordset.Fields("component") = CStr(Combo_bom_component_cod.Text)
  DataBD.Recordset.Fields("qty") = CLng(txt_component_qty.Text)
  DataBD.Recordset.Update
 Else
  DataBD.Recordset.Edit
  DataBD.Recordset.Fields("qty") = CLng(txt_component_qty.Text)
  DataBD.Recordset.Update
End If
DataBD.Recordset.Close
Call desenha_grelha_BOM(CStr(Prod_cod_bom.Text))
End Sub

Private Sub save_man_management_Click()
Dim rs As Recordset
Dim bd As Database
Dim opt(5) As Integer

If Combo_man_management1.Text = "" Then
  If Combo_man_management2.Text <> "" Or Combo_man_management3.Text <> "" Or Combo_man_management4.Text
<> "" Then
     MsgBox "ERROR: The Sort reference is not valid ...", vbExclamation
      Exit Sub
   End If
End If
If Combo_man_management2.Text = "" Then
  If Combo_man_management3.Text <> "" Or Combo_man_management4.Text <> "" Then
     MsgBox "ERROR: The Sort reference is not valid ...", vbExclamation
      Exit Sub
   End If
End If
If Combo_man_management3.Text = "" Then
  If Combo_man_management4.Text <> "" Then
     MsgBox "ERROR: The Sort reference is not valid. ...", vbExclamation
      Exit Sub
   End If
End If

If Combo_man_management4.Text = "" Then
  If Combo_man_management5.Text <> "" Then
     MsgBox "ERROR: The Sort reference is not valid. ...", vbExclamation
      Exit Sub
   End If
End If

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_man_table", dbOpenTable)
    If Option9.Value = True Then opt(1) = 0 Else opt(1) = 1
    If Option11.Value = True Then opt(2) = 0 Else opt(2) = 1
    If Option13.Value = True Then opt(3) = 0 Else opt(3) = 1
    If Option15.Value = True Then opt(4) = 0 Else opt(4) = 1
    If Option18.Value = True Then opt(5) = 0 Else opt(5) = 1
 With rs
    .MoveLast
```

```
    .MoveFirst
     If Not .NoMatch Then
        .Edit
        If Combo_man_management1.Text <> "" Then !First = Combo_man_management1.Text: !first_v = opt(1)
        If  Combo_man_management2.Text  <>  ""  Then  !Second  =  Combo_man_management2.Text:  !second_v  =
opt(2)
        If Combo_man_management3.Text <> "" Then !third = Combo_man_management3.Text: !third_v = opt(3)
        If  Combo_man_management4.Text  <>  ""  Then  !fourth  =  Combo_man_management4.Text:  !fourth_v  =
opt(4)
        If Combo_man_management5.Text <> "" Then !fifth = Combo_man_management5.Text: !fifth_v = opt(5)
        .Update
     End If
  End With
rs.Close
bd.Close
End Sub

Private Sub save_opr_management_Click()
Dim rs As Recordset
Dim bd As Database
Dim opt(4) As Integer

If Combo_opr_management.Text < 1 Then
   MsgBox "ERROR: The Sort reference is not valid ...", vbExclamation
  Exit Sub
End If
If Combo_opr_management1.Text = "" Then
 If Combo_opr_management2.Text <> "" Or Combo_opr_management3.Text <> "" Or Combo_opr_management4.Text
<> "" Then
     MsgBox "ERROR: The Sort reference is not valid ...", vbExclamation
      Exit Sub
   End If
End If
 If Combo_opr_management2.Text = "" Then
  If Combo_opr_management3.Text <> "" Or Combo_opr_management4.Text <> "" Then
     MsgBox "ERROR: The Sort reference is not valid ...", vbExclamation
      Exit Sub
   End If
End If
If Combo_opr_management3.Text = "" Then
  If Combo_opr_management4.Text <> "" Then
     MsgBox "ERROR: The Sort reference is not valid. ...", vbExclamation
      Exit Sub
   End If
End If
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_opr_table", dbOpenTable)
    If Option1.Value = True Then opt(1) = 0 Else opt(1) = 1
    If Option3.Value = True Then opt(2) = 0 Else opt(2) = 1
    If Option5.Value = True Then opt(3) = 0 Else opt(3) = 1
    If Option7.Value = True Then opt(4) = 0 Else opt(4) = 1
 With rs
    .Index = "PrimaryKey"
    .MoveLast
    .MoveFirst
    .Seek "=", CInt(Combo_opr_management.Text)
     If Not .NoMatch Then
        .Edit
        !Line = CInt(Combo_opr_management.Text)
        If Combo_opr_management1.Text <> "" Then !First = Combo_opr_management1.Text: !first_v = opt(1)
        If  Combo_opr_management2.Text  <>  ""  Then  !Second  =  Combo_opr_management2.Text:  !second_v  =
opt(2)
        If Combo_opr_management3.Text <> "" Then !third = Combo_opr_management3.Text: !third_v = opt(3)
        If  Combo_opr_management4.Text  <>  ""  Then  !fourth  =  Combo_opr_management4.Text:  !fourth_v  =
opt(4)
        .Update
     End If
  End With
rs.Close
bd.Close
End Sub

Private Sub Combo_opr_management_click()
On Error Resume Next
sql = "SELECT * "
sql = sql & "From DEF_opr_table "
sql = sql & "WHERE DEF_opr_table.line=" & Combo_opr_management.Text
With DataBD
      .RecordSource = sql
      .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
  If DataBD.Recordset.Fields("first") <> "" Then
    Combo_opr_management1.Text = DataBD.Recordset.Fields("first")
    If DataBD.Recordset.Fields("first_v") = 0 Then Option1.Value = True Else Option2.Value = True
     Else
      Combo_opr_management1.Text = ""
      Option1.Value = True
  End If
  If DataBD.Recordset.Fields("second") <> "" Then
    Combo_opr_management2.Text = DataBD.Recordset.Fields("second")
    If DataBD.Recordset.Fields("second_v") = 0 Then Option3.Value = True Else Option4.Value = True
     Else
      Combo_opr_management2.Text = ""
      Option1.Value = True
  End If
  If DataBD.Recordset.Fields("third") <> "" Then
    Combo_opr_management3.Text = DataBD.Recordset.Fields("third")
    If DataBD.Recordset.Fields("third_v") = 0 Then Option5.Value = True Else Option6.Value = True
     Else
      Combo_opr_management3.Text = ""
      Option1.Value = True
  End If
  If DataBD.Recordset.Fields("fourth") <> "" Then
    Combo_opr_management4.Text = DataBD.Recordset.Fields("fourth")
    If DataBD.Recordset.Fields("fourth_v") = 0 Then Option7.Value = True Else Option8.Value = True
```

```
     Else
      Combo_opr_management4.Text = ""
      Option1.Value = True
  End If
End If
End Sub

Private Sub combo_order_cod_Change()
  combo_order_desc.ListIndex = combo_order_cod.ListIndex
End Sub

Private Sub combo_order_cod_Click()
  combo_order_desc.ListIndex = combo_order_cod.ListIndex
End Sub

Private Sub combo_order_desc_Click()
  On Error Resume Next
  combo_order_cod.ListIndex = combo_order_desc.ListIndex
End Sub

Private Sub combo_order_desc_KeyPress(KeyAscii As Integer)
  combo_order_cod.ListIndex = combo_order_desc.ListIndex
End Sub

Private Sub combo_receipts_component_click()
  On Error Resume Next
  combo_receipts_component_desc.ListIndex = combo_receipts_component.ListIndex
  Combo_rm_cod.ListIndex = combo_receipts_component.ListIndex
  sql = "SELECT DEF_components.component,DEF_components.cod,DEF_components.desc "
  sql = sql & "From DEF_components "
  sql = sql & "WHERE DEF_components.cod=" & combo_receipts_component.ListIndex
  With DataBD
    .RecordSource = sql
    .Refresh
  End With
  txt_desc.Text = CStr(DataBD.Recordset.Fields("desc"))
End Sub

Private Sub combo_receipts_component_desc_Click()
  combo_receipts_component.ListIndex = combo_receipts_component_desc.ListIndex
End Sub

Private Sub Combo_trf_component_Click()
  Combo_trf_desc.ListIndex = Combo_trf_component.ListIndex
End Sub

Private Sub Combo_trf_desc_Click()
  Combo_trf_component.ListIndex = Combo_trf_desc.ListIndex
End Sub

Private Sub Combo_ver_orders_Click()
  Call desenha_grelha_ordens
End Sub

Private Sub Combo_ver_receipts_Click()
  Call desenha_grelha_receipts
End Sub

Private Sub Combo1_Click()
Dim linha As String
Dim maquina As String

grid.Row = 0
grid.Col = 1
grid.CellAlignment = 4
linha = Combo1.List(Combo1.ListIndex)
grid.Text = "Line" + str(linha)
For i = 1 To grid.rows - 14
  grid.Col = 2
  grid.Row = i
  maquina = grid.Text
  grid.Col = 1
  grid.CellAlignment = flexAlignCenterCenter
  If maquina = "Total" Then
    grid.Text = calcula_total(1)
    Exit For
  End If
  If maquina <> "" Then grid.Text = CStr(devolve_count_maquinas(maquina, linha))
Next i
End Sub

Function calcula_total(grelha As Integer)
total_aux = 0
For i = 1 To grid.rows
  grid.Col = 0
  grid.Row = i
  grid.Col = 2
  maquina = grid.Text
  grid.Col = 1
  If maquina <> "Total" And maquina <> "" Then
    grid.Col = 1
    total_aux = total_aux + CLng(grid.Text)
   Else
     Exit For
  End If
Next i
calcula_total = total_aux
End Function

Function devolve_linha_image(linha As String)
Dim sql As String

sql = "SELECT imagem_op "
sql = sql & "From DEF_line_table "
sql = sql & "WHERE line=" & linha & ""
With DataBD
      .RecordSource = sql
```

```
      .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
   DataBD.Recordset.MoveFirst

  If Not IsNull(DataBD.Recordset.Fields("imagem_op")) Then
   devolve_linha_image = CStr(DataBD.Recordset.Fields("imagem_op"))
   Else
  devolve_linha_image = "0"
  End If
 Else
  devolve_linha_image = "0"
End If
End Function

Function devolve_tempo_operador(linha As String)
Dim sql As String

sql = "SELECT opr_time "
sql = sql & "From DEF_line_table "
sql = sql & "WHERE line=" & linha & ""
With DataBD
      .RecordSource = sql
      .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
   DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("opr_time")) Then
   devolve_tempo_operador = CStr(DataBD.Recordset.Fields("opr_time"))
   Else
  devolve_tempo_operador = "0"
  End If
 Else
  devolve_tempo_operador = "0"
End If
End Function

Function devolve_nova_encomenda()
Dim sql As String

sql = "SELECT n_order"
sql = sql & " From CTRL_orders "
sql = sql & "ORDER BY n_order DESC"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("n_order")) Then
    devolve_nova_encomenda = CLng(DataBD.Recordset.Fields("n_order")) + 1
   Else
    devolve_nova_encomenda = 1
  End If
 Else
   devolve_nova_encomenda = 1
End If
End Function

Function devolve_nova_recepcao()
Dim sql As String

sql = "SELECT trnbr"
sql = sql & " From TRN_warehouse "
sql = sql & "ORDER BY trnbr "
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
  If Not IsNull(DataBD.Recordset.Fields("trnbr")) Then
    devolve_nova_recepcao = CLng(DataBD.Recordset.Fields("trnbr")) + 1
   Else
    devolve_nova_recepcao = 1
  End If
 Else
   devolve_nova_recepcao = 1
End If
End Function

Function devolve_movimentos(component As String, Tipo As String)
Dim sql As String

sql = "SELECT Sum(TRN_warehouse.qty) AS total "
sql = sql & "From TRN_warehouse "
sql = sql & "Where TRN_warehouse.component='" & component & "' and TRN_warehouse.type='" & Tipo & "'"
sql = sql & " GROUP BY TRN_warehouse.component"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("total")) Then
    devolve_movimentos = DataBD.Recordset.Fields("total")
   Else
     devolve_movimentos = 0
  End If
 Else
   devolve_movimentos = 0
End If
End Function
```

```
Function devolve_orders_produto(produto As String)
Dim sql As String

sql = "SELECT Sum(CTRL_orders.qty_inicial) AS total "
sql = sql & "From CTRL_orders "
sql = sql & "Where CTRL_orders.cod='" & produto & "'"
sql = sql & " GROUP BY CTRL_orders.cod"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("total")) Then
    devolve_orders_produto = DataBD.Recordset.Fields("total")
   Else
     devolve_orders_produto = 0
  End If
 Else
   devolve_orders_produto = 0
End If
End Function

Function devolve_stock_produto(produto As String)
Dim sql As String

sql = "SELECT Sum(CTRL_orders.qty_prod) AS total "
sql = sql & "From CTRL_orders "
sql = sql & "Where CTRL_orders.cod='" & produto & "'"
sql = sql & " GROUP BY CTRL_orders.cod"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("total")) Then
    devolve_stock_produto = DataBD.Recordset.Fields("total")
   Else
     devolve_stock_produto = 0
  End If
 Else
   devolve_stock_produto = 0
End If
End Function

Function devolve_movimentos_transfer(component As String)
  devolve_movimentos_transfer = devolve_movimentos(component, "Transfer")
End Function

Function devolve_stock(component As String)
  devolve_stock  =  devolve_movimentos(CStr(component),  "Reception")  -  devolve_movimentos(component,
"Transfer")
End Function

Function devolve_movimentos_Reception(component As String)
  devolve_movimentos_Reception = devolve_movimentos(CStr(component), "Reception")
End Function

Function devolve_count_maquinas(tipo_maquina As String, linha As String)
Dim sql As String

sql = "SELECT DEF_machine_line.line, DEF_machine_line.machine_cod, Count(*) AS [result] "
sql = sql & "From DEF_machine_line "
sql = sql & "WHERE DEF_machine_line.line=" & linha & " and DEF_machine_line.machine_cod=" & tipo_maquina
& ""
sql = sql & " GROUP BY DEF_machine_line.line, DEF_machine_line.machine_cod"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("result")) Then
    devolve_count_maquinas = DataBD.Recordset.Fields("result")
   Else
     devolve_count_maquinas = 0
  End If
 Else
   devolve_count_maquinas = 0
End If
End Function

Function devolve_count_maquinas_linha(linha As String)
Dim sql As String

sql = "SELECT DEF_machine_line.line, Count(*) AS [result] "
sql = sql & "From DEF_machine_line "
sql = sql & "WHERE DEF_machine_line.line=" & linha
sql = sql & " GROUP BY DEF_machine_line.line"
With Databd2
  .RecordSource = sql
  .Refresh
End With
If Not Databd2.Recordset.EOF Then
  Databd2.Recordset.MoveLast
  Databd2.Recordset.MoveFirst
  If Not IsNull(Databd2.Recordset.Fields("result")) Then
    devolve_count_maquinas_linha = Databd2.Recordset.Fields("result")
   Else
     devolve_count_maquinas_linha = 0
    End If
   Else
```

```
  devolve_count_maquinas_linha = 0
End If
Databd2.Recordset.Close
End Function

Function devolve_count_linhas() As Long
Dim sql As String
sql = "SELECT DEF_machine_line.line "
sql = sql & "From DEF_machine_line "
sql = sql & " GROUP BY DEF_machine_line.line"
With Databd2
  .RecordSource = sql
  .Refresh
End With
devolve_count_linhas = 0
Databd2.Recordset.MoveLast
Databd2.Recordset.MoveFirst
While Not Databd2.Recordset.EOF
  devolve_count_linhas = devolve_count_linhas + 1
  Databd2.Recordset.MoveNext
Wend
Databd2.Recordset.Close
End Function

Function permite_maquina_produto(maquina As String, produto As String)

sql = "SELECT DEF_machine_products.machine_cod "
sql = sql & "From DEF_machine_products "
sql = sql & " where machine_cod=" & CStr(maquina) & " and product_cod='" & produto & "'"
With Databd2
      .RecordSource = sql
      .Refresh
End With
If Databd2.Recordset.RecordCount > 0 Then permite_maquina_produto = False Else permite_maquina_produto =
True
Databd2.Recordset.Close
End Function

Function existe_maquina(maquina As String)
sql = "SELECT DEF_types_table.machine_cod "
sql = sql & "From DEF_types_table "
sql = sql & " where DEF_types_table.machine_cod=" & CStr(maquina) & ""
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then existe_maquina = True Else existe_maquina = False
End Function

Function existe_produto_bom(produto As String)
sql = "SELECT DEF_bom.bom_cod "
sql = sql & " FROM DEF_bom "
sql = sql & " where  DEF_bom.bom_cod='" & CStr(produto) & "'"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then existe_produto_bom = True Else existe_produto_bom = False
End Function

Function existe_produto(produto As String)
sql = "SELECT CTRL_orders.cod "
sql = sql & "From CTRL_orders "
sql = sql & " where CTRL_orders.cod='" & CStr(produto) & "'"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then existe_produto = True Else existe_produto = False
End Function

Function existe_maquina_linha(maquina As String)
sql = "SELECT DEF_machine_line.line "
sql = sql & "From DEF_machine_line "
sql = sql & " where DEF_machine_line.machine_cod=" & CStr(maquina) & ""
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then existe_maquina_linha = True Else existe_maquina_linha = False
End Function

Function existe_solucao()
Dim sql As String
sql = "SELECT Count(CTRL_manufacturing.number) AS total "
sql = sql & "From CTRL_manufacturing "
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If (DataBD.Recordset.Fields("total")) > 0 Then
    existe_solucao = True
    'DataBD.Recordset.Fields("total")
   Else
   existe_solucao = False
  End If
 Else
    existe_solucao = False
End If
DataBD.Recordset.Close
End Function

Sub desenha_grid_global()
```

```
Dim sql As String

sql = "SELECT DEF_machine_line.line, DEF_machine_line.machine_cod, Count(*) AS [result] "
sql = sql & "From DEF_machine_line "
sql = sql & " GROUP BY DEF_machine_line.line, DEF_machine_line.machine_cod order by
DEF_machine_line.line"
On Error Resume Next
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
  DataBD.Recordset.MoveLast
  DataBD.Recordset.MoveFirst
 Else
   Exit Sub
End If
total_aux = 0
grid_global.Col = 0
grid_global.CellAlignment = flexAlignCenterCenter
While Not DataBD.Recordset.EOF
  coluna = DataBD.Recordset.Fields("line")
  grid_global.Col = coluna
  grid_global.CellAlignment = flexAlignCenterCenter
  maquina = DataBD.Recordset.Fields("machine_cod")
  'escreve em todas as maquinas da linha o valor 0
  For j = 1 To grid_global.rows - 14
    grid_global.Row = j
    grid_global.CellAlignment = flexAlignCenterCenter
    If grid_global.Text = "" Then grid_global.Text = "0"
  Next j
  'posiciona na maquina corrrecta
  For j = 1 To grid_global.rows - 15
    grid_global.Col = 41
    grid_global.Row = j
    grid_global.CellAlignment = flexAlignCenterCenter
    If CInt(maquina) = CInt(grid_global.Text) Then Exit For
  Next j
  grid_global.Col = coluna
  'escreve o valor
  grid_global.Text = CStr(DataBD.Recordset.Fields("result"))
  'escreve o total
  grid_global.Row = grid_global.rows - 14
  grid_global.Text = CStr(CLng(grid_global.Text) + CLng(DataBD.Recordset.Fields("result")))
  DataBD.Recordset.MoveNext
Wend
End Sub

Sub desenha_global_uma(linha As Integer)
Dim sql As String

sql = "SELECT DEF_machine_line.line, DEF_machine_line.machine_cod, Count(*) AS [result] "
sql = sql & "From DEF_machine_line where line=" & CStr(linha)
sql = sql & " GROUP BY DEF_machine_line.line, DEF_machine_line.machine_cod order by
DEF_machine_line.line"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
  DataBD.Recordset.MoveLast
  DataBD.Recordset.MoveFirst
 Else
   Exit Sub
End If
total_aux = 0
grid_global.Col = 0
grid_global.CellAlignment = flexAlignCenterCenter
coluna = linha
grid_global.Col = coluna
grid_global.Row = grid_global.rows - 14
grid_global.Text = "0"
While Not DataBD.Recordset.EOF
  coluna = DataBD.Recordset.Fields("line")
  grid_global.Col = coluna
  grid_global.CellAlignment = flexAlignCenterCenter
  maquina = DataBD.Recordset.Fields("machine_cod")
  'escreve em todas as maquinas da linha o valor 0
  For j = 1 To grid_global.rows - 14
    grid_global.Row = j
    grid_global.CellAlignment = flexAlignCenterCenter
    If grid_global.Text = "" Then grid_global.Text = "0"
  Next j
  'posiciona na maquina corrrecta
  For j = 1 To grid_global.rows - 15
    grid_global.Col = 41
    grid_global.Row = j
    grid_global.CellAlignment = flexAlignCenterCenter
    If CInt(maquina) = CInt(grid_global.Text) Then Exit For
  Next j
  grid_global.Col = coluna
  'escreve o valor
  grid_global.Text = CStr(DataBD.Recordset.Fields("result"))
  'escreve o total
  grid_global.Row = grid_global.rows - 14
  grid_global.Text = CStr(CLng(grid_global.Text) + CLng(DataBD.Recordset.Fields("result")))
  DataBD.Recordset.MoveNext
Wend
End Sub

Private Sub Command_model_export_Click()
Dim i As Long
Dim NumMaquinas, tmp As Long
Dim txt, txt2, txt3, Sinaltxt, sql, cod_maquina As String
Dim arena_mod As Arena.Module
Dim indice(), tipos()
```

```
Sinal = 0
If Modelo_Criado = 0 Then
  Call Create_TAF_Model
 Else
   Call Create_TAF_Model ' cria outro modelo
End If

Call Entity_Pictures_create

num_linhas = devolve_count_linhas
indice = devolve_linhas_usadas
For i = 1 To num_linhas
  linha = indice(i)

  x_opr_image = devolve_cod_imagem_operador_linha(str(linha))

  tipos = devolve_tipo_maquinas_linha(CStr(linha))
  num_maquinas = devolve_count_maquinas_linha(CStr(linha))
  tmp = linha    'para colocacao no ecran
  txt = Trim(CStr(linha))

  Call Create_Variables("Linha " + Trim(CStr(linha)), CStr(num_maquinas))
  Set arena_mod = M.Modules.Create("AdvancedProcess", "Advanced Set", 100, 100)
  arena_mod.Data("Name") = "Destino" + Trim(CStr(linha))
  arena_mod.Data("Type") = "Other"

  Call Create_Variables("NumOp_" + txt, "1")
  Call Create_Variables("ImagemOp_" + txt, Trim(CStr(x_opr_image)))
  destino = escolhe_primeira_maquina(CStr(txt))

 ' o tempo de rota  para a 1a maquina
 ' Call Create_Variables("TempRotaOp_" + txt, 1)

  tr = 0
  For k = 1 To destino
    tr = tr + devolve_temporota_maquina(CStr(tipos(k)))
  Next k
  Call Create_Variables("TempRotaOp_" + txt, CStr(tr))

  destino = "Maquina_" + txt + "_" + Trim(CStr(destino))
  Call Create_Operador("NumOp_" + txt, "ImagemOp_" + txt, "EstacaoOperador_" + txt, "TempRotaOp_" + txt,
CStr(destino), -31000, -31000 + tmp * 1000 - 51)

  For j = 1 To num_maquinas

    arena_mod.Data("Other(" + Trim(CStr(j)) + ")") = "Maquina_" + txt + "_" + Trim(CStr(j))

    cod_maquina = CStr(tipos(j))
    With DataBD
      .RecordSource = "Select DEF_types_table.*  from DEF_types_table where machine_cod=" & cod_maquina
      .Refresh
      x_Route_time = .Recordset("Route_time")
      x_Proc_time = .Recordset("Proc_time")
      x_Setup_time = .Recordset("Setup_time")
      x_Work_time = .Recordset("Work_time")
      x_Fail_time = .Recordset("Fail_time")
      x_qtd_batch = .Recordset("qtd_batch")
      x_maq_image = .Recordset("machine_image")
    End With
    DataBD.Recordset.Close
    txt2 = Trim(CStr(j))
    txt3 = Trim(CStr(j + 1))
    Call Create_Variables("TempRotaMaq_" + txt + "_" + txt2, CStr(x_Route_time))
    Call Create_Variables("TempoProc_" + txt + "_" + txt2, CStr(x_Proc_time))
    Call Create_Variables("Quantidade_" + txt + "_" + txt2, CStr(x_qtd_batch))
    Call Create_Variables("Imagem_" + txt + "_" + txt2, Trim(CStr(x_opr_image)))
    Call Create_Expression("TempoFunc_" + txt + "_" + txt2, CStr(x_Work_time))
    Call Create_Expression("TempoAvaria_" + txt + "_" + txt2, CStr(x_Fail_time))
    Call Create_Variables("TempoSetup_" + txt + "_" + txt2, CStr(x_Setup_time))
    Call Create_Variables("TipoMaquina_" + txt + "_" + txt2, CStr(cod_maquina))
    Call Create_Variables("TipoProduto_" + txt + "_" + txt2, "0")
    Call Create_Variables("extra_" + txt + "_" + txt2, "0")
    Call Create_Variables("Produzidas_" + txt + "_" + txt2, "0")

    Call Create_Variables("Imagemfio_" + txt + "_" + txt2, "0")     ' inicializar c/ valor
    Call Create_Variables("Imagempalete_" + txt + "_" + txt2, "0")  ' inicializar c/ valor
    Call Create_Variables("Imagemproduto_" + txt + "_" + txt2, "0") ' inicializar c/ valor

    Sinal = Sinal + 1
    Sinaltxt = CStr(Sinal)
    If j = num_maquinas Then
      Call Create_Variables("prox_" + txt + "_" + txt2, "1")
      Call Create_Maquina("Maquina_" + txt + "_" + txt2, CStr(Sinaltxt), "TempRotaMaq_" + txt + "_" +
txt2, "Destino" + txt + "(prox_" + txt + "_" + txt2 + ")", "Maquina_" + txt + "_" + txt2 + " Recurso",
"TempoProc_" + txt + "_" + txt2, "Quantidade_" + txt + "_" + txt2, "Imagem_" + txt + "_" + txt2,
"TempoSetup_" + txt + "_" + txt2, "TempoFunc_" + txt + "_" + txt2, "TempoAvaria_" + txt + "_" + txt2,
"Tipo_" + txt + "_" + txt2, -31000 + 1000 * j, -31000 + 1000 * tmp, "Imagemproduto_" + txt + "_" + txt2,
"Imagemfio_" + txt + "_" + txt2, "Imagempalete_" + txt + "_" + txt2)
      Call Resource_Pictures("Maquina_" + txt + "_" + txt2 + " Recurso", -31000 + 1000 * j, -31500 +
1000 * tmp, CLng(x_maq_image))
     Else
      Call Create_Variables("prox_" + txt + "_" + txt2, CStr(txt3))
      Call Create_Maquina("Maquina_" + txt + "_" + txt2, CStr(Sinaltxt), "TempRotaMaq_" + txt + "_" +
txt2, "Destino" + txt + "(prox_" + txt + "_" + txt2 + ")", "Maquina_" + txt + "_" + txt2 + " Recurso",
"TempoProc_" + txt + "_" + txt2, "Quantidade_" + txt + "_" + txt2, "Imagem_" + txt + "_" + txt2,
"TempoSetup_" + txt + "_" + txt2, "TempoFunc_" + txt + "_" + txt2, "TempoAvaria_" + txt + "_" + txt2,
"Tipo_" + txt + "_" + txt2, -31000 + 1000 * j, -31000 + 1000 * tmp, "Imagemproduto_" + txt + "_" + txt2,
"Imagemfio_" + txt + "_" + txt2, "Imagempalete_" + txt + "_" + txt2)
      Call Resource_Pictures("Maquina_" + txt + "_" + txt2 + " Recurso", -31000 + 1000 * j, -31500 +
1000 * tmp, CLng(x_maq_image))
    End If
    If j = num_maquinas Then
       Call Connect_Stations_TAF(CLng(num_maquinas), CStr(linha))
    End If
  Next j
```

```
Next i
Form_Main.Show
MsgBox "End of Model Construction"
End Sub

Sub declara_prod(cod As String, qty As Long)
 If (qty) <= 0 Then
  Exit Sub
 End If
With DataBD
  .RecordSource  =  "Select  CTRL_orders.n_order,CTRL_orders.qty_inicial,CTRL_orders.qty_prod     from
CTRL_orders where  qty_inicial>qty_prod and cod='" & RTrim(cod) & "' order by n_order"
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
  DataBD.Recordset.MoveLast
  DataBD.Recordset.MoveFirst
End If
qty_aux = qty
qty_bd_aux = 0
qty_bd_falta_aux = 0
While Not DataBD.Recordset.EOF And qty_aux > 0
  If IsNull(DataBD.Recordset.Fields("qty_prod")) Then
    qty_bd_aux = 0
    qty_bd_falta_aux = CLng(DataBD.Recordset.Fields("qty_inicial"))
  Else
   qty_bd_aux = CLng(DataBD.Recordset.Fields("qty_prod"))
   qty_bd_falta_aux = CLng(DataBD.Recordset.Fields("qty_inicial")) - qty_bd_aux
  End If
  If qty_aux <= qty_bd_falta_aux Then
    Call actualiza_qty_prod(CStr(DataBD.Recordset.Fields("n_order")), CLng(qty_aux + qty_bd_aux))
    qty_aux = 0
  Else
   Call       actualiza_qty_prod(CStr(DataBD.Recordset.Fields("n_order")),       CLng(qty_bd_aux       +
qty_bd_falta_aux))
   qty_aux = qty_aux - qty_bd_falta_aux
  End If
  DataBD.Recordset.MoveNext
Wend
DataBD.Recordset.Close
End Sub

Sub insert_maquina_prod(maquina As String, prod As String)
With DataBD
   .RecordSource = "Select  product_cod,machine_cod   from  DEF_machine_products  where   product_cod='" &
RTrim(prod) & "' and machine_cod=" & maquina
  .Refresh
End With
If DataBD.Recordset.RecordCount <= 0 Then
  DataBD.Recordset.AddNew
  DataBD.Recordset.Fields("product_cod") = RTrim(prod)
  DataBD.Recordset.Fields("machine_cod") = maquina
  DataBD.Recordset.Update
End If
DataBD.Recordset.Close
End Sub

Sub delete_maquina_prod(maquina As String, prod As String)
With DataBD
   .RecordSource = "Select  product_cod,machine_cod   from  DEF_machine_products  where   product_cod='" &
RTrim(prod) & "' and machine_cod=" & maquina
  .Refresh
End With
If DataBD.Recordset.RecordCount = 1 Then
  DataBD.Recordset.Delete
End If
DataBD.Recordset.Close
End Sub

Sub actualiza_qty_prod(n_order As String, qty As Long)
With Databd2
  .RecordSource = "Select * from CTRL_orders where n_order=" & n_order
  .Refresh
End With
If Databd2.Recordset.RecordCount = 1 Then
  Databd2.Recordset.Edit
  Databd2.Recordset.Fields("qty_prod") = CLng(qty)
  Databd2.Recordset.Update
End If
Databd2.Recordset.Close
End Sub

Sub posiciona_receipts(n_receipt As String)
With Databd3
  .RecordSource = "Select * from TRN_warehouse where trnbr=" & n_receipt
  .Refresh
End With
If Not Databd3.Recordset.NoMatch Then
  txt_trnbr.Text = Databd3.Recordset.Fields("trnbr")
  txt_receipts_qty.Text = Databd3.Recordset.Fields("qty")
  'DataBD.Recordset.Fields("type") = Tipo
  For i = 1 To combo_receipts_component.ListCount
  combo_receipts_component.ListIndex = i
  If combo_receipts_component.Text = Databd3.Recordset.Fields("component") Then Exit For
  Next i
End If
Databd3.Recordset.Close
End Sub

Sub posiciona_bom(linha As Long)
  Dim comp
  Dim qty

  bom_grid.Row = linha
  bom_grid.Col = 0
  comp = bom_grid.Text
```

```
  bom_grid.Col = 2
  qty = bom_grid.Text
  txt_component_qty.Text = CStr(qty)
  For i = 1 To Combo_bom_component_cod.ListCount
    Combo_bom_component_cod.ListIndex = i
    If Combo_bom_component_cod.Text = comp Then Exit For
  Next i
End Sub

Sub posiciona_ordem(n_ordem As String)
With DataBD
  .RecordSource = "Select * from CTRL_orders where n_order=" & n_ordem
  .Refresh
End With
If Not DataBD.Recordset.NoMatch Then
  txt_n_order.Text = DataBD.Recordset.Fields("n_order")
  txt_qty.Text = DataBD.Recordset.Fields("qty_inicial")
  For i = 0 To combo_order_cod.ListCount
  combo_order_cod.ListIndex = i
  If combo_order_cod.Text = Trim(DataBD.Recordset.Fields("cod")) Then Exit For
  Next i
End If
DataBD.Recordset.Close
End Sub

Private Sub delete_orders_Click()
On Error Resume Next
 If combo_order_cod.Text = "" Or Not IsNumeric(txt_n_order.Text) Or Not IsNumeric(txt_qty.Text) Then
   MsgBox "ERROR: Invalid Values ..."
 Exit Sub
 End If
With DataBD
  .RecordSource = "Select * from CTRL_orders where n_order=" & txt_n_order.Text
  .Refresh
End With
If Not DataBD.Recordset.NoMatch Then
  If DataBD.Recordset.Fields("qty_prod") = 0 Then
    DataBD.Recordset.Delete
    Call order_new_Click
    Else
    MsgBox "Order not Found or Prodution Qty is diferent of zero..."
  End If
End If
DataBD.Recordset.Close
Call desenha_grelha_ordens
Call desenha_grelha_produto_stock
End Sub

Private Sub delete_product_Click()
Dim produto As String
produto = prod_cod.Text
If existe_produto_bom(CStr(produto)) Then
  MsgBox "ERROR: Product Attributed to an B.O.M. ..." & Chr(13) & "             Cannot be Removed",
vbExclamation
  Exit Sub
End If
If Not existe_produto(CStr(produto)) And Not existe_produto_bom(CStr(produto)) Then
  product.Recordset.Delete
  product.Recordset.MoveFirst
  prod_cod.Text = product.Recordset.Fields("cod")
  Else
    MsgBox "ERROR: Product Attributed to an Order ..." & Chr(13) & "             Cannot be Removed",
vbExclamation
End If
End Sub

Private Sub delete_receipts_Click()
 If    combo_receipts_component.Text    =    ""    Or    Not    IsNumeric(txt_trnbr.Text)    Or    Not
IsNumeric(txt_receipts_qty.Text) Then
   MsgBox "ERROR: Invalid Values ..."
 Exit Sub
 End If
With DataBD
  .RecordSource = "Select * from TRN_warehouse where trnbr=" & txt_trnbr.Text
  .Refresh
End With
If Not DataBD.Recordset.NoMatch Then
  DataBD.Recordset.Delete
  Call new_receipts_Click
End If
DataBD.Recordset.Close
Call desenha_grelha_receipts
Call desenha_grelha_stock
End Sub

Private Sub Drive1_Change()
On Error GoTo Drive1_Error
Dir1.Path = Drive1.Drive
Dir1.Refresh
Exit Sub
Drive1_Error:
  MsgBox Err.Description
  Drive1.Drive = "C:"
End Sub

Private Sub Dir1_Change()
 File1.Path = Dir1.Path
 File1.Refresh
End Sub

Private Sub File1_Click()
If Len(Dir1.Path) = 3 Then
  Text3.Text = Dir1.Path + File1.filename
  Model_FileName = File1.Path + File1.filename
 Else
  Text3.Text = Dir1.Path + "\" + File1.filename
```

```
  Model_FileName = Dir1.Path + "\" + File1.filename
End If
End Sub

Private Sub cmd_adiciona_nova_maquina_Click()
  Data1.Recordset.AddNew
  Txt_machine_cod.SetFocus
End Sub

Private Sub cmd_delete_maquina_Click()
'this may produce an error if you delete the last
'record or the only record in the recordset
Dim maquina As String
maquina = Txt_machine_cod.Text
If Not existe_maquina_linha(CStr(maquina)) Then
  Data1.Recordset.Delete
  Data1.Recordset.MoveNext
  Call inicializacao
  If Combo1.ListCount > 0 Then
    Combo1.ListIndex = 0
    Call Combo1_Click
  End If
 Else
    MsgBox "ERROR: Machine Attributed to a Line ..." & Chr(13) & "             Cannot be Removed",
vbExclamation
End If
End Sub

Private Sub cmdRefresh_Click()
  'this is really only needed for multi user apps
  Data1.Refresh
End Sub

Private Sub cmd_save_maquina_Click()
On Error Resume Next
If    Not    IsNumeric(txt_qtd_batch.Text)    Or    Not    IsNumeric(txt_Route_time.Text)    Or    Not
IsNumeric(txt_Proc_time.Text) Or Not IsNumeric(txt_Setup_time.Text) Or Not IsNumeric(txt_Work_time.Text)
Or Not IsNumeric(txt_Fail_time.Text) Then
  MsgBox "ERROR: Invalid Values ...", vbExclamation
  Exit Sub
End If
If CInt(txt_machine_image.Text) <= 0 Then
  MsgBox "ERROR: Image Not Selected ...", vbExclamation
  Exit Sub
End If
If Not IsNumeric(Txt_machine_cod.Text) Then
  MsgBox "ERROR: Invalid Machine Code ...", vbExclamation
  Txt_machine_cod.Text = ""
  Txt_machine_cod.SetFocus
  Exit Sub
End If
Data1.UpdateRecord
Data1.Recordset.Bookmark = Data1.Recordset.LastModified

Call inicializacao
  If Combo1.ListCount > 0 Then
    Combo1.ListIndex = 0
    Call Combo1_Click
  End If
End Sub

Private Sub cmdClose_Click()
  Unload Me
End Sub

Private Sub Data1_Error(DataErr As Integer, Response As Integer)
'This is where you would put error handling code
'If you want to ignore errors, comment out the next line
'If you want to trap them, add code here to handle them
MsgBox "Data error event hit err:" & Error$(DataErr)
Response = 0  'throw away the error
End Sub

Private Sub Data1_Reposition()
Screen.MousePointer = vbDefault
On Error Resume Next
'This will display the current record position
'for dynasets and snapshots
Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
'for the table object you must set the index property when
'the recordset gets created and use the following line
'Data1.Caption = "Record: " & (Data1.Recordset.RecordCount * (Data1.Recordset.PercentPosition * 0.01)) +
1
End Sub

Private Sub Data1_Validate(Action As Integer, Save As Integer)
'This is where you put validation code
'This event gets called when the following actions occur
Select Case Action
  Case vbDataActionMoveFirst
  Case vbDataActionMovePrevious
  Case vbDataActionMoveNext
  Case vbDataActionMoveLast
  Case vbDataActionAddNew
  Case vbDataActionUpdate
  Case vbDataActionDelete
  Case vbDataActionFind
  Case vbDataActionBookmark
  Case vbDataActionClose
End Select
Screen.MousePointer = vbHourglass
End Sub

Sub lista_produtos_maquina(cod_maquina As String)
With DataBD
  .RecordSource = "Select cod,desc,time,profit from DEF_product_table "
  .Refresh
```

```
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
lista_produtos_sim(0).Clear
lista_produtos_sim(1).Clear
lista_produtos_sim(0).Refresh
lista_produtos_sim(1).Refresh
lista_produtos_nao(0).Clear
lista_produtos_nao(1).Clear
lista_produtos_nao(0).Refresh
lista_produtos_nao(1).Refresh
While Not DataBD.Recordset.EOF
 If permite_maquina_produto(cod_maquina, CStr(DataBD.Recordset.Fields("cod"))) Then
  lista_produtos_sim(0).AddItem DataBD.Recordset.Fields("cod")
  lista_produtos_sim(1).AddItem DataBD.Recordset.Fields("desc")
 Else
  lista_produtos_nao(0).AddItem DataBD.Recordset.Fields("cod")
  lista_produtos_nao(1).AddItem DataBD.Recordset.Fields("desc")
 End If
  DataBD.Recordset.MoveNext
Wend
End Sub

Sub desenha_grelha_BOM(produto As String)
Dim i, a As Integer
Dim sql As String
On Error Resume Next

bom_grid.Clear
bom_grid.Refresh
Combo_bom_component_cod.ListIndex = 0
txt_component_qty.Text = ""
sql = "SELECT DEF_bom.bom_cod,DEF_components.desc,  DEF_bom.component, DEF_bom.qty "
sql = sql & "FROM DEF_bom , DEF_components where  DEF_bom.component = DEF_components.component "
sql = sql & "and DEF_bom.bom_cod ='" & RTrim(CStr(produto)) & "'"
' combo product
With Databd2
  .RecordSource = sql
  .Refresh
End With
bom_grid.Cols = 4
Databd2.Recordset.MoveLast
Databd2.Recordset.MoveFirst
bom_grid.rows = Databd2.Recordset.RecordCount + 25
linha_bom_max.Text = Databd2.Recordset.RecordCount
bom_grid.Row = 0
bom_grid.Col = 0
bom_grid.ColWidth(0) = 1000
bom_grid.CellAlignment = flexAlignCenterCenter
bom_grid.Text = "Component"
bom_grid.Col = 1
bom_grid.ColWidth(1) = 1500
bom_grid.CellAlignment = flexAlignCenterCenter
bom_grid.Text = "Description"
bom_grid.Col = 2
bom_grid.CellAlignment = flexAlignCenterCenter
bom_grid.ColWidth(2) = 1000
bom_grid.Text = "Qty"
bom_grid.ColWidth(3) = 0
bom_grid.Row = 0
While Not Databd2.Recordset.EOF
  i = i + 1
  bom_grid.Row = i
  bom_grid.Col = 0
  bom_grid.CellAlignment = flexAlignCenterCenter
  bom_grid.Text = Databd2.Recordset.Fields("component")
  bom_grid.Col = 1
  bom_grid.CellAlignment = flexAlignCenterCenter
  bom_grid.Text = Databd2.Recordset.Fields("desc")
  bom_grid.Col = 2
  bom_grid.CellAlignment = flexAlignCenterCenter
  bom_grid.Text = Databd2.Recordset.Fields("qty")
  Databd2.Recordset.MoveNext
Wend
linha_bom_actual.Text = "1"
bom_grid.Row = 1
bom_grid.Col = 3
Call bom_grid_Click
Databd2.Recordset.Close
End Sub

Private Sub Form_Load()
'centar o formulario
Me.Left = (Screen.Width - Me.Width) / 2
Me.Top = (Screen.Height - Me.Height) / 2
Call inicializacao
If Combo1.ListCount > 0 Then Combo1.ListIndex = 0
End Sub

Private Sub go_dec_Click()
If Not IsNumeric(txt_dec_prod.Text) Then
 MsgBox "Invalid Values ..."
 Exit Sub
End If
Call declara_prod(Combo_dec_prod_cod.Text, CLng(txt_dec_prod.Text))
Call desenha_grelha_ordens
End Sub

Private Sub in_prod_Click()
Call                                                    insert_maquina_prod(CStr(Txt_machine_cod.Text),
CStr(lista_produtos_sim(0).List(lista_produtos_sim(0).ListIndex)))
lista_produtos_maquina (Txt_machine_cod.Text)
End Sub

Private Sub line_combo_aux_Change()
txt_imagem_op = devolve_linha_image(CStr(line_combo_aux))
```

```
txt_time_op = devolve_tempo_operador(CStr(line_combo_aux))
End Sub

Private Sub line_Combo_Click()
'MsgBox devolve_linha_image(CStr(line_Combo.Text))
txt_imagem_op = devolve_linha_image(CStr(line_Combo.Text))
List2.ListIndex = CInt(txt_imagem_op.Text) - 1
txt_time_op = devolve_tempo_operador(CStr(line_Combo.Text))
'line_combo_aux.Text = CInt(txt_imagem_op.Text)
End Sub

Sub inicializacao_produtos()
' combo product
With DataBD
  .RecordSource = "Select cod,desc,time,profit  from DEF_product_table"
  .Refresh
End With
combo_order_cod.Clear
combo_order_desc.Clear
Combo_trf_component.Clear
Combo_trf_desc.Clear
DataBD.Recordset.MoveFirst
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
While Not DataBD.Recordset.EOF
  combo_order_cod.AddItem DataBD.Recordset.Fields("cod")
  combo_order_desc.AddItem DataBD.Recordset.Fields("desc")
  DataBD.Recordset.MoveNext
Wend
End Sub

Sub inicializacao()
Dim i, a As Integer

grid_global.Clear
grid_global.Refresh
On Error Resume Next
' ORDENS ------------------------------------------------------------
With DataBD
  .RecordSource = "Select cod,desc,time,profit  from DEF_product_table"
  .Refresh
End With
combo_order_cod.Clear
combo_order_desc.Clear
DataBD.Recordset.MoveFirst
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
combo_order_cod.AddItem " "
combo_order_desc.AddItem " "
While Not DataBD.Recordset.EOF
  combo_order_cod.AddItem Trim(DataBD.Recordset.Fields("cod"))
  combo_order_desc.AddItem DataBD.Recordset.Fields("desc")
  Combo_dec_prod_cod.AddItem Trim(DataBD.Recordset.Fields("cod"))
  Combo_dec_prod_desc.AddItem DataBD.Recordset.Fields("desc")
  DataBD.Recordset.MoveNext
Wend
DataBD.Recordset.Close
Call desenha_grelha_ordens
Call order_new_Click
Combo_ver_orders.ListIndex = 0
List6.ListIndex = 0
' ------------------------------------FIM ORDENS ------------------

' RECEIPTS ----------------------------------------------------------
With DataBD
  .RecordSource = "Select component,desc from DEF_components"
  .Refresh
End With
combo_receipts_component.Clear
combo_receipts_component_desc.Clear
DataBD.Recordset.MoveFirst
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
combo_receipts_component.AddItem " "
combo_receipts_component_desc.AddItem " "
Combo_bom_component_desc.AddItem " "
Combo_bom_component_cod.AddItem " "
Combo_mrp_component_cod.AddItem " "
Combo_mrp_component_desc.AddItem " "
While Not DataBD.Recordset.EOF
  combo_receipts_component.AddItem DataBD.Recordset.Fields("component")
  combo_receipts_component_desc.AddItem DataBD.Recordset.Fields("desc")

  Combo_bom_component_desc.AddItem DataBD.Recordset.Fields("desc")
  Combo_bom_component_cod.AddItem DataBD.Recordset.Fields("component")

  Combo_trf_component.AddItem DataBD.Recordset.Fields("component")
  Combo_trf_desc.AddItem DataBD.Recordset.Fields("desc")

  Combo_mrp_component_desc.AddItem DataBD.Recordset.Fields("desc")
  Combo_mrp_component_cod.AddItem DataBD.Recordset.Fields("component")

  DataBD.Recordset.MoveNext
Wend
DataBD.Recordset.Close
Call receipts_new_Click
Combo_ver_receipts.ListIndex = 0
' ------------------------------------FIM RECEIPTS --------------------

'------------------------ combo machine types
With Databd3
  .RecordSource = "Select Machine_types,Machine_cod  from DEF_types_table"
  .Refresh
End With
DataBD.Recordset.Close
Combo1.Clear
```

```
Combo_machine_type.Clear
Combo_machine_cod.Clear
Databd3.Recordset.MoveLast
Databd3.Recordset.MoveFirst
grid.Cols = 3
grid_global.Cols = 42
grid_global.rows = Databd3.Recordset.RecordCount + 15
grid.rows = Databd3.Recordset.RecordCount + 15
grid.Col = 0
For i = 1 To 40
  Combo1.AddItem CStr(i)
  line_Combo.AddItem CStr(i)
  Combo_opr_management.AddItem CStr(i)
Next i
line_Combo.ListIndex = 0
i = 0
While Not Databd3.Recordset.EOF
  i = i + 1
  Combo_machine_type.AddItem Databd3.Recordset.Fields("Machine_types")
  Combo_machine_cod.AddItem Databd3.Recordset.Fields("Machine_Cod")
  grid.Row = i
  grid.Col = 0
  grid.CellAlignment = flexAlignCenterCenter
  grid.Text = Databd3.Recordset.Fields("Machine_types")
  grid.Col = 1
  grid.CellAlignment = flexAlignCenterCenter
  grid.Text = 0
  grid.Col = 2
  grid.CellAlignment = flexAlignCenterCenter
  grid.Text = Databd3.Recordset.Fields("Machine_Cod")
  grid_global.Row = i
  grid_global.Col = 0
  grid_global.CellAlignment = flexAlignCenterCenter
  grid_global.Text = Databd3.Recordset.Fields("Machine_types")
  grid_global.Col = 41
  'coluna que guarda o código da maquina (=0 )para ficar invisivel
  grid_global.ColWidth(41) = 0
  grid_global.Text = Databd3.Recordset.Fields("Machine_Cod")
  Databd3.Recordset.MoveNext
Wend
Databd3.Recordset.Close
grid.Row = grid.Row + 1
grid_global.Row = grid_global.Row + 1
grid.Col = 0
grid.CellAlignment = flexAlignCenterCenter
grid.CellFontBold = True
grid.Text = "Total"
grid_global.Col = 0
grid_global.CellAlignment = flexAlignCenterCenter
grid_global.CellFontBold = True
grid_global.Text = "Total"
grid.Col = 2
grid_global.Col = 41
grid.Text = "Total"
grid_global.Text = "Total"
grid.Col = 1
grid_global.CellAlignment = flexAlignCenterCenter
grid.Text = "0"
grid.Row = grid.Row + 2
grid_global.Row = grid_global.Row + 2
grid.Col = 0
grid_global.CellAlignment = flexAlignCenterCenter
grid_global.Col = 0
grid.Text = ""
grid_global.Text = ""
grid.Col = 1
grid.Text = ""
For j = 1 To 40
  grid_global.Row = 0
  grid_global.Col = j
  grid_global.CellAlignment = flexAlignCenterCenter
  grid_global.ColWidth(j) = 300
  grid_global.Text = CStr(j)
Next j
Call desenha_grid_global
Call desenha_grelha_stock
Call desenha_grelha_produto_stock
'Call esconde_grelhas_Click

SSTab1.Tab = 2

End Sub

Sub desenha_grelha_receipts()
Dim i, a As Integer
receipts_grid.Clear
receipts_grid.Refresh
Dim sql As String
sql  =  "Select  TRN_warehouse.component,TRN_warehouse.trnbr,TRN_warehouse.qty,TRN_warehouse.type   from
TRN_warehouse "
If Combo_ver_receipts.ListIndex = 1 Then sql = sql & " where  (TRN_warehouse.type= '" & "reception" &
"')"
If Combo_ver_receipts.ListIndex = 2 Then sql = sql & " where ( TRN_warehouse.type= '" & "transfer" &
"')"
sql = sql & " order by TRN_warehouse.trnbr"
' combo product
With Databd3
 .RecordSource = sql
 .Refresh
End With
receipts_grid.Cols = 6
receipts_grid.ColWidth(4) = 0
receipts_grid.ColWidth(5) = 0
receipts_grid.rows = Databd3.Recordset.RecordCount + 15
linha_receipts_max.Text = Databd3.Recordset.RecordCount
receipts_grid.Row = 0
```

```
receipts_grid.Col = 0
receipts_grid.CellBackColor = vbBlue
receipts_grid.CellForeColor = vbWhite
receipts_grid.CellAlignment = flexAlignCenterCenter
receipts_grid.CellFontBold = True
receipts_grid.Text = "Transaction"
receipts_grid.Col = 1
receipts_grid.CellBackColor = vbBlue
receipts_grid.CellForeColor = vbWhite
receipts_grid.CellAlignment = flexAlignCenterCenter
receipts_grid.CellFontBold = True
receipts_grid.Text = "Component"
receipts_grid.Col = 2
receipts_grid.CellBackColor = vbBlue
receipts_grid.CellForeColor = vbWhite
receipts_grid.CellAlignment = flexAlignCenterCenter
receipts_grid.CellFontBold = True
receipts_grid.Text = "Qty"
receipts_grid.Col = 3
receipts_grid.CellBackColor = vbBlue
receipts_grid.CellForeColor = vbWhite
receipts_grid.CellAlignment = flexAlignCenterCenter
receipts_grid.CellFontBold = True
receipts_grid.Text = "Type"
receipts_grid.Row = 0
total1 = 0
total2 = 0
  receipts_grid.ColWidth(0) = 1500
  receipts_grid.ColWidth(1) = 2000
  receipts_grid.ColWidth(2) = 1200
  receipts_grid.ColWidth(3) = 1200
While Not Databd3.Recordset.EOF
  i = i + 1
  receipts_grid.Row = i
  receipts_grid.Col = 0
  receipts_grid.ColWidth(0) = 1500
  receipts_grid.CellAlignment = flexAlignCenterCenter
  receipts_grid.Text = Databd3.Recordset.Fields("trnbr")
  receipts_grid.Col = 1
  receipts_grid.ColWidth(1) = 2000
  receipts_grid.CellAlignment = flexAlignCenterCenter
  receipts_grid.Text = Databd3.Recordset.Fields("component")
  receipts_grid.Col = 2
  receipts_grid.ColWidth(2) = 1200
  receipts_grid.CellAlignment = flexAlignCenterCenter
 receipts_grid.Text = Databd3.Recordset.Fields("qty")
  If IsNumeric(Databd3.Recordset.Fields("qty")) Then
  receipts_grid.CellAlignment = flexAlignCenterCenter
  total1 = total1 + Databd3.Recordset.Fields("qty")
  End If
  receipts_grid.Col = 3
  receipts_grid.ColWidth(3) = 1200
  receipts_grid.CellAlignment = flexAlignCenterCenter
  receipts_grid.Text = Databd3.Recordset.Fields("type")
  If Databd3.Recordset.Fields("type") = "Reception" Then
    'Verde
    receipts_grid.CellBackColor = vbGreen
    receipts_grid.CellForeColor = vbBlack
    'BackColorSel = vbGreen
  Else
    'Vermelho
    receipts_grid.CellBackColor = vbRed
    receipts_grid.CellForeColor = vbWhite
  End If '
  Databd3.Recordset.MoveNext
Wend
linha_receipts_actual.Text = 1
receipts_grid.Row = 1
Databd3.Recordset.Close
End Sub

Sub desenha_grelha_stock()
Dim i As Integer
Dim a As Integer
Dim component
Dim sql As String

stock_grid.Clear
stock_grid.Refresh
With Databd2
  .RecordSource = "Select distinct component,desc from DEF_components"
  .Refresh
End With
If Not Databd2.Recordset.EOF Then
  Databd2.Recordset.MoveLast
  Databd2.Recordset.MoveFirst
 Else
   Exit Sub
End If
stock_grid.rows = Databd2.Recordset.RecordCount + 15
stock_grid.Row = 0
stock_grid.Col = 0
stock_grid.CellBackColor = vbBlue
stock_grid.CellForeColor = vbWhite
stock_grid.CellFontBold = True
stock_grid.CellAlignment = flexAlignCenterCenter
stock_grid.Text = "Component"
stock_grid.Col = 1
stock_grid.CellBackColor = vbBlue
stock_grid.CellForeColor = vbWhite
stock_grid.CellFontBold = True
stock_grid.CellAlignment = flexAlignCenterCenter
stock_grid.Text = "Description"
stock_grid.Col = 2
stock_grid.CellBackColor = vbBlue
stock_grid.CellForeColor = vbWhite
```

```
stock_grid.CellFontBold = True
stock_grid.CellAlignment = flexAlignCenterCenter
stock_grid.Text = "Recepts"
stock_grid.Col = 3
stock_grid.CellBackColor = vbBlue
stock_grid.CellForeColor = vbWhite
stock_grid.CellFontBold = True
stock_grid.CellAlignment = flexAlignCenterCenter
stock_grid.Text = "Transfer"
stock_grid.Col = 4
stock_grid.CellBackColor = vbBlue
stock_grid.CellForeColor = vbWhite
stock_grid.CellFontBold = True
stock_grid.CellAlignment = flexAlignCenterCenter
stock_grid.Text = "Stock"
total1 = 0
total2 = 0
i = 0
While Not Databd2.Recordset.EOF
  i = i + 1
  stock_grid.ColWidth(0) = 1200
  stock_grid.Row = i
  stock_grid.Col = 0
  stock_grid.CellAlignment = flexAlignCenterCenter
  component = CStr(Databd2.Recordset.Fields("component"))
  stock_grid.Text = component
  stock_grid.Col = 1
  stock_grid.ColWidth(1) = 1800
  stock_grid.CellAlignment = flexAlignCenterCenter
  stock_grid.Text = Databd2.Recordset.Fields("Desc")
  stock_grid.Col = 2
  stock_grid.ColWidth(2) = 1000
  stock_grid.CellAlignment = flexAlignCenterCenter
  aux1 = devolve_movimentos_Reception(CStr(component))
  stock_grid.Text = aux1
  total1 = total1 + aux1
  stock_grid.Col = 3
  stock_grid.ColWidth(3) = 1000
  stock_grid.CellAlignment = flexAlignCenterCenter
  aux2 = devolve_movimentos_transfer(CStr(component))
  stock_grid.Text = aux2
  total2 = total2 + aux2
  stock_grid.Col = 4
  stock_grid.ColWidth(4) = 1000
  stock_grid.CellAlignment = flexAlignCenterCenter
  stock_grid.Text = aux1 - aux2
  Databd2.Recordset.MoveNext
Wend
stock_txt_1.Text = "Total:"
stock_txt_2.Text = CStr(total1)
stock_txt_3.Text = CStr(total2)
stock_txt_4.Text = CStr(total1 - total2)
Databd2.Recordset.Close
End Sub

Sub desenha_grelha_produto_stock()
Dim i As Integer
Dim a As Integer
Dim component
Dim sql As String

product_stock_grid.Clear
product_stock_grid.Refresh
With Databd2
  .RecordSource = "Select distinct cod,desc,time,profit from DEF_product_table"
  .Refresh
End With
If Not Databd2.Recordset.EOF Then
  Databd2.Recordset.MoveLast
  Databd2.Recordset.MoveFirst
 Else
   Exit Sub
End If
product_stock_grid.rows = Databd2.Recordset.RecordCount + 15
product_stock_grid.Row = 0
product_stock_grid.Col = 0
product_stock_grid.CellBackColor = vbBlue
product_stock_grid.CellForeColor = vbWhite
product_stock_grid.CellFontBold = True
product_stock_grid.CellAlignment = flexAlignCenterCenter
product_stock_grid.Text = "Product"
product_stock_grid.Col = 1
product_stock_grid.CellBackColor = vbBlue
product_stock_grid.CellForeColor = vbWhite
product_stock_grid.CellFontBold = True
product_stock_grid.CellAlignment = flexAlignCenterCenter
product_stock_grid.Text = "Description"
product_stock_grid.Col = 2
product_stock_grid.CellBackColor = vbBlue
product_stock_grid.CellForeColor = vbWhite
product_stock_grid.CellFontBold = True
product_stock_grid.CellAlignment = flexAlignCenterCenter
product_stock_grid.Text = "Order"
product_stock_grid.Col = 3
product_stock_grid.CellBackColor = vbBlue
product_stock_grid.CellForeColor = vbWhite
product_stock_grid.CellFontBold = True
product_stock_grid.CellAlignment = flexAlignCenterCenter
product_stock_grid.Text = "Done"
product_stock_grid.Col = 4
product_stock_grid.CellBackColor = vbBlue
product_stock_grid.CellForeColor = vbWhite
product_stock_grid.CellFontBold = True
product_stock_grid.CellAlignment = flexAlignCenterCenter
product_stock_grid.Text = "Dif"
total1 = 0
```

```
total2 = 0
i = 0
While Not Databd2.Recordset.EOF
  i = i + 1
  product_stock_grid.Row = i
  product_stock_grid.Col = 0
  product_stock_grid.ColWidth(0) = 900
  product_stock_grid.CellAlignment = flexAlignCenterCenter
  produto = Trim(CStr(Databd2.Recordset.Fields("cod")))
  product_stock_grid.Text = produto
  product_stock_grid.Col = 1
  product_stock_grid.ColWidth(1) = 1500
  product_stock_grid.CellAlignment = flexAlignCenterCenter
  product_stock_grid.Text = Databd2.Recordset.Fields("Desc")
  product_stock_grid.Col = 2
  product_stock_grid.ColWidth(2) = 1200
  product_stock_grid.CellAlignment = flexAlignCenterCenter
  aux1 = devolve_orders_produto(CStr(produto))
  product_stock_grid.Text = aux1
  total1 = total1 + aux1
  product_stock_grid.Col = 3
  product_stock_grid.ColWidth(3) = 1200
  product_stock_grid.CellAlignment = flexAlignCenterCenter
  aux2 = devolve_stock_produto(CStr(produto))
  product_stock_grid.Text = aux2
  total2 = total2 + aux2
  product_stock_grid.Col = 4
  product_stock_grid.ColWidth(4) = 1200
  product_stock_grid.CellAlignment = flexAlignCenterCenter
  If aux1 - aux2 > 0 Then
    'Verde
    product_stock_grid.CellBackColor = vbGreen
    product_stock_grid.CellForeColor = vbBlack
    'BackColorSel = vbGreen
  Else
    'Vermelho
    product_stock_grid.CellBackColor = vbRed
    product_stock_grid.CellForeColor = vbWhite
  End If '
  product_stock_grid.Text = aux1 - aux2
  Databd2.Recordset.MoveNext
Wend
prod_txt_1.Text = "Total:"
prod_txt_2.Text = CStr(total1)
prod_txt_3.Text = CStr(total2)
prod_txt_4.Text = CStr(total1 - total2)
Databd2.Recordset.Close
End Sub

Sub desenha_grelha_ordens()
Dim i As Integer
Dim a As Integer
Dim sql As String
```

```
orders_grid.Clear
orders_grid.Refresh
sql =  "Select  CTRL_orders.n_order,CTRL_orders.qty_inicial,CTRL_orders.qty_prod,DEF_product_table.desc
from CTRL_orders,DEF_product_table where CTRL_orders.cod=DEF_product_table.cod "
If Combo_ver_orders.ListIndex = 1 Then sql = sql & " and CTRL_orders.qty_inicial=CTRL_orders.qty_prod "
If Combo_ver_orders.ListIndex = 2 Then sql = sql & " and CTRL_orders.qty_inicial>CTRL_orders.qty_prod "
sql = sql & " order by n_order"
' ----------------------------combo product
With DataBD
  .RecordSource = sql
  .Refresh
End With
orders_grid.Cols = 6
orders_grid.rows = DataBD.Recordset.RecordCount + 15
linha_max.Text = DataBD.Recordset.RecordCount
orders_grid.Row = 0
orders_grid.Col = 0
orders_grid.CellBackColor = vbBlue
orders_grid.CellForeColor = vbWhite
orders_grid.CellFontBold = True
orders_grid.CellAlignment = flexAlignCenterCenter
orders_grid.Text = "Order Nr."
orders_grid.Col = 1
orders_grid.CellBackColor = vbBlue
orders_grid.CellForeColor = vbWhite
orders_grid.CellFontBold = True
orders_grid.CellAlignment = flexAlignCenterCenter
orders_grid.Text = "Product"
orders_grid.Col = 2
orders_grid.CellBackColor = vbBlue
orders_grid.CellForeColor = vbWhite
orders_grid.CellFontBold = True
orders_grid.CellAlignment = flexAlignCenterCenter
orders_grid.Text = "Start Qty"
orders_grid.Col = 3
orders_grid.CellBackColor = vbBlue
orders_grid.CellForeColor = vbWhite
orders_grid.CellFontBold = True
orders_grid.CellAlignment = flexAlignCenterCenter
orders_grid.Text = "Done Qty"
orders_grid.Col = 4
orders_grid.CellBackColor = vbBlue
orders_grid.CellForeColor = vbWhite
orders_grid.CellFontBold = True
orders_grid.CellAlignment = flexAlignCenterCenter
orders_grid.Text = "Dif"
orders_grid.ColWidth(5) = 0
orders_grid.Row = 0
total1 = 0
total2 = 0
 orders_grid.ColWidth(0) = 1200
 orders_grid.ColWidth(1) = 900
```

```
 orders_grid.ColWidth(2) = 1500
 orders_grid.ColWidth(3) = 1200
While Not DataBD.Recordset.EOF
  i = i + 1
  orders_grid.Row = i
  orders_grid.Col = 0
  orders_grid.ColWidth(0) = 900
  orders_grid.CellAlignment = flexAlignCenterCenter
  orders_grid.Text = DataBD.Recordset.Fields("n_order")
  orders_grid.Col = 1
  orders_grid.ColWidth(1) = 1500
  orders_grid.CellAlignment = flexAlignCenterCenter
  orders_grid.Text = DataBD.Recordset.Fields("desc")
  orders_grid.Col = 2
  orders_grid.ColWidth(2) = 1200
  orders_grid.CellAlignment = flexAlignCenterCenter
  If IsNull(DataBD.Recordset.Fields("qty_inicial")) Then
   aux1 = 0
  Else
    orders_grid.CellAlignment = flexAlignCenterCenter
    aux1 = DataBD.Recordset.Fields("qty_inicial")
    total1 = total1 + aux1
  End If
  orders_grid.Text = CStr(aux1)
  orders_grid.Col = 3
  orders_grid.ColWidth(3) = 1200
  orders_grid.CellAlignment = flexAlignCenterCenter
  If IsNull(DataBD.Recordset.Fields("qty_prod")) Then
    aux2 = 0
  Else
   orders_grid.CellAlignment = flexAlignCenterCenter
   aux2 = DataBD.Recordset.Fields("qty_prod")
   total2 = total2 + aux2
  End If
  orders_grid.Text = CStr(aux2)
  orders_grid.Col = 4
  orders_grid.ColWidth(4) = 1200
  If aux1 - aux2 > 0 Then
    'Verde
    orders_grid.CellBackColor = vbGreen
    orders_grid.CellForeColor = vbBlack
    'BackColorSel = vbGreen
  Else
    'Vermelho
    orders_grid.CellBackColor = vbRed
    orders_grid.CellForeColor = vbWhite
  End If
  orders_grid.CellAlignment = flexAlignCenterCenter
  orders_grid.Text = CStr(aux1 - aux2)
  'orders_grid.BackColorSel = QBColor(0)
  DataBD.Recordset.MoveNext
Wend
ord_txt_1.Text = "Total:"
ord_txt_2.Text = CStr(total1)
ord_txt_3.Text = CStr(total2)
ord_txt_4.Text = CStr(total1 - total2)
DataBD.Recordset.Close
End Sub

Private Sub cod_bom_Change()
Call desenha_grelha_BOM(cod_bom.Text)
End Sub

Private Sub new_model_Click()
Text3.Text = "New Model"
Model_FileName = ""
End Sub

Private Sub new_orders_Click()
txt_n_order.Text = CStr(devolve_nova_encomenda())
combo_order_desc.ListIndex = 0
txt_qty.Text = ""
combo_order_desc.SetFocus
End Sub

Private Sub new_product_Click()
product.Recordset.AddNew
prod_desc.Text = ""
prod_time.Text = ""
prod_profit.Text = ""
prod_cod.SetFocus
End Sub

Private Sub new_receipts_Click()
txt_trnbr.Text = CStr(devolve_nova_recepcao())
combo_receipts_component.ListIndex = 0
txt_receipts_qty.Text = ""
combo_receipts_component.SetFocus
txt_receipts_qty.Text = ""
'txt_desc.Text = ""
End Sub

Private Sub Prod_cod_bom_Change()
Prod_cod_bom.Text = Trim(Prod_cod_bom.Text)
Call desenha_grelha_BOM(Prod_cod_bom.Text)
End Sub

Private Sub receipts_grid_Click()
aux = receipts_grid.Row
If receipts_grid.Row > 0 And receipts_grid.Row <= CInt(linha_receipts_max.Text) Then
  receipts_grid.Row = CInt(linha_receipts_actual.Text)
  For i = 0 To 2
    receipts_grid.Col = i
    receipts_grid.CellBackColor = vbGray
    receipts_grid.CellForeColor = vbBlack
  Next
```

```
  receipts_grid.Row = CInt(aux)
  For i = 0 To 2
    receipts_grid.Col = i
    receipts_grid.CellBackColor = vbBlue
    receipts_grid.CellForeColor = vbWhite
  Next
  receipts_grid.Col = 0
  Call posiciona_receipts(receipts_grid.Text)
  linha_receipts_actual.Text = aux
End If
End Sub

Private Sub save_dec_prod_Click()
If Not IsNumeric(txt_dec_prod.Text) Then
 MsgBox "Invalid Values ..."
 Exit Sub
End If
aux1 = devolve_orders_produto(CStr(Combo_dec_prod_cod.Text))
aux2 = devolve_stock_produto(CStr(Combo_dec_prod_cod.Text))
If aux1 - aux2 < txt_dec_prod.Text Then
  MsgBox "Invalid Values ..."
  Exit Sub
End If
Call declara_prod(Combo_dec_prod_cod.Text, CLng(txt_dec_prod.Text))
Call desenha_grelha_ordens
Call desenha_grelha_produto_stock
End Sub

Private Sub save_operator_Click()
Dim rs As Recordset
Dim bd As Database
'On Error Resume Next
 If txt_imagem_op.Text = "" Then
   MsgBox "ERROR: Image Not Selected ...", vbExclamation
  Exit Sub
End If
If CInt(txt_imagem_op.Text) < 1 Then
  MsgBox "ERROR: Image Not Selected ...", vbExclamation
  Exit Sub
End If
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_line_table", dbOpenTable)
 With rs
   .Index = "PrimaryKey"
   .MoveLast
   .MoveFirst
   .Seek "=", line_Combo.Text
    If Not .NoMatch Then
      .Edit
      !imagem_op = CInt(txt_imagem_op.Text)
      !opr_time = CInt(txt_time_op.Text)
      .Update
    End If
  End With
rs.Close
bd.Close
End Sub

Private Sub save_orders_Click()
 If combo_order_cod.Text = "" Or Not IsNumeric(txt_n_order.Text) Or Not IsNumeric(txt_qty.Text) Then
 MsgBox "Invalid Values ..."
 Exit Sub
 End If
 If CLng(txt_qty.Text) <= 0 Then
 MsgBox "Inicial Qty must be greater than 0 ..."
 Exit Sub
 End If
 With DataBD
    .RecordSource = "Select * from CTRL_orders where n_order=" & txt_n_order.Text
    .Refresh
 End With
 If DataBD.Recordset.RecordCount <= 0 Then
  DataBD.Recordset.AddNew
  DataBD.Recordset.Fields("n_order") = CLng(txt_n_order.Text)
  DataBD.Recordset.Fields("cod") = combo_order_cod.Text
  DataBD.Recordset.Fields("qty_inicial") = CLng(txt_qty.Text)
  DataBD.Recordset.Fields("qty_prod") = 0
  DataBD.Recordset.Update
 Else
  DataBD.Recordset.Edit
  DataBD.Recordset.Fields("cod") = combo_order_cod.Text
  DataBD.Recordset.Fields("qty_inicial") = CLng(txt_qty.Text)
  DataBD.Recordset.Update
End If
DataBD.Recordset.Close
Call new_orders_Click
Call desenha_grelha_ordens
Call desenha_grelha_produto_stock
End Sub

Private Sub save_product_Click()
On Error Resume Next
If CInt(txt_prod_image.Text) <= 0 Then
  MsgBox "ERROR: Image Not Selected ...", vbExclamation
  Exit Sub
End If
If CInt(txt_batch_image.Text) <= 0 Then
  MsgBox "ERROR: Image Not Selected ...", vbExclamation
  Exit Sub
End If
If CInt(prod_time.Text) <= 0 Then
  MsgBox "ERROR: Must have a Process Time ...", vbExclamation
  Exit Sub
End If

 With DataBD
```

```
    .RecordSource = "Select * from DEF_product_table where DEF_product_table.cod=" & prod_cod_aux.Text
    .Refresh
 End With
If DataBD.Recordset.RecordCount < 0 Then
  DataBD.Recordset.AddNew
  DataBD.Recordset.Fields("cod") = CStr(prod_cod.Text)
  DataBD.Recordset.Fields("desc") = CStr(prod_desc.Text)
  DataBD.Recordset.Fields("wire_image") = CStr(txt_batch_image.Text)
  DataBD.Recordset.Fields("batch_image") = CLng(txt_final_image.Text)
  DataBD.Recordset.Fields("time") = CStr(prod_time.Text)
  DataBD.Recordset.Fields("profit") = CStr(prod_profit.Text)
  DataBD.Recordset.Fields("prod_image") = CStr(txt_prod_image.Text)
  prod_cod_aux.Text = CStr(prod_cod.Text)
  DataBD.Recordset.Update
 Else
  DataBD.Recordset.Edit
  DataBD.Recordset.Fields("qty") = CLng(txt_component_qty.Text)
  DataBD.Recordset.Update
End If
DataBD.Recordset.Close
End Sub

Private Sub save_receipts_Click()
Tipo = "Reception"
If    combo_receipts_component.Text    =    ""    Or    Not    IsNumeric(txt_trnbr.Text)    Or    Not
IsNumeric(txt_receipts_qty.Text) Then
MsgBox "Invalid Values ..."
Exit Sub
End If
If CLng(txt_receipts_qty.Text) <= 0 Then
  MsgBox "Qty must be greater than 0 ..."
  Exit Sub
End If
With DataBD
  .RecordSource = "Select * from TRN_warehouse where trnbr=" & txt_trnbr.Text
  .Refresh
 End With
If DataBD.Recordset.RecordCount <= 0 Then
  DataBD.Recordset.AddNew
  DataBD.Recordset.Fields("trnbr") = CLng(txt_trnbr.Text)
  DataBD.Recordset.Fields("qty") = CLng(txt_receipts_qty.Text)
  DataBD.Recordset.Fields("type") = Tipo
  DataBD.Recordset.Fields("component") = combo_receipts_component
  DataBD.Recordset.Update
 Else
  DataBD.Recordset.Edit
  DataBD.Recordset.Fields("trnbr") = CLng(txt_trnbr.Text)
  DataBD.Recordset.Fields("qty") = CLng(txt_receipts_qty.Text)
  DataBD.Recordset.Fields("type") = Tipo
  DataBD.Recordset.Fields("component") = combo_receipts_component
  DataBD.Recordset.Update
End If
DataBD.Recordset.Close
Call new_receipts_Click
Call desenha_grelha_stock
Call desenha_grelha_receipts
End Sub

Private Sub MSFlexGridEdit(MSFlexGrid As Control, Edt As Control, KeyAscii As Integer)
Select Case KeyAscii
  Case 0 To 32
    Edt = MSFlexGrid
    Edt.SelStart = 1000
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
End Select
Edt.Move MSFlexGrid.CellLeft, MSFlexGrid.CellTop, MSFlexGrid.CellWidth, MSFlexGrid.CellHeight
Edt.Visible = True
Edt.SetFocus
End Sub

Private Function Fgi3(r As Integer, c As Integer) As Integer
  Fgi3 = c + MSFlexGrid3.Cols * r
End Function

Private Sub MsFlexGrid3_GotFocus()
If txtEdit3.Visible = False Then Exit Sub
MSFlexGrid3 = txtEdit3
txtEdit3.Visible = False
End Sub

Private Sub MsFlexGrid3_LeaveCell()
If txtEdit3.Visible = False Then Exit Sub
MSFlexGrid3 = txtEdit3
txtEdit3.Visible = False
End Sub

Private Sub EditKeyCode(MSFlexGrid As Control, Edt As Control, KeyCode As Integer, Shift As Integer)
Select Case KeyCode
  Case 27
   Edt.Visible = False
   MSFlexGrid.SetFocus
  Case 13
   MSFlexGrid.SetFocus
  Case 38
   MSFlexGrid.SetFocus
   DoEvents
   If MSFlexGrid.Row > MSFlexGrid.FixedRows Then
     MSFlexGrid.Row = MSFlexGrid.Row - 1
   End If
  Case 40
   MSFlexGrid.SetFocus
   DoEvents
   If MSFlexGrid.Row < MSFlexGrid.FixedRows - 1 Then
     MSFlexGrid.Row = MSFlexGrid.Row + 1
```

```
   End If
End Select
End Sub

Private Sub order_desc_Click()
order_cod.ListIndex = Combo_machine_type.ListIndex
End Sub

Private Sub order_Change()
combo_order_desc.ListIndex = combo_order_cod.ListIndex
End Sub

Private Sub Orders_Validate(Action As Integer, Save As Integer)
combo_order_desc.ListIndex = combo_order_cod.ListIndex
End Sub

Private Sub lista_produtos_nao_Click(Index As Integer)
If Index = 1 Then
 lista_produtos_nao(0).ListIndex = lista_produtos_nao(1).ListIndex
End If
End Sub

Private Sub lista_produtos_sim_Click(Index As Integer)
If Index = 1 Then
 lista_produtos_sim(0).ListIndex = lista_produtos_sim(1).ListIndex
End If
End Sub

Private Sub receipts_new_Click()
  txt_trnbr.Text = CStr(devolve_nova_recepcao())
  combo_receipts_component.ListIndex = 0
  txt_receipts_qty.Text = ""
  combo_receipts_component.SetFocus
End Sub

Private Sub order_new_Click()
  txt_n_order.Text = CStr(devolve_nova_encomenda())
  combo_order_desc.ListIndex = 0
  txt_qty.Text = ""
  combo_order_desc.SetFocus
End Sub

Private Sub orders_grid_Click()
aux = orders_grid.Row
If orders_grid.Row > 0 And orders_grid.Row <= CInt(linha_max.Text) Then
  orders_grid.Row = CInt(linha_actual.Text)
  For i = 0 To 3
    orders_grid.Col = i
    orders_grid.CellBackColor = vbGray
    orders_grid.CellForeColor = vbBlack
  Next
  orders_grid.Row = CInt(aux)
  For i = 0 To 3
    orders_grid.Col = i
    orders_grid.CellBackColor = vbWhite
    orders_grid.CellForeColor = vbBlue
  Next
  orders_grid.Col = 0
  Call posiciona_ordem(orders_grid.Text)
  linha_actual.Text = aux
End If
End Sub

Private Sub out_prod_Click()
Call                                                delete_maquina_prod(CStr(Txt_machine_cod.Text),
CStr(lista_produtos_nao(0).List(lista_produtos_nao(0).ListIndex)))
lista_produtos_maquina (Txt_machine_cod.Text)
End Sub

Private Sub save_trf_component_Click()
If Not IsNumeric(txt_trf_qty.Text) Then
   MsgBox "Invalid Values ..."
   Exit Sub
End If
Tipo = "Transfer"
If Combo_trf_component.Text = "" Or Not IsNumeric(txt_trf_qty.Text) Then
  MsgBox "Invalid Values ..."
  Exit Sub
End If
If CLng(txt_trf_qty.Text) <= 0 Then
  MsgBox "Qty must be greater than 0 ..."
  Exit Sub
End If
transaction = CStr(devolve_nova_recepcao())
With DataBD
  .RecordSource = "Select * from TRN_warehouse where trnbr=" & transaction
  .Refresh
End With
If DataBD.Recordset.RecordCount <= 0 Then
  DataBD.Recordset.AddNew
  DataBD.Recordset.Fields("trnbr") = CLng(transaction)
  DataBD.Recordset.Fields("qty") = CLng(txt_trf_qty.Text)
  DataBD.Recordset.Fields("type") = Tipo
  DataBD.Recordset.Fields("component") = Combo_trf_component
  DataBD.Recordset.Update
 Else
  DataBD.Recordset.Edit
  DataBD.Recordset.Fields("trnbr") = CLng(transaction)
  DataBD.Recordset.Fields("qty") = CLng(txt_trf_qty.Text)
  DataBD.Recordset.Fields("type") = Tipo
  DataBD.Recordset.Fields("component") = Combo_trf_component
  DataBD.Recordset.Fields("stock") = CLng(stockk)
  DataBD.Recordset.Update
End If
DataBD.Recordset.Close
Call new_receipts_Click
```

```
Call desenha_grelha_stock
Call desenha_grelha_receipts
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
If Button.Index = 1 Then
End If
If Button.Index = 3 Then
  'Unload Me
End If
End Sub

Private Sub Toolbar1_ButtonMenuClick(ByVal ButtonMenu As MSComctlLib.ButtonMenu)
If ButtonMenu.Parent.Index = 1 And ButtonMenu.Index = 1 Then
  On Error GoTo Arena_Error2
  'Call Open_Arena
End If
If ButtonMenu.Parent.Index = 1 And ButtonMenu.Index = 2 Then
  On Error GoTo Arena_Error3
 ' a.Quit
End If
If ButtonMenu.Parent.Index = 3 And ButtonMenu.Index = 1 Then
 ' Form1.Show
End If
If ButtonMenu.Parent.Index = 4 And ButtonMenu.Index = 1 Then
  Form_About.Show
End If
If ButtonMenu.Parent.Index = 4 And ButtonMenu.Index = 2 Then
 ' Form_Author.Show
End If
Exit Sub
Arena_Error3:
  MsgBox "Problem encountered trying to retrieve Arena information." & vbCr & _
  "Be Sure that Arena is Installed in your Computer", vbOKOnly, "Error Encountered"
Exit Sub
Arena_Error2:
   MsgBox "Problem encountered trying to retrieve Arena information." & vbCr & _
  "Be sure that Arena is intalled in your computer", vbOKOnly, "Error Encountered"
End Sub

Private Sub cmdViewVariables_Click()
Dim oArena As Arena.Application
Dim omodel As Arena.Model
Dim oSIMAN As Arena.SIMAN
Dim nMaxQueues, nIndex, nSpaceCt As Long
Dim sQueueName, sReport As String
Dim lista(1000, 1)

On Error GoTo Choke
Set oArena = GetObject("", "Arena.Application")
Set omodel = oArena.ActiveModel
Set oSIMAN = omodel.SIMAN
Label1 = "Variable Name": Label2 = "# Value"
txtReportArea = "": sReport = "": Me.Refresh
lsReport = ""
sReport = ""
For i = 1 To 50
lsReport = lsReport + "."
Next i
With oSIMAN
  nMaxVar = oSIMAN.VariablesMaximum
  sReport = ""
  conta = 0
  For nIndex = 1 To nMaxVar
    lista(nIndex, 0) = .ConstructString(smSimanConstructVariables, nIndex, 1)
    lista(nIndex, 1) = .VariableArrayValue(nIndex)
  Next nIndex
  ordena lista, nMaxVar, 1
  For i = 1 To nMaxVar
   sReport = sReport + Mid$(lista(i, 0), 1, Len(lista(i, 0)))
   sReport = sReport + Mid$(lsReport, 1, 40 - Len(lista(i, 0)))
   sReport = sReport + Mid$(lista(i, 1), 1, Len(lista(i, 1))) + vbNewLine
  Next i
  txtReportArea.Text = sReport
End With
Exit Sub
Choke:
  MsgBox "Problem Encountered Trying to Retrieve Arena Information." & vbCr & _
  "Be Sure that an Arena Model is Loaded and Running.", vbOKOnly, "Error Encountered"
Exit Sub
End Sub

Private Sub cmdViewenty_Click()
Dim oArena As Arena.Application
Dim omodel As Arena.Model
Dim oSIMAN As Arena.SIMAN
Dim nMaxQueues, nIndex, nSpaceCt, nMaxmaqs, nlinhas, nMaxVar, VarMax As Long
Dim sQueueName, sReport As String
Dim lista(1000, 1)

On Error GoTo Choke
Set oArena = GetObject("", "Arena.Application")
Set omodel = oArena.ActiveModel
Set oSIMAN = omodel.SIMAN
Label1 = "Variable Name": Label2 = "# Value"
txtReportArea = "": sReport = "": Me.Refresh
lsReport = ""
sReport = ""
For i = 1 To 50
lsReport = lsReport + "."
Next i
cc = 0
With oSIMAN
  nMaxE = oSIMAN.EntitiesNumberOfActive
  sReport = ""
  For nIndex = 1 To nMaxE
```

```
    lista(nIndex, 0) = .ConstructString(smSimanConstructEntities, nIndex, 1)
    lista(nIndex, 1) = .VariableArrayValue(nIndex)
    If lista(nIndex, 0) <> "" Then cc = cc + 1
  Next nIndex
  ordena lista, cc, 1
  For i = 1 To cc
   sReport = sReport + Mid$(lista(i, 0), 1, Len(lista(i, 0)))
   sReport = sReport + Mid$(lsReport, 1, 40 - Len(lista(i, 0)))
   sReport = sReport + Mid$(lista(i, 1), 1, Len(lista(i, 1))) + vbNewLine
  Next i
  txtReportArea.Text = sReport
End With
Exit Sub
Choke:
  MsgBox "Problem Encountered Trying to Retrieve Arena Information." & vbCr & _
  "Be Sure that an Arena Model is Loaded and Running.", vbOKOnly, "Error Encountered"
Exit Sub
End Sub

Private Sub cmdViewQueues_Click()
Dim oArena As Arena.Application
Dim omodel As Arena.Model
Dim oSIMAN As Arena.SIMAN
Dim nMaxQueues, nIndex, nSpaceCt As Long
Dim sReport, sQueueName As String
Dim lista(1000, 1)

On Error GoTo Choke
Set oArena = GetObject("", "Arena.Application")
Set omodel = oArena.ActiveModel
Set oSIMAN = omodel.SIMAN
Label1 = "Queue Name": Label2 = "# in Queue": Label3 = ""
txtReportArea = "": sReport = "": Me.Refresh
lsReport = ""
sReport = ""
For i = 1 To 50
lsReport = lsReport + "."
Next i
With oSIMAN
  nMaxQueues = oSIMAN.QueuesMaximum
  For nIndex = 1 To nMaxQueues
    lista(nIndex, 0) = .ConstructString(smSimanConstructQueues, nIndex, 1)
    lista(nIndex, 1) = .QueueNumberOfEntities(nIndex)
  Next nIndex
  ordena lista, nMaxQueues, 1
  For i = 1 To nMaxQueues
   sReport = sReport + Mid$(lista(i, 0), 1, Len(lista(i, 0)))
   sReport = sReport + Mid$(lsReport, 1, 40 - Len(lista(i, 0)))
   sReport = sReport + Mid$(lista(i, 1), 1, Len(lista(i, 1))) + vbNewLine
  Next i
    txtReportArea.Text = sReport
End With
Exit Sub
Choke:
  MsgBox "Problem Encountered Trying to Retrieve Arena Information." & vbCr & _
  "Be Sure That an Arena Model is Loaded and Running.", vbOKOnly, "Error Encountered"
Exit Sub
End Sub

Private Sub cmdViewResources_Click()
Dim oArena As Arena.Application
Dim omodel As Arena.Model
Dim oSIMAN As Arena.SIMAN
Dim nMaxResources, nSpaceCt, nIndex, i As Long
Dim sResourceName, sReport As String
Dim lista(60, 4)

On Error GoTo Choke
Set oArena = GetObject("", "Arena.Application")
Set omodel = oArena.ActiveModel
Set oSIMAN = omodel.SIMAN
Label1 = "Resource Name": Label2 = "# busy": Label3 = "Capacity"
txtReportArea = "": sReport = "": Me.Refresh
sReport = "Resource Name" + Chr(9) + Chr(9) + Chr(9) + "Busy" + Chr(9) + "Capacity" + Chr(9) + "State" +
vbNewLine
lsReport = ""
sReport = ""
For i = 1 To 50
lsReport = lsReport + "."
Next i
With oSIMAN
  nMaxResources = oSIMAN.ResourcesMaximum
  For nIndex = 1 To nMaxResources
    lista(nIndex, 0) = .ConstructString(smSimanConstructResources, nIndex, 1)
    lista(nIndex, 1) = .ResourceNumberBusy(nIndex)
    lista(nIndex, 2) = .ResourceCapacity(nIndex)
    lista(nIndex, 3) = .ResourceState(nIndex)
    lista(nIndex, 4) = .ConstructString(smSimanConstructStateSets, nIndex, .ResourceState(nIndex))
  Next nIndex
  ordena lista, nMaxResources, 4
  sReport = "Resource Name" +  Chr(9) + Chr(9) + Chr(9) + "Busy" + Chr(9) + "cap" + Chr(9) + "state" +
Chr(9) + "situation" + vbNewLine
  For i = 1 To nMaxResources
   sReport = sReport + CStr(lista(i, 0)) + Chr(9)
   sReport = sReport + Chr(9) + CStr(lista(i, 1))
   sReport = sReport + Chr(9) + CStr(lista(i, 2))
   sReport = sReport + Chr(9) + CStr(lista(i, 3))
   sReport = sReport + Chr(9) + CStr(lista(i, 4)) + vbNewLine
  Next i
  txtReportArea.Text = sReport
End With
Exit Sub
Choke:
  MsgBox "Problem Encountered Trying to Retrieve Arena Information." & vbCr & _
  "Be Sure That an Arena Model is Loaded and Running.", vbOKOnly, "Error Encountered"
Exit Sub
```

```
End Sub

Sub cmdViewexpressions_Click()
Dim oArena As Arena.Application
Dim omodel As Arena.Model
Dim oSIMAN As Arena.SIMAN
Dim nMaxResources As Long
Dim nIndex, i As Long
Dim sResourceName As String
Dim nSpaceCt As Long
Dim sReport As String
Dim lista(60, 1)

On Error GoTo Choke
Set oArena = GetObject("", "Arena.Application")
Set omodel = oArena.ActiveModel
Set oSIMAN = omodel.SIMAN
Label1 = "Expressions Name": Label2 = "# busy": Label3 = "Capacity"
txtReportArea = "": sReport = "": Me.Refresh
sReport = "Expressions Name" + Chr(9) + Chr(9) + Chr(9) + "Busy" + Chr(9) + "Expression" + vbNewLine
lsReport = ""
sReport = ""
For i = 1 To 50
lsReport = lsReport + "."
Next i
With oSIMAN
  nMaxexpress = oSIMAN.ExpressionsMaximum
  For nIndex = 1 To nMaxexpress
    lista(nIndex, 0) = .ConstructString(smSimanConstructExpressions, nIndex, 1)
    lista(nIndex, 1) = .ExpressionValue(nIndex)
  Next nIndex
  ordena lista, nMaxexpress, 1
  For i = 1 To nMaxexpress
   sReport = sReport + Mid$(lista(i, 0), 1, Len(lista(i, 0)))
   sReport = sReport + Mid$(lsReport, 1, 40 - Len(lista(i, 0)))
   sReport = sReport + Mid$(lista(i, 1), 1, Len(lista(i, 1))) + vbNewLine
  Next i
  txtReportArea.Text = sReport
End With
Exit Sub
Choke:
  MsgBox "Problem Encountered Trying to Retrieve Arena Information." & vbCr & _
  "Be Sure That an Arena Model is Loaded and Running.", vbOKOnly, "Error Encountered"
Exit Sub
End Sub

Private Function FillText(nValue As Long, nChars As Integer) As String
Dim nSize As Integer
Dim sTemp As String
Dim nIndex As Integer
Dim semaforo As Integer
' Return a string with a number <nValue> filled into <nChars> length, right-justified
FillText = ""
sTemp = nValue
nSize = Len(sTemp)
For nIndex = 1 To nChars - nSize
  FillText = FillText & " "
Next nIndex
FillText = FillText & sTemp
End Function

Private Sub sstab1_Click(PreviousTab As Integer)
   actualTab = SSTab1.Tab
   Select Case actualTab
     Case 3
       Call refresca_ordens_Click
     Case 7
       Call refresca_warehouse_Click
    Case 3
       Call refresca_man_management_Click
       Combo_opr_management.Text = devolve_primeira_linha_opr
       Call Combo_opr_management_click
      Case 4
        Call desenha_grelhas_limites
        DBGrid_limites.Refresh
        DBGrid_produtos.Refresh
        DBGrid_limites.Refresh
        DBGrid_limites.Visible = True
        DBGrid_produtos.Visible = True
        DBGrid_componentes.Visible = True
        grid_mach_refresh.Visible = True
        grid_prod_refresh.Visible = True
        grid_comp_refresh.Visible = True
        Frame(44).Visible = True
        Frame(45).Visible = True
        Frame(46).Visible = True
      Case 9
        res_grafico_Index = 0
        res_grafico_Index1 = 0
        Call Opt_tipo_grf_Click(0)
      End Select
     If actualTab <> 4 Then Call esconde_grelhas_Click
End Sub

Private Sub txt_batch_image_Change()
List5.ListIndex = CInt(txt_batch_image.Text) - 1
End Sub

Private Sub txt_final_image_Change()
List3.ListIndex = CInt(txt_final_image.Text) - 1
End Sub

Private Sub txt_imagem_op_Change()
List2.ListIndex = CInt(txt_imagem_op.Text) - 1
End Sub
```

```
Private Sub Txt_machine_cod_Change()
lista_produtos_maquina (Txt_machine_cod.Text)
End Sub

Private Sub txt_machine_image_Change()
List4.ListIndex = CInt(txt_machine_image.Text) - 1
End Sub

Private Sub txt_opr_combo_Change()
txt_opr_combo = devolve_linha_opr(CStr(Combo_opr_management.Text))
End Sub

Private Sub txt_prod_image_Change()
List1.ListIndex = CInt(txt_prod_image.Text) - 1
End Sub

Function devolve_linha_opr(linha As String)
Dim sql As String

sql = "SELECT line "
sql = sql & "From DEF_opr_table "
sql = sql & "WHERE line=" & linha & ""
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
   DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("line")) Then
   devolve_linha_opr = CStr(DataBD.Recordset.Fields("line"))
   Else
  devolve_linha_opr = "0"
  End If
 Else
  devolve_linha_opr = "0"
End If
End Function

Function devolve_primeira_linha_opr()
Dim sql As String
sql = "SELECT *"
sql = sql & "From DEF_opr_table "
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveFirst
devolve_primeira_linha_opr = "0"
enc = 0
While DataBD.Recordset.RecordCount > 0 And enc = 0
    If Not IsNull(DataBD.Recordset.Fields("first")) Then
        devolve_primeira_linha_opr = CStr(DataBD.Recordset.Fields("line"))
        enc = 1
    End If
    DataBD.Recordset.MoveNext
Wend
End Function

Function ordena(ParamArray lista())
Dim aux, pos, i, lim As Integer
lim = lista(1)
Numarray = lista(2)
While lim > 1
  pos = 0
  For i = 1 To lim - 1
    If lista(0)(i, 0) > lista(0)(i + 1, 0) Then
      For k = 0 To Numarray
        aux = lista(0)(i, k)
        lista(0)(i, k) = lista(0)(i + 1, k)
        lista(0)(i + 1, k) = aux
      Next k
      pos = i
    End If
    lim = pos
  Next i
Wend
ordena = lista
End Function

Function devolve_num_produtos()
Dim sql As String
sql = "SELECT Count(DEF_product_table.cod) AS total "
sql = sql & "From DEF_product_table "
With DataBD
      .RecordSource = sql
      .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("total")) Then
    devolve_num_produtos = DataBD.Recordset.Fields("total")
   Else
   devolve_num_produtos = 0
  End If
 Else
    devolve_num_produtos = 0
End If
DataBD.Recordset.Close
End Function

' ************************apenas produtos usados ************************
Function devolve_produtos_usados()
Dim num_produtos, produtos_nome(), profit(), i, j
num_produtos = devolve_num_produtos()
ReDim produtos_nome(num_produtos)
```

```
ReDim profit(num_produtos)
produtos_nome = devolve_tipo_produtos(CLng(num_produtos))

j = 0
For i = 1 To num_produtos
  If devolve_orders_produto(CStr(produtos_nome(i))) > 0 Then
    j = j + 1
    profit(j) = produtos_nome(i)
  End If
Next i
num_produtos = j
ReDim produtos_nome(num_produtos)
For i = 1 To num_produtos
    produtos_nome(i) = profit(i)
Next i
' ************************************************************************
devolve_produtos_usados = produtos_nome
End Function

Function devolve_tipo_maquinas(num_maq As Long)
Dim tipos()
Dim sql As String
ReDim tipos(num_maq)

sql = "SELECT DEF_machine_line.Machine_Cod "
sql = sql & "From DEF_machine_line "
With DataBD
      .RecordSource = sql
      .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_maq
  tipos(i) = CLng(DataBD.Recordset.Fields("Machine_cod"))
  DataBD.Recordset.MoveNext
Next i
devolve_tipo_maquinas = tipos
DataBD.Recordset.Close
End Function

Function devolve_limites_tempo(num_maqs As Long)
Dim tipos(), i
Dim sql As String
ReDim tipos(num_maqs * 2)

sql = "SELECT TEMP_maquinas.Max_Time,  TEMP_maquinas.Min_Time "
sql = sql & "From TEMP_maquinas "
With DataBD
      .RecordSource = sql
      .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_maqs
  tipos(i) = CLng(DataBD.Recordset.Fields("Min_Time"))
  tipos(i + num_maqs) = CLng(DataBD.Recordset.Fields("Max_Time"))
  DataBD.Recordset.MoveNext
Next i
devolve_limites_tempo = tipos
DataBD.Recordset.Close
End Function

Function devolve_limites_tempo_produtos(num_prod As Long)
Dim tipos(), i
Dim sql As String
ReDim tipos(num_prod * 2)

sql = "SELECT TEMP_produtos.max_time,  TEMP_produtos.min_time "
sql = sql & "From TEMP_produtos "
With DataBD
      .RecordSource = sql
      .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_prod
  tipos(i) = CLng(DataBD.Recordset.Fields("min_time"))
  tipos(i + num_prod) = CLng(DataBD.Recordset.Fields("max_time"))
  DataBD.Recordset.MoveNext
Next i
devolve_limites_tempo_produtos = tipos
DataBD.Recordset.Close
End Function
Function devolve_limites_quantidades_produtos(num_prod As Long)
Dim tipos(), i
Dim sql As String
ReDim tipos(num_prod * 2)

sql = "SELECT TEMP_produtos.max_qty,  TEMP_produtos.min_qty "
sql = sql & "From TEMP_produtos "
With DataBD
      .RecordSource = sql
      .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_prod
  tipos(i) = CLng(DataBD.Recordset.Fields("min_qty"))
  tipos(i + num_prod) = CLng(DataBD.Recordset.Fields("max_qty"))
  DataBD.Recordset.MoveNext
Next i
devolve_limites_quantidades_produtos = tipos
DataBD.Recordset.Close
End Function

Function devolve_limites_materiais(num_materiais As Long)
```

```
Dim tipos(), i
Dim sql As String
ReDim tipos(num_materiais * 2)

sql = "SELECT TEMP_componentes.Max_Qty,  TEMP_componentes.Min_Qty "
sql = sql & "From TEMP_componentes "
With DataBD
      .RecordSource = sql
      .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_materiais
  tipos(i) = CLng(DataBD.Recordset.Fields("Min_Qty"))
  tipos(i + num_materiais) = CLng(DataBD.Recordset.Fields("Max_Qty"))
  DataBD.Recordset.MoveNext
Next i
devolve_limites_materiais = tipos
DataBD.Recordset.Close
End Function

Function devolve_limites_quantidades(num_maqs As Long)
Dim tipos(), i
Dim sql As String
ReDim tipos(num_maqs * 2)

sql = "SELECT TEMP_maquinas.max_qty,  TEMP_maquinas.min_qty "
sql = sql & "From TEMP_maquinas "
With DataBD
      .RecordSource = sql
      .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_maqs
  tipos(i) = CLng(DataBD.Recordset.Fields("min_qty"))
  tipos(i + num_maqs) = CLng(DataBD.Recordset.Fields("max_qty"))
  DataBD.Recordset.MoveNext
Next i
devolve_limites_quantidades = tipos
DataBD.Recordset.Close
End Function

Function nao_usada_devolve_limites_quantidades_produtos(num_prod As Long)
Dim tipos(), i
Dim sql As String
ReDim tipos(num_prod * 2)

sql = "SELECT DEF_product_table.max_qty,  DEF_product_table.min_qty "
sql = sql & "From DEF_product_table "
With DataBD
      .RecordSource = sql
      .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_prod
  tipos(i) = CLng(DataBD.Recordset.Fields("min_qty"))
  tipos(i + num_prod) = CLng(DataBD.Recordset.Fields("max_qty"))
  DataBD.Recordset.MoveNext
Next i
nao_usada_devolve_limites_quantidades_produtos = tipos
DataBD.Recordset.Close
End Function

Function devolve_tipo_maquinas_nome(Tipo As Long)
Dim tipos()
Dim sql As String
ReDim tipos(num_maq)

sql = "SELECT DEF_types_table.Machine_Cod, DEF_types_table.Machine_Types "
sql = sql & "From DEF_types_table "
sql = sql & "WHERE Machine_cod=" & Tipo
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
    DataBD.Recordset.MoveFirst
    If Not IsNull(DataBD.Recordset.Fields("Machine_Types")) Then
   devolve_tipo_maquinas_nome = CStr(DataBD.Recordset.Fields("Machine_Types"))
   Else
  devolve_tipo_maquinas_nome = ""
  End If
 Else
  devolve_tipo_maquinas_nome = ""
End If
DataBD.Recordset.Close
End Function

Function devolve_tipo_materiais(num_mat As Long)
Dim tipos()
Dim sql As String
ReDim tipos(num_mat)

sql = "SELECT DEF_components.component "
sql = sql & "From DEF_components "
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_mat
  tipos(i) = DataBD.Recordset.Fields("component")
  DataBD.Recordset.MoveNext
```

```
Next i
devolve_tipo_materiais = tipos
DataBD.Recordset.Close
End Function

Function devolve_tipo_produtos(num_prod As Long)
Dim tipos()
Dim sql As String
ReDim tipos(num_prod)

sql = "SELECT DEF_product_table.cod "
sql = sql & "From DEF_product_table "
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_prod
  tipos(i) = DataBD.Recordset.Fields("cod")
  DataBD.Recordset.MoveNext
Next i
devolve_tipo_produtos = tipos
DataBD.Recordset.Close
End Function

Function devolve_desc_produtos(num_prod As Long)
Dim tipos()
Dim sql As String
ReDim tipos(num_prod)

sql = "SELECT DEF_product_table.desc "
sql = sql & "From DEF_product_table "
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_prod
  tipos(i) = DataBD.Recordset.Fields("desc")
  DataBD.Recordset.MoveNext
Next i
devolve_desc_produtos = tipos
DataBD.Recordset.Close
End Function

Function devolve_velocidade_maquina(maq As String)
Dim sql As String

sql = "SELECT DEF_types_table.Proc_time "
sql = sql & "From DEF_types_table "
sql = sql & "WHERE Machine_cod=" & maq & ""
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
    DataBD.Recordset.MoveFirst
    If Not IsNull(DataBD.Recordset.Fields("Proc_time")) Then
   devolve_velocidade_maquina = CLng(DataBD.Recordset.Fields("Proc_time"))
   Else
  devolve_velocidade_maquina = 0
  End If
 Else
  devolve_velocidade_maquina = 0
End If
DataBD.Recordset.Close
End Function

Function devolve_temporota_maquina(maq As String)
Dim sql As String

sql = "SELECT DEF_types_table.route_time "
sql = sql & "From DEF_types_table "
sql = sql & "WHERE Machine_cod=" & maq & ""
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
    DataBD.Recordset.MoveFirst
    If Not IsNull(DataBD.Recordset.Fields("route_time")) Then
   devolve_temporota_maquina = CLng(DataBD.Recordset.Fields("route_time"))
   Else
  devolve_temporota_maquina = 0
  End If
 Else
  devolve_temporota_maquina = 0
End If
DataBD.Recordset.Close
End Function

Function devolve_qty_material(mat As String, prod As String)
Dim sql As String
sql = "SELECT DEF_bom.qty "
sql = sql & "From DEF_bom "
sql = sql & "Where DEF_bom.bom_cod='" & prod & "' and DEF_bom.component='" & mat & "'"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
  DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("qty")) Then
   devolve_qty_material = CStr(DataBD.Recordset.Fields("qty"))
```

```
   Else
  devolve_qty_material = "0"
  End If
 Else
  devolve_qty_material = "0"
End If
DataBD.Recordset.Close
End Function

Function devolve_tempo_processo(prod As String)
Dim sql As String

sql = "SELECT DEF_product_table.time "
sql = sql & "From DEF_product_table "
sql = sql & "Where DEF_product_table.cod='" & prod & "'"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
  DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("time")) Then
   devolve_tempo_processo = CLng(DataBD.Recordset.Fields("time"))
   Else
  devolve_tempo_processo = 0
  End If
 Else
  devolve_tempo_processo = 0
End If
DataBD.Recordset.Close
End Function

Function devolve_num_maquinas()
Dim sql As String

sql = "SELECT Count(DEF_machine_line.Machine_Cod) AS total "
sql = sql & "From DEF_machine_line "
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("total")) Then
    devolve_num_maquinas = DataBD.Recordset.Fields("total")
   Else
   devolve_num_maquinas = 0
  End If
 Else
    devolve_num_maquinas = 0
End If
DataBD.Recordset.Close
End Function

Function devolve_num_materiais()
Dim sql As String

sql = "SELECT Count(DEF_components.Cod) AS total "
sql = sql & "From DEF_components "
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("total")) Then
    devolve_num_materiais = DataBD.Recordset.Fields("total")
   Else
   devolve_num_materiais = 0
  End If
 Else
    devolve_num_materiais = 0
End If
DataBD.Recordset.Close
End Function

Function devolve_num_materiais_usados()
Dim aux(), compo(), produtos_nome, materiais_tipos
num_materiais = devolve_num_materiais
num_prod = devolve_num_produtos
ReDim materiais_tipos(num_materiais)
ReDim produtos_nome(num_prod)
ReDim compo(num_materiais)
produtos_nome = devolve_tipo_produtos(CLng(num_prod))
materiais_tipos = devolve_tipo_materiais(CLng(num_materiais))
For i = 1 To num_prod
For j = 1 To num_materiais
 If pertence_bom(CStr(Trim(produtos_nome(i))), CStr(materiais_tipos(j))) Then
 compo(j) = j
 End If
 Next j
Next i
num_mat = 0
For j = 1 To num_materiais
 If compo(j) <> 0 Then num_mat = num_mat + 1
Next j
devolve_num_materiais_usados = num_mat
End Function

Function devolve_tipo_materiais_usados()
Dim aux(), compo(), produtos_nome, materiais_tipos
num_materiais = devolve_num_materiais
num_prod = devolve_num_produtos
ReDim materiais_tipos(num_materiais)
```

```
ReDim produtos_nome(num_prod)
ReDim compo(num_materiais)
ReDim aux(num_materiais)
produtos_nome = devolve_tipo_produtos(CLng(num_prod))
materiais_tipos = devolve_tipo_materiais(CLng(num_materiais))
aux = materiais_tipos
For i = 1 To num_prod
For j = 1 To num_materiais
 If pertence_bom(CStr(Trim(produtos_nome(i))), CStr(materiais_tipos(j))) Then
 compo(j) = j
 End If
 Next j
Next i
num_mat = 0
For j = 1 To num_materiais
 If compo(j) <> 0 Then num_mat = num_mat + 1
Next j
ReDim materiais_tipos(num_mat)
c = 1
For j = 1 To num_materiais
 If compo(j) <> 0 Then
   materiais_tipos(c) = aux(compo(j))
   c = c + 1
   End If
Next j
devolve_tipo_materiais_usados = materiais_tipos
End Function

Function devolve_profit_produto(prod As String)
Dim sql As String
sql = "SELECT DEF_product_table.profit "
sql = sql & "From DEF_product_table "
sql = sql & "Where DEF_product_table.cod='" & prod & "'"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
   DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("profit")) Then
   devolve_profit_produto = CStr(DataBD.Recordset.Fields("profit"))
   Else
  devolve_profit_produto = "0"
  End If
 Else
  devolve_profit_produto = "0"
End If
DataBD.Recordset.Close
End Function

Private Sub txt_qtd_batch_Change()
Dim sql As String
 If txt_qtd_batch.Text = "" Then
   MsgBox "ERROR: Quantity of Batch Must be Greather Than Zero ...", vbExclamation
  Exit Sub
 End If
sql = "SELECT * "
sql = sql & "From DEF_types_table "
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveFirst
While Not DataBD.Recordset.EOF
  DataBD.Recordset.Edit
  DataBD.Recordset.Fields("qtd_batch") = CLng(txt_qtd_batch.Text) * CLng(qty_declaration.Text)
  DataBD.Recordset.Update
  DataBD.Recordset.MoveNext
Wend
DataBD.Recordset.Close
Call delete_CTRL_planned_orders
End Sub

Sub nao_usada_desenha_grelha_manufacturing()
Dim maqs
grid_man.Refresh
maqs = 0
Call delete_table_CTRL_manufacturing
grid_global.Row = grid_global.rows - 14
For j = 1 To grid_global.Cols - 1
grid_global.Col = j
maqs = maqs + Val(grid_global.Text)
Next j

grid_man.Cols = 8
grid_man.rows = maqs + 1

num_maq = 7
num_pro = maqs
largura = 800
col_ini = 1500
If num_maq >= 7 Then max_pro = 7 Else max_pro = num_maq
grid_man.Width = col_ini + (largura * max_pro) + 255

If num_pro >= 4 Then max_pro = 4 Else max_pro = num_pro
grid_man.Height = (max_pro + 2) * 240 - 150
grid_man.Row = 0
grid_man.Col = 0
grid_man.ColWidth(0) = 1500
grid_man.CellAlignment = flexAlignCenterCenter
grid_man.Text = "N. Machine"
grid_man.Col = 1
grid_man.CellAlignment = flexAlignCenterCenter
grid_man.Text = "Line"
grid_man.Col = 2
grid_man.CellAlignment = flexAlignCenterCenter
```

```
grid_man.Text = "Type"
grid_man.Col = 3
grid_man.CellAlignment = flexAlignCenterCenter
grid_man.Text = "Max Time"
grid_man.Col = 4
grid_man.CellAlignment = flexAlignCenterCenter
grid_man.Text = "Min Time"
grid_man.Col = 5
grid_man.CellAlignment = flexAlignCenterCenter
grid_man.Text = "Max Qty"
grid_man.Col = 6
grid_man.CellAlignment = flexAlignCenterCenter
grid_man.Text = "Min Qty"
grid_man.CellAlignment = flexAlignCenterCenter
num_maq = devolve_num_maquinas()
conta = 0
l = 1
nlinha = 1
For j = 1 To grid_global.Cols - 2
  For i = 1 To grid_global.rows - 15
    grid_global.Row = i
    grid_global.Col = j
    xmaq = Val(grid_global.Text)
    For k = 1 To xmaq
      grid_man.ColWidth(j) = largura
      grid_man.Row = nlinha
      grid_man.Col = 0
      grid_man.CellAlignment = flexAlignCenterCenter
      grid_man.Text = nlinha
      grid_man.Row = nlinha
      grid_man.Col = 1
      grid_man.CellAlignment = flexAlignCenterCenter
      grid_man.Text = j
      grid_man.Row = nlinha
      grid_man.Col = 2
      grid_global.Col = 0
      grid_man.CellAlignment = flexAlignCenterCenter
      grid_man.Text = grid_global.Text
      grid_man.Row = nlinha
        grid_man.Col = 3
        grid_man.CellAlignment = flexAlignCenterCenter
        grid_man.Text = man_max_time.Text
        grid_man.Col = 4
        grid_man.CellAlignment = flexAlignCenterCenter
        grid_man.Text = man_min_time.Text
        grid_man.Col = 5
        grid_man.CellAlignment = flexAlignCenterCenter
        grid_man.Text = man_max_qty.Text
        grid_man.Col = 6
        grid_man.CellAlignment = flexAlignCenterCenter
        grid_man.Text = man_min_qty.Text
      conta = conta + 1
      nlinha = nlinha + 1
      If conta = num_maq Then Exit For
    Next k
    If conta = num_maq Then Exit For
  Next i
  If conta = num_maq Then Exit For
Next j
'grid_man.Top = 360
'grid_man.Left = 5040
End Sub

Sub create_TEMP_maquinas(maqs As Integer)
With DataBD
  .RecordSource = "select TEMP_maquinas.* FROM TEMP_maquinas "
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
DataBD.Recordset.MoveFirst
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
While Not DataBD.Recordset.EOF
  DataBD.Recordset.Delete
  DataBD.Recordset.MoveNext
Wend
End If
  For i = 1 To maqs
    DataBD.Recordset.AddNew
    DataBD.Recordset.Fields("Machine") = "  "
    DataBD.Recordset.Update
  Next i
DataBD.Recordset.Close
End Sub

Sub delete_table_CTRL_manufacturing()
With DataBD
  .RecordSource = "select CTRL_manufacturing.* FROM CTRL_manufacturing "
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
DataBD.Recordset.MoveFirst
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
While Not DataBD.Recordset.EOF
  DataBD.Recordset.Delete
  DataBD.Recordset.MoveNext
Wend
End If
DataBD.Recordset.Close
End Sub

Sub delete_CTRL_planned_orders()
With DataBD
  .RecordSource = "select CTRL_planned_orders.* FROM CTRL_planned_orders "
  .Refresh
```

```
End With
If DataBD.Recordset.RecordCount > 0 Then
DataBD.Recordset.MoveFirst
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
While Not DataBD.Recordset.EOF
  DataBD.Recordset.Delete
  DataBD.Recordset.MoveNext
Wend
End If
DataBD.Recordset.Close
End Sub

Sub desenha_grelha_result()
Dim num_maqs, num_pro, produtos_nome(), maquinas_tipos(), linhasactivas()
Dim largura, col_ini, i, j, fim
grid_result.Refresh
num_maq = devolve_num_maquinas()
ReDim maquinas_tipos(num_maq)
maquinas_tipos = devolve_tipo_maquinas(CLng(num_maq))
num_linhas = devolve_count_linhas()
ReDim linhasactivas(num_linhas)
produtos_nome = devolve_produtos_usados
num_pro = UBound(produtos_nome)
fim = 0
For i = 1 To 40
  qt = devolve_count_maquinas_linha(CLng(i))
  If qt > 0 Then
    linhasactivas(fim + 1) = i
    fim = fim + 1
   End If
  If fim = num_linhas Then Exit For
Next i
num_maq = devolve_num_maquinas()
grid_result.rows = num_pro + 1
grid_result.Cols = num_maq + 1
largura = 800
col_ini = 1500
If num_maq >= 5 Then max_pro = 5 Else max_pro = num_maq
grid_result.Width = col_ini + 55 + (largura * max_pro)
If num_pro >= 3 Then max_pro = 3 Else max_pro = num_pro
grid_result.Height = (max_pro + 3) * 240 - 150
grid_result.Row = 0
grid_result.Col = 0
grid_result.CellAlignment = flexAlignCenterCenter
grid_result.Text = "Prod/N.Mach-Type"
For i = 1 To num_pro
  grid_result.Row = i
  grid_result.CellAlignment = flexAlignCenterCenter
  grid_result.Text = Trim(produtos_nome(i))
Next i
grid_result.ColWidth(0) = col_ini
grid_result.Row = 0
c = 1
For j = 1 To num_linhas
 For i = 1 To devolve_count_maquinas_linha(CLng(linhasactivas(j)))
  grid_result.ColWidth(c) = largura
  grid_result.Col = c
  grid_result.CellAlignment = flexAlignCenterCenter
  grid_result.Text = "M" & Trim(str(c)) & "-" & Trim(str(maquinas_tipos(c)))
  c = c + 1
 Next i
 Next j
For i = 1 To num_pro
For j = 1 To num_maq
 grid_result.Col = j
 grid_result.Row = i
 grid_result.CellAlignment = flexAlignCenterCenter
 grid_result.Text = "---"
Next j
Next i
End Sub

Sub desenha_grelha_materiais()
Dim num_prod, produtos_nome(), mat_tipos(), num_mat, aux(), compo()
grid_mat.Refresh
num_materiais = devolve_num_materiais()
ReDim materiais_tipos(num_materiais)
produtos_nome = devolve_produtos_usados
num_prod = UBound(produtos_nome)
materiais_tipos = devolve_tipo_materiais(CLng(num_materiais))
ReDim compo(num_materiais)
ReDim aux(num_materiais)
aux = materiais_tipos
For i = 1 To num_prod
For j = 1 To num_materiais
 If pertence_bom(CStr(Trim(produtos_nome(i))), CStr(materiais_tipos(j))) Then
 compo(j) = j
 End If
 Next j
Next i
num_mat = 0
For j = 1 To num_materiais
 If compo(j) <> 0 Then num_mat = num_mat + 1
Next j
ReDim materiais_tipos(num_mat)
c = 1
For j = 1 To num_materiais
 If compo(j) <> 0 Then
   materiais_tipos(c) = aux(compo(j))
   c = c + 1
   End If
Next j
grid_mat.Cols = num_mat + 1
grid_mat.rows = num_prod + 1
largura = 800
```

```
col_ini = 1500
If num_mat >= 5 Then max_pro = 5 Else max_pro = num_mat
grid_mat.Width = col_ini + 55 + (largura * max_pro)
If num_prod >= 3 Then max_pro = 3 Else max_pro = num_prod
grid_mat.Height = (max_pro + 3) * 240 - 150
grid_mat.Row = 0
grid_mat.Col = 0
grid_mat.CellAlignment = flexAlignCenterCenter
grid_mat.Text = "Prod / Comp"
For i = 1 To num_prod
  grid_mat.Row = i
  grid_mat.CellAlignment = flexAlignCenterCenter
  grid_mat.Text = Trim(produtos_nome(i))
Next i
grid_mat.Row = 0
grid_mat.ColWidth(0) = col_ini
For i = 1 To num_mat
  grid_mat.ColWidth(i) = largura
  grid_mat.Col = i
  grid_mat.CellAlignment = flexAlignCenterCenter
  grid_mat.Text = materiais_tipos(i)
Next i
For i = 1 To num_prod
For j = 1 To num_mat
grid_mat.Col = j
grid_mat.Row = i
grid_mat.CellAlignment = flexAlignCenterCenter
grid_mat.Text = "---"
Next j
Next i
End Sub

Sub nao_usada_desenha_grelha_componentes()
Exit Sub ' anula procedimento
Dim num_prod, produtos_nome(), mat_tipos(), num_mat, compo(), aux
'grid_comp.Refresh
num_mat = devolve_num_materiais()
num_prod = devolve_num_produtos()
ReDim materiais_tipos(num_mat)
ReDim produtos_nome(num_pro)
num_produtos = devolve_num_produtos()
num_maquinas = devolve_num_maquinas()
ReDim produtos_nome(num_produtos)
ReDim aux(num_materiais)
ReDim mat(num_produtos, num_materiais)
ReDim compo(num_mat)

produtos_nome = devolve_tipo_produtos(CLng(num_produtos))
num_materiais = devolve_num_materiais_usados
ReDim materiais_tipos(num_materiais)
materiais_tipos = devolve_tipo_materiais_usados

grid_comp.Cols = num_prod + 1
grid_comp.rows = num_materiais + 1

num_maq = num_prod
num_pro = num_materiais
largura = 800
col_ini = 1500

grid_comp.Visible = True

If num_maq > 5 Then
max_pro = 5
Else
max_pro = num_maq + 1
End If
grid_comp.Width = col_ini + (largura * (max_pro)) + 255

If num_pro > 5 Then
max_pro = 5
grid_comp.Height = (max_pro + 1) * 240
Else
max_pro = num_pro + 1
grid_comp.Height = (max_pro) * 240 + 60
End If

grid_comp.Col = 0
grid_comp.Row = 0
grid_comp.ColWidth(0) = col_ini
grid_comp.CellAlignment = flexAlignCenterCenter
grid_comp.Text = "Comp / Prod"
For i = 1 To num_materiais
  grid_comp.Row = i
  grid_comp.CellAlignment = flexAlignCenterCenter
  grid_comp.Text = materiais_tipos(i)
Next i
grid_comp.Row = 0
For i = 1 To num_prod
  grid_comp.ColWidth(i) = largura
  grid_comp.Col = i
  grid_comp.CellAlignment = flexAlignCenterCenter
  grid_comp.Text = Trim(produtos_nome(i))
Next i

For i = 1 To num_prod
  For j = 1 To num_materiais
     grid_comp.Row = j
     grid_comp.Col = i
     grid_comp.CellAlignment = flexAlignCenterCenter

     grid_comp.Text = devolve_qty_material(CStr(materiais_tipos(j)), CStr(produtos_nome(i)))
     If grid_comp.Text = 0 Then grid_comp.Text = "---"
  Next j
Next i
```

```
'grid_comp.Top = 360
'grid_comp.Left = 6040

Exit Sub ' filipe

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("tabela_compo_aux", dbOpenTable)
rs.MoveFirst
For i = 0 To num_materiais
  For j = 0 To num_prod
    rs.Edit
    grid_comp.Row = i
    grid_comp.Col = j
    rs.Fields("componente") = grid_comp.Text
    rs.MoveNext
  Next j
Next i
rs.Close
End Sub

Sub desenha_grelha_maquinas()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_manufacturing", dbOpenTable)
Dim num_maquinas, num_linhas, i, j, max_pro, largura, col_ini
Dim linhasactivas(), maqlinha(), maquinas_tipos()
num_maquinas = devolve_num_maquinas()
num_linhas = devolve_count_linhas()
maquinas_tipos = devolve_tipo_maquinas(CLng(num_maquinas))
linhas = devolve_count_linhas()
ReDim linhasactivas(linhas)
ReDim maqlinha(linhas)
fim = 0
For i = 1 To 40
  qt = devolve_count_maquinas_linha(CLng(i))
  If qt > 0 Then
    linhasactivas(fim + 1) = i
    maqlinha(fim + 1) = qt
    fim = fim + 1
   End If
  If fim = linhas Then Exit For
Next i
grid_maquinas.Cols = num_maquinas + 1
grid_maquinas.rows = num_linhas + 1
largura = 800
col_ini = 1500
If num_maquinas >= 5 Then max_pro = 5 Else max_pro = num_maquinas + 1
grid_maquinas.Width = col_ini + 55 + (largura * max_pro)
If num_linhas >= 3 Then
max_pro = 3
grid_maquinas.Width = grid_maquinas.Width + 330
Else
max_pro = num_linhas
End If
grid_maquinas.Height = (max_pro + 3) * 240 - 150
largura = 800
col_ini = 1500
If num_maquinas >= 5 Then max_pro = 5 Else max_pro = num_maquinas
grid_maquinas.Width = col_ini + 55 + (largura * max_pro)
If num_linhas >= 3 Then max_pro = 3 Else max_pro = num_linhas
grid_maquinas.Height = (max_pro + 3) * 240 - 150
grid_maquinas.Col = 0
grid_maquinas.Row = 0
grid_maquinas.ColWidth(0) = col_ini
grid_maquinas.CellAlignment = flexAlignCenterCenter
grid_maquinas.Text = "Line/N.Mach-Type"
For i = 1 To num_linhas
  grid_maquinas.Row = i
  grid_maquinas.CellAlignment = flexAlignCenterCenter
  grid_maquinas.Text = linhasactivas(i)
Next i
grid_maquinas.Row = 0
For i = 1 To num_maquinas
  grid_maquinas.ColWidth(i) = largura
  grid_maquinas.Col = i
  grid_maquinas.CellAlignment = flexAlignCenterCenter
  grid_maquinas.Text = Trim(str(i)) & "-" & Trim(str(maquinas_tipos(i)))
Next i
For i = 1 To num_linhas
  For j = 1 To num_maquinas
     grid_maquinas.Row = i
     grid_maquinas.Col = j
     grid_maquinas.CellAlignment = flexAlignCenterCenter
     grid_maquinas.Text = "---"
  Next j
Next i
rs.MoveFirst
While Not rs.EOF
  With rs
    rs.Edit
    i = CLng(rs.Fields("line_prod"))
    j = CLng(rs.Fields("number"))
    grid_maquinas.Row = i
    grid_maquinas.Col = j
    grid_maquinas.CellAlignment = flexAlignCenterCenter
    grid_maquinas.Text = CDbl(rs.Fields("qty")) + CDbl(Val(grid_maquinas.Text))
End With
rs.MoveNext
Wend
rs.Close
grid_maquinas.Visible = True
End Sub

Sub desenha_grelha_ocupacao()
Dim num_maquinas, num_linhas, i, j, max_pro, largura, col_ini
Dim linhasactivas(), maqlinha(), maquinas_tipos()
num_maquinas = devolve_num_maquinas()
```

```
num_linhas = devolve_count_linhas()
maquinas_tipos = devolve_tipo_maquinas(CLng(num_maquinas))
linhas = devolve_count_linhas()
ReDim linhasactivas(linhas)
ReDim maqlinha(linhas)
fim = 0
For i = 1 To 40
  qt = devolve_count_maquinas_linha(CLng(i))
  If qt > 0 Then
    linhasactivas(fim + 1) = i
    maqlinha(fim + 1) = qt
    fim = fim + 1
   End If
  If fim = linhas Then Exit For
Next i
grid_ocupacao.Cols = num_maquinas + 1
grid_ocupacao.rows = num_linhas + 1
largura = 800
col_ini = 1500
If num_maquinas >= 5 Then max_pro = 5 Else max_pro = num_maquinas
grid_ocupacao.Width = col_ini + 55 + (largura * max_pro)
If num_linhas >= 3 Then max_pro = 3 Else max_pro = num_linhas
grid_ocupacao.Height = (max_pro + 3) * 240 - 150
grid_ocupacao.Col = 0
grid_ocupacao.Row = 0
grid_ocupacao.ColWidth(0) = col_ini
grid_ocupacao.CellAlignment = flexAlignCenterCenter
grid_ocupacao.Text = "Line/N.Mach-Type"
For i = 1 To num_linhas
  grid_ocupacao.Row = i
  grid_ocupacao.CellAlignment = flexAlignCenterCenter
  grid_ocupacao.Text = linhasactivas(i)
Next i
grid_ocupacao.Row = 0
For i = 1 To num_maquinas
  grid_ocupacao.ColWidth(i) = largura
  grid_ocupacao.Col = i
  grid_ocupacao.CellAlignment = flexAlignCenterCenter
  grid_ocupacao.Text = Trim(str(i)) & "-" & Trim(str(maquinas_tipos(i)))
Next i
For i = 1 To num_linhas
  For j = 1 To num_maquinas
     grid_ocupacao.Row = i
     grid_ocupacao.Col = j
     grid_ocupacao.CellAlignment = flexAlignCenterCenter
     grid_ocupacao.Text = "---"
  Next j
Next i
End Sub

Function pertence_bom(produto As String, material As String)
sql = "SELECT * "
sql = sql & " FROM DEF_bom "
sql = sql & " where  DEF_bom.bom_cod='" & CStr(produto) & "' "
sql = sql & " And  DEF_bom.component='" & CStr(material) & "' "
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then pertence_bom = True Else pertence_bom = False
End Function

Sub  escreve_CTRL_planned_orders(maquina  As  String, produto  As  Long, quant  As  Long, lote  As  Long,
num_lotes As Long)
With DataBD
  .RecordSource = "Select *  from CTRL_planned_orders "
  .Refresh
End With
DataBD.Recordset.AddNew
DataBD.Recordset.Fields("machine") = maquina
DataBD.Recordset.Fields("qty") = quant
DataBD.Recordset.Fields("product") = produto
DataBD.Recordset.Fields("lot") = num_lotes
DataBD.Recordset.Fields("lot_det") = lote
DataBD.Recordset.Update
DataBD.Recordset.Close
End Sub

Sub desenha_grelha_mrp(ParamArray lista())
Dim pedidos(), lancamentos(), inv_inicial(), inv_final(), custo_p(), custo_e(), colunazero()
periodos = lista(0)
ReDim pedidos(periodos), lancamentos(periodos), inv_inicial(periodos)
ReDim inv_final(periodos), custo_p(periodos), custo_e(periodos), colunazero(periodos)
gross = lista(1)
lancamentos = lista(2)
inv_inicial = lista(3)
inv_final = lista(4)
custo_p = lista(5)
custo_e = lista(6)
metodo = lista(7)
custo_encomenda = Val(mrp_custo_encomenda.Text)
custo_posse = Val(mrp_custo_posse.Text)
custo_total = 0
largura = 800
col_ini = 2000
grid_mrp.Visible = True
num_maquinas = periodos
grid_mrp.Height = 9 * 240 - 150
If num_maquinas >= 14 Then max_pro = 14 Else max_pro = num_maquinas
grid_mrp.Width = col_ini + (largura * max_pro)
max_periodos = max_pro
max_periodos = max_periodos
For i = 1 To CLng(mrp_days.Text)
colunazero(i) = "Day" + str(i)
Next i
fim = CLng(mrp_weeks.Text) + CLng(mrp_days.Text)
```

```
inicio = CLng(mrp_days.Text)
For i = 1 + inicio To fim
colunazero(i) = "Week" + str(i - inicio)
Next i
fim = CLng(mrp_months.Text) + CLng(mrp_days.Text) + CLng(mrp_weeks.Text)
inicio = CLng(mrp_days.Text) + CLng(mrp_weeks.Text)
For i = 1 + inicio To fim
  colunazero(i) = "Month" + str(i - inicio)
Next i
largura_f = largura * max_periodos + 2000 + 255
larg_grid = Form_Main.Width - (largura_f + 2255 / 2)
esq = larg_grid / 2
grid_mrp.Cols = periodos + 2
grid_mrp.rows = 7
grid_mrp.Left = esq
grid_mrp.Row = 0
grid_mrp.Col = 0
grid_mrp.CellAlignment = flexAlignCenterCenter
grid_mrp.Text = "Period"
For i = 1 To periodos
  grid_mrp.Col = i
  grid_mrp.CellFontBold = True
  grid_mrp.CellAlignment = flexAlignCenterCenter
  grid_mrp.Text = colunazero(i)
Next i
grid_mrp.Col = 0
grid_mrp.ColWidth(0) = col_ini
grid_mrp.CellFontBold = True
grid_mrp.Row = 1
grid_mrp.CellFontBold = True
grid_mrp.CellAlignment = flexAlignCenterCenter
grid_mrp.Text = "Gross Requeriments"
grid_mrp.Row = 2
grid_mrp.CellFontBold = True
grid_mrp.CellAlignment = flexAlignCenterCenter
grid_mrp.Text = "Order Quantity"
grid_mrp.Row = 3
grid_mrp.CellFontBold = True
grid_mrp.CellAlignment = flexAlignCenterCenter
grid_mrp.Text = "Beginning Inventory"
grid_mrp.Row = 4
grid_mrp.CellFontBold = True
grid_mrp.CellAlignment = flexAlignCenterCenter
grid_mrp.Text = "Ending Inventory"
grid_mrp.Row = 5
grid_mrp.CellFontBold = True
grid_mrp.CellAlignment = flexAlignCenterCenter
grid_mrp.Text = "Ordering Cost"
grid_mrp.Row = 6
grid_mrp.CellFontBold = True
grid_mrp.CellAlignment = flexAlignCenterCenter
grid_mrp.Text = "Carrying Cost"
For i = 1 To periodos
  grid_mrp.ColWidth(i) = largura
  grid_mrp.Col = i
  grid_mrp.Row = 1
  grid_mrp.CellAlignment = flexAlignCenterCenter
  grid_mrp.Text = gross(i)
  grid_mrp.Row = 2
  grid_mrp.CellAlignment = flexAlignCenterCenter
  grid_mrp.Text = lancamentos(i)
  grid_mrp.Row = 3
  grid_mrp.CellAlignment = flexAlignCenterCenter
  grid_mrp.Text = inv_inicial(i)
  grid_mrp.Row = 4
  grid_mrp.CellAlignment = flexAlignCenterCenter
  grid_mrp.Text = inv_final(i)
  grid_mrp.Row = 5
  grid_mrp.CellAlignment = flexAlignCenterCenter
  grid_mrp.Text = custo_e(i)
  grid_mrp.Row = 6
  grid_mrp.CellAlignment = flexAlignCenterCenter
  grid_mrp.Text = custo_p(i)
  custo_total = custo_p(i) + custo_e(i) + custo_total
Next
mrp_total.Width = 3600
mrp_total.Left = 800 * (periodos - 3) + 800 + esq
mrp_total.Alignment = 2
mrp_total.Text = metodo + "=" + str(custo_total)

End Sub

Private Sub calculate_mrp_Click()
Dim txt, componente As String

Call escreve_mrp_periodos
componente = Combo_mrp_component_cod.Text
If componente = "" Then
  Exit Sub
End If
If Optionmrp1.Value = True Then opcao = 1
If Optionmrp2.Value = True Then opcao = 2
If Optionmrp3.Value = True Then opcao = 3
If Optionmrp4.Value = True Then opcao = 4
If Optionmrp5.Value = True Then opcao = 5
If Optionmrp6.Value = True Then opcao = 6
If Optionmrp7.Value = True Then opcao = 7
If Optionmrp8.Value = True Then opcao = 8

If opcao >= 1 And opcao <= 8 Then
larg_frame = (Form_Main.Width - Int(Frame(15).Width * 10000 + 0.5)) / 2
largura = 800 * (CLng(mrp_days) + CLng(mrp_months) + CLng(mrp_weeks)) + 2000
larg_grid = Form_Main.Width - largura
esq = larg_grid / 2 - 240
    mrp_total.Width = 3600
    mrp_total.Left = 800 * (CLng(mrp_days) + CLng(mrp_months) + CLng(mrp_weeks)) * 3 + 800 + esq
```

```
    mrp_total.Text = "Wait a moment...."
    mrp_total.Alignment = 2
    mrp_total.Refresh
End If

Select Case opcao
  Case 1:
    Call MRP_lot_for_lot
    txt = "Lot For Lot"
  Case 2:
    Call MRP_period_order_quantity
    txt = "Period Order Quantity"
  Case 3:
    Call MRP_silver_meal
    txt = "Silver Meal"
  Case 4:
    Call MRP_wagner_within
    txt = "Wagner Within"
  Case 5:
    Call MRP_least_unit_cost
    txt = "Least Unit Cost"
  Case 6:
    Call MRP_part_period_balancing
    txt = "Part Period Balancing"
  Case 7:
    Call MRP_incremental_ppb
    txt = "Incremental Ppb"
  Case 8:
    Call MRP_economic_order_quantity
    txt = "Economic Order Quantity"
End Select
Call faz_grafico(CStr(txt), CStr(componente))

End Sub

Sub Form_Load_mrp()
'centar o formulario
Me.Left = (Screen.Width - Me.Width) / 2
Me.Top = (Screen.Height - Me.Height) / 2 + 500
End Sub

Sub MRP_lot_for_lot()
Dim pedidos(), lancamentos(), inv_inicial(), inv_final(), custo_p(), custo_e(), gross()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
metodo = "Lot For Lot"
periodos = devolve_num_periodos()
ReDim pedidos(periodos), lancamentos(periodos), inv_inicial(periodos)
ReDim inv_final(periodos), custo_p(periodos), custo_e(periodos), gross(periodos)
rs.MoveLast
rs.MoveFirst
While Not rs.EOF
  rs.Edit
  If rs.Fields("component") = Combo_mrp_component_cod.Text Then
    period = rs.Fields("period")
    pedidos(period) = rs.Fields("qty")
    gross(period) = rs.Fields("qty")
    lancamentos(period) = 0
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
inv_inicial(0) = 0
inv_final(0) = 0
invent = CLng(mrp_initial_stock.Text)
For i = 1 To periodos
  If invent < pedidos(i) Then
    pedidos(i) = pedidos(i) - invent
    inv_inicial(i) = 0
    inv_final(i) = 0
    Exit For
   Else:
    inv_inicial(i) = invent
    invent = invent - pedidos(i)
    inv_final(i) = invent
    pedidos(i) = 0
  End If
Next i
For i = 1 To periodos
  custo_e(i) = 0
  If pedidos(i) > 0 Then
    lancamentos(i) = Abs(pedidos(i) - inv_inicial(i))
    inv_inicial(i) = inv_final(i - 1)
    inv_final(i) = Abs(gross(i) - lancamentos(i) - inv_inicial(i))
    custo_e(i) = Val(mrp_custo_encomenda.Text)
  End If
  custo_p(i) = inv_final(i) * Val(mrp_custo_posse.Text)
Next
Call desenha_grelha_mrp(periodos, gross, lancamentos, inv_inicial, inv_final, custo_p, custo_e, metodo)
End Sub

Sub MRP_period_order_quantity()
Dim pedidos(), lancamentos(), inv_inicial(), inv_final(), custo_p(), custo_e()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
metodo = "Period Order Quantity"
periodos = devolve_num_periodos()
ReDim pedidos(periodos), lancamentos(periodos), inv_inicial(periodos)
ReDim inv_final(periodos), custo_p(periodos), custo_e(periodos), gross(periodos)
rs.MoveLast
rs.MoveFirst
While Not rs.EOF
  rs.Edit
  If rs.Fields("component") = Combo_mrp_component_cod.Text Then
    period = rs.Fields("period")
```

```
    pedidos(period) = rs.Fields("qty")
    gross(period) = rs.Fields("qty")
    lancamentos(period) = 0
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
custo_posse = Val(mrp_custo_posse.Text)
custo_encomenda = Val(mrp_custo_encomenda.Text)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
For i = 1 To periodos
  If invent < pedidos(i) Then
    pedidos(i) = pedidos(i) - invent
    Exit For
   Else:
     invent = invent - pedidos(i)
     pedidos(i) = 0
  End If
Next i
k = 0: media = 0
For i = 1 To periodos
  media = media + pedidos(i)
  If pedidos(i) <> 0 Then k = k + 1
Next
media = media / k
eoq = Int(Sqr((2 * custo_encomenda) / (media * custo_posse)) + 0.5)

For k = 1 To periodos
  lanca = 0
  fim = k + eoq - 1

  If fim > periodos Then fim = periodos
  For j = k To fim
    lanca = pedidos(j) + lanca
  Next j
  lancamentos(k) = lanca
  k = fim
Next k
For i = 1 To periodos
  inv_final(i) = Abs(gross(i) - inv_inicial(i) - lancamentos(i))
  custo_p(i) = inv_final(i) * custo_posse
  If i <> periodos Then inv_inicial(i + 1) = inv_final(i)
  If lancamentos(i) > 0 Then
    custo_e(i) = custo_encomenda
    Else
     custo_e(i) = 0
  End If
Next
Call desenha_grelha_mrp(periodos, gross, lancamentos, inv_inicial, inv_final, custo_p, custo_e, metodo)
End Sub
```

```
Sub MRP_wagner_within()
Dim pedidos(), lancamentos(), inv_inicial(), inv_final(), custo_p(), custo_e(), valor()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
metodo = "Wagner Within"
periodos = devolve_num_periodos()
ReDim pedidos(periodos), lancamentos(periodos), inv_inicial(periodos)
ReDim inv_final(periodos), custo_p(periodos), custo_e(periodos), gross(periodos)
rs.MoveLast
rs.MoveFirst
While Not rs.EOF
  rs.Edit
  If rs.Fields("component") = Combo_mrp_component_cod.Text Then
    period = rs.Fields("period")
    pedidos(period) = rs.Fields("qty")
    gross(period) = rs.Fields("qty")
    lancamentos(period) = 0
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
custo_posse = Val(mrp_custo_posse.Text)
custo_encomenda = Val(mrp_custo_encomenda.Text)
ReDim valor(periodos, periodos)
ReDim finla(periodos), custo(periodos)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
For i = 1 To periodos
  If invent < pedidos(i) Then
    pedidos(i) = pedidos(i) - invent
    Exit For
   Else:
    invent = invent - pedidos(i)
    pedidos(i) = 0
  End If
Next i
valor(i, 1) = custo_encomenda
bxb = i
For i = bxb + 1 To periodos
  p = i - 1
  For j = 1 To i - 1
   valor(j, i) = valor(j, i - 1) + pedidos(i) * custo_posse * p
   p = p - 1
  Next j
  minimo = valor(1, j - 1)
  For k = 1 To j - 1
    If valor(k, j - 1) < minimo Then minimo = valor(k, j - 1)
  Next k
  valor(j, i) = minimo + valor(1, 1)
Next i
For i = periodos To bxb Step -1
  Minimoc = 999999
```

```
  For j = 1 To periodos
    If (valor(j, i) < Minimoc And valor(j, i) > 0) Then
      Minimoc = valor(j, i)
      l = j: c = i
    End If
  Next
lancamentos(l) = 0
KK = 0
For xm = l To i
    lancamentos(l) = pedidos(xm) + lancamentos(l)
    KK = pedidos(xm) + KK
Next
i = l
Next
inv_inicial(0) = 0
For i = 1 To periodos
  inv_final(i) = Abs(gross(i) - inv_inicial(i) - lancamentos(i))
  custo_p(i) = inv_final(i) * custo_posse
  If i <> periodos Then inv_inicial(i + 1) = inv_final(i)
  If lancamentos(i) > 0 Then
    custo_e(i) = custo_encomenda
    Else
     custo_e(i) = 0
  End If
Next
Call desenha_grelha_mrp(periodos, gross, lancamentos, inv_inicial, inv_final, custo_p, custo_e, metodo)
End Sub

Sub MRP_silver_meal()
Dim pedidos(), lancamentos(), inv_inicial(), inv_final(), custo_p(), custo_e(), valor()
Dim media()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
metodo = "Silver Meal"
periodos = devolve_num_periodos()
ReDim pedidos(periodos), lancamentos(periodos), inv_inicial(periodos)
ReDim inv_final(periodos), custo_p(periodos), custo_e(periodos), gross(periodos)
rs.MoveLast
rs.MoveFirst
While Not rs.EOF
  rs.Edit
  If rs.Fields("component") = Combo_mrp_component_cod.Text Then
    period = rs.Fields("period")
    pedidos(period) = rs.Fields("qty")
    gross(period) = rs.Fields("qty")
    lancamentos(period) = 0
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
custo_posse = Val(mrp_custo_posse.Text)
custo_encomenda = Val(mrp_custo_encomenda.Text)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
ReDim media(periodos), valor(periodos)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
For i = 1 To periodos
  If invent < pedidos(i) Then
    pedidos(i) = pedidos(i) - invent
    Exit For
   Else:
    invent = invent - pedidos(i)
    pedidos(i) = 0
  End If
Next i
valor(1) = custo_encomenda
media(1) = custo_encomenda
aux = custo_encomenda
lanca = pedidos(i)
j = i - 1
k = i
For i = k + 1 To periodos
  aux = custo_posse * pedidos(i) * (i - 1 - j) + aux
  valor(i) = aux
  media(i) = aux / (i - j)
  If media(i) > media(i - 1) Then
    aux = custo_encomenda
    valor(i) = custo_encomenda
    media(i) = custo_encomenda
    aux = custo_encomenda
    lancamentos(j + 1) = lanca
    j = i - 1
    lanca = pedidos(j + 1)
    Else
     lanca = pedidos(i) + lanca
  End If
  If lanca <> 0 Then lancamentos(j + 1) = lanca
Next
inv_i = 0
inv_inicial(0) = 0
For i = 1 To periodos
  inv_final(i) = Abs(gross(i) - inv_inicial(i) - lancamentos(i))
  custo_p(i) = inv_final(i) * custo_posse
  If i <> periodos Then inv_inicial(i + 1) = inv_final(i)
  If lancamentos(i) > 0 Then
    custo_e(i) = custo_encomenda
    Else
     custo_e(i) = 0
  End If
Next
Call desenha_grelha_mrp(periodos, gross, lancamentos, inv_inicial, inv_final, custo_p, custo_e, metodo)
End Sub

Sub MRP_least_unit_cost()
```

```
Dim pedidos(), lancamentos(), inv_inicial(), inv_final(), custo_p(), custo_e(), valor()
Dim media()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
metodo = "Least Unit Cost"
periodos = devolve_num_periodos()
ReDim pedidos(periodos), lancamentos(periodos), inv_inicial(periodos)
ReDim inv_final(periodos), custo_p(periodos), custo_e(periodos), gross(periodos)
rs.MoveLast
rs.MoveFirst
While Not rs.EOF
  rs.Edit
  If rs.Fields("component") = Combo_mrp_component_cod.Text Then
    period = rs.Fields("period")
    pedidos(period) = rs.Fields("qty")
    gross(period) = rs.Fields("qty")
    lancamentos(period) = 0
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
custo_posse = Val(mrp_custo_posse.Text)
custo_encomenda = Val(mrp_custo_encomenda.Text)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
ReDim media(periodos), valor(periodos)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
For i = 1 To periodos
  If invent < pedidos(i) Then
    pedidos(i) = pedidos(i) - invent
    Exit For
   Else:
    invent = invent - pedidos(i)
    pedidos(i) = 0
  End If
Next i
valor(1) = custo_encomenda
media(1) = custo_encomenda
aux = custo_encomenda
lanca = pedidos(i)
j = i - 1
k = i
den_pedidos = pedidos(k)
For i = k + 1 To periodos
  aux = custo_posse * pedidos(i) * (i - 1 - j) + aux
  valor(i) = aux
  den_pedidos = pedidos(i) + den_pedidos
  If den_pedidos <> 0 Then
    media(i) = aux / den_pedidos
  End If
  If media(i) > media(i - 1) Then
    aux = custo_encomenda
    valor(i) = custo_encomenda
    media(i) = custo_encomenda
    aux = custo_encomenda
    lancamentos(j + 1) = lanca
    j = i - 1
    lanca = pedidos(j + 1)
   Else
    lanca = pedidos(i) + lanca
  End If
If lanca <> 0 Then lancamentos(j + 1) = lanca
Next
inv_i = 0
inv_inicial(0) = 0
For i = 1 To periodos
  inv_final(i) = Abs(gross(i) - inv_inicial(i) - lancamentos(i))
  custo_p(i) = inv_final(i) * custo_posse
  If i <> periodos Then inv_inicial(i + 1) = inv_final(i)
  If lancamentos(i) > 0 Then
    custo_e(i) = custo_encomenda
    Else
   custo_e(i) = 0
  End If
Next
Call desenha_grelha_mrp(periodos, gross, lancamentos, inv_inicial, inv_final, custo_p, custo_e, metodo)
End Sub

Sub MRP_part_period_balancing()
Dim pedidos(), lancamentos(), inv_inicial(), inv_final(), custo_p(), custo_e(), valor()
Dim media()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
metodo = "Part Period Balancing"
periodos = devolve_num_periodos()
ReDim pedidos(periodos), lancamentos(periodos), inv_inicial(periodos)
ReDim inv_final(periodos), custo_p(periodos), custo_e(periodos), gross(periodos)
rs.MoveLast
rs.MoveFirst
While Not rs.EOF
  rs.Edit
  If rs.Fields("component") = Combo_mrp_component_cod.Text Then
    period = rs.Fields("period")
    pedidos(period) = rs.Fields("qty")
    gross(period) = rs.Fields("qty")
    lancamentos(period) = 0
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
custo_posse = Val(mrp_custo_posse.Text)
custo_encomenda = Val(mrp_custo_encomenda.Text)
invent = CLng(mrp_initial_stock.Text)
```

```
inv_inicial(1) = invent
ReDim media(periodos), valor(periodos)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
For i = 1 To periodos
  If invent < pedidos(i) Then
    pedidos(i) = pedidos(i) - invent
    Exit For
   Else:
    invent = invent - pedidos(i)
    pedidos(i) = 0
  End If
Next i
aux = custo_encomenda
bxb = i
lanca = pedidos(bxb)
l = 0: t = bxb: pp = 0: k = bxb
For i = bxb + 1 To periodos
  pp = pp + pedidos(i) * (k - 1) * custo_posse
  If (i = periodos And pp = 0) Then ultima = 1
  valor(i) = pp
  If (pp) > custo_encomenda Then
    If pp > custo_encomenda Then
      For g = i To periodos
        va = Abs(valor(g - 1) - custo_encomenda)
        xa = Abs(valor(g) - custo_encomenda)
        If xa < va Then
          valor(i) = pp
          i = i + 1
          valor(i) = valor(i - 1) + pedidos(g + 1) * (k) * custo_posse
          k = k + 1
         Else: g = periodos + 1
        End If
      Next
    End If
    l = l + 1: aux = custo_encomenda
    lanca = 0
    For j = t To i - 1
      lanca = pedidos(j) + lanca
    Next
    lancamentos(t) = lanca
    t = i: i = i - 1: k = 0: pp = 0
  End If
  k = k + 1
Next
If pp <> 0 Then
  lanca = 0
  For j = t To i - 1
    lanca = pedidos(j) + lanca
  Next
  lancamentos(t) = lanca
End If
If ultima = 1 Then lancamentos(periodos) = pedidos(periodos)
inv_inicial(0) = 0
For i = 1 To periodos
  inv_final(i) = Abs(gross(i) - inv_inicial(i) - lancamentos(i))
  custo_p(i) = inv_final(i) * custo_posse
  If i <> periodos Then inv_inicial(i + 1) = inv_final(i)
  If lancamentos(i) > 0 Then
    custo_e(i) = custo_encomenda
    Else
     custo_e(i) = 0
  End If
Next
Call desenha_grelha_mrp(periodos, gross, lancamentos, inv_inicial, inv_final, custo_p, custo_e, metodo)
End Sub

Sub MRP_incremental_ppb()
Dim pedidos(), lancamentos(), inv_inicial(), inv_final(), custo_p(), custo_e(), valor()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
metodo = "Incremental PPB"
periodos = devolve_num_periodos()
ReDim pedidos(periodos), lancamentos(periodos), inv_inicial(periodos)
ReDim inv_final(periodos), custo_p(periodos), custo_e(periodos), gross(periodos)
rs.MoveLast
rs.MoveFirst
While Not rs.EOF
  rs.Edit
  If rs.Fields("component") = Combo_mrp_component_cod.Text Then
    period = rs.Fields("period")
    pedidos(period) = rs.Fields("qty")
    gross(period) = rs.Fields("qty")
    lancamentos(period) = 0
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
custo_posse = Val(mrp_custo_posse.Text)
custo_encomenda = Val(mrp_custo_encomenda.Text)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
ReDim media(periodos), valor(periodos)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
For i = 1 To periodos
  If invent < pedidos(i) Then
    pedidos(i) = pedidos(i) - invent
    Exit For
   Else:
    invent = invent - pedidos(i)
    pedidos(i) = 0
  End If
Next i
valor(1) = custo_encomenda
```

```
media(1) = custo_encomenda
aux = custo_encomenda
lanca = pedidos(1)
l = 0: t = 1: pp = 0: k = 1
For i = 1 To periodos
  pp = pedidos(i) * (k - 1) * custo_posse
  If (i = periodos And pp = 0) Then ultima = 1
  valor(i) = pp
  If (pp) > custo_encomenda Then
    l = l + 1: lanca = 0
    If (pedidos(i + 2) > pedidos(i + 1) * k) Then i = i + 1
    For j = t To i - 1
      lanca = pedidos(j) + lanca
    Next
    lancamentos(t) = lanca
    pp = 0: t = i: i = i - 1: k = 0
  End If
  k = k + 1
Next
If pp <> 0 Then
  lanca = 0
  For j = t To i - 1
    lanca = pedidos(j) + lanca
  Next
  lancamentos(t) = lanca
End If
If ultima = 1 Then lancamentos(periodos) = pedidos(periodos)
inv_inicial(0) = 0
For i = 1 To periodos
  inv_final(i) = Abs(gross(i) - inv_inicial(i) - lancamentos(i))
  custo_p(i) = inv_final(i) * custo_posse
  If i <> periodos Then inv_inicial(i + 1) = inv_final(i)
  If lancamentos(i) > 0 Then
    custo_e(i) = custo_encomenda
    Else
     custo_e(i) = 0
  End If
Next
Call desenha_grelha_mrp(periodos, gross, lancamentos, inv_inicial, inv_final, custo_p, custo_e, metodo)
End Sub

Function devolve_num_periodos()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
rs.MoveLast
rs.MoveFirst
max_periodo = 0
While Not rs.EOF
  rs.Edit
  If rs.Fields("component") = Combo_mrp_component_cod.Text And rs.Fields("period") > max_periodo Then
    max_periodo = rs.Fields("period")
  End If
  rs.MoveNext
Wend
devolve_num_periodos = max_periodo
rs.Close
bd.Close
End Function

Sub MRP_economic_order_quantity()
Dim pedidos(), lancamentos(), inv_inicial(), inv_final(), custo_p(), custo_e()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
metodo = "Economic Order Quantity"
periodos = devolve_num_periodos()
ReDim pedidos(periodos), lancamentos(periodos), inv_inicial(periodos)
ReDim inv_final(periodos), custo_p(periodos), custo_e(periodos), gross(periodos)
rs.MoveLast
rs.MoveFirst
While Not rs.EOF
  rs.Edit
  If rs.Fields("component") = Combo_mrp_component_cod.Text Then
    period = rs.Fields("period")
    pedidos(period) = rs.Fields("qty")
    gross(period) = rs.Fields("qty")
    lancamentos(period) = 0
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
custo_posse = Val(mrp_custo_posse.Text)
custo_encomenda = Val(mrp_custo_encomenda.Text)
invent = CLng(mrp_initial_stock.Text)
inv_inicial(1) = invent
For i = 1 To periodos
  If invent < pedidos(i) Then
    pedidos(i) = pedidos(i) - invent
    Exit For
   Else:
    invent = invent - pedidos(i)
    pedidos(i) = 0
  End If
Next i
media = 0
For i = 1 To periodos
  media = media + pedidos(i)
Next
media = media / periodos
qee = Int(Sqr(2 * custo_encomenda * media / custo_posse) + 0.5)
lanca = qee
lancamentos(1) = qee
For k = 1 To periodos
  If (qee + lanca) < pedidos(k) Then
    lancamentos(k) = pedidos(k) - lanca
    lanca = pedidos(k)
```

```
  End If
  If lanca < pedidos(k) Then
    lancamentos(k) = qee
    lanca = qee + lanca
  End If
    lanca = Abs(lanca - pedidos(k))
Next k
For i = 1 To periodos
  inv_final(i) = Abs(gross(i) - inv_inicial(i) - lancamentos(i))
  custo_p(i) = inv_final(i) * custo_posse
  If i <> periodos Then inv_inicial(i + 1) = inv_final(i)
  If lancamentos(i) > 0 Then
    custo_e(i) = custo_encomenda
   Else
     custo_e(i) = 0
  End If
Next
Call desenha_grelha_mrp(periodos, gross, lancamentos, inv_inicial, inv_final, custo_p, custo_e, metodo)
End Sub

Sub escreve_mrp_periodos()
Dim tempo_final, tempo_inicial As Long
componente = Combo_mrp_component_cod.Text
If componente = "" Then
MsgBox "Please Choose a Component.."
Exit Sub
End If
Call delete_mrp_periodos
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
material = Combo_mrp_component_cod.Text
descricao = Combo_mrp_component_desc
tempo_inicial = 0
tempo_final = 60 * 60 * CLng(mrp_hours_day.Text)
dia = 60 * 60 * CLng(mrp_hours_day.Text)
For i = 1 To CLng(mrp_days.Text)
rs.AddNew
rs.Fields("period") = i
rs.Fields("initial_time") = tempo_inicial
rs.Fields("final_time") = tempo_final
rs.Fields("component") = material
rs.Fields("desc") = descricao
tempo_inicial = tempo_final + 1
tempo_final = tempo_final + dia
rs.Update
Next i
semana = dia * CLng(mrp_day_week.Text)
fim = CLng(mrp_weeks.Text) + CLng(mrp_days.Text)
inicio = CLng(mrp_days.Text)
tempo_final = tempo_final + semana - dia
For i = 1 + inicio To fim
rs.AddNew
rs.Fields("period") = i
rs.Fields("initial_time") = tempo_inicial
rs.Fields("final_time") = tempo_final
rs.Fields("component") = material
rs.Fields("desc") = descricao
tempo_inicial = tempo_final + 1
tempo_final = tempo_final + semana
rs.Update
Next i
fim = CLng(mrp_months.Text) + CLng(mrp_days.Text) + CLng(mrp_weeks.Text)
inicio = CLng(mrp_days.Text) + CLng(mrp_weeks.Text)
mes = dia * CLng(mrp_day_month.Text)
tempo_final = tempo_final + mes - semana
For i = 1 + inicio To fim
rs.AddNew
rs.Fields("period") = i
rs.Fields("initial_time") = tempo_inicial
rs.Fields("final_time") = tempo_final
rs.Fields("component") = material
rs.Fields("desc") = descricao
tempo_inicial = tempo_final + 1
tempo_final = tempo_final + mes
rs.Update
Next i
rs.Close
bd.Close
Call atribui_valores_mrp(CStr(material))
End Sub

Sub delete_mrp_periodos()
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
If rs.RecordCount > 0 Then
rs.MoveFirst
rs.MoveLast
rs.MoveFirst
While Not rs.EOF
  rs.Delete
  rs.MoveNext
Wend
End If
rs.Close
bd.Close
End Sub

Sub atribui_valores_mrp(material As String)
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
periodos = devolve_num_periodos()
For i = 1 To periodos
  rs.Edit
  inicio = rs.Fields("initial_time")
  fim = rs.Fields("final_time")
  qtd = devolve_qt_hist(CStr(material), CLng(inicio), CLng(fim))
```

```
  rs.Fields("qty") = qtd
  rs.Update
  rs.MoveNext
Next i
rs.Close
bd.Close
End Sub

Function devolve_qt_hist(material As String, inicio As Long, fim As Long) As Long
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("TRN_mrp", dbOpenTable)
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
qtd = 0
While Not rs.EOF
  rs.Edit
  tetse = rs.Fields("time")
  mmmm = rs.Fields("component")
  If material = rs.Fields("component") And inicio <= rs.Fields("time") And fim >= rs.Fields("time") Then
    qtd = rs.Fields("qty") + qtd
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
devolve_qt_hist = qtd
End Function

Function devolve_qty_bom(produto As String, material As String)
sql = "SELECT * "
sql = sql & " FROM DEF_bom "
sql = sql & " where  DEF_bom.bom_cod='" & CStr(produto) & "' "
sql = sql & " And  DEF_bom.component='" & CStr(material) & "' "
With DataBD
  .RecordSource = sql
  .Refresh
End With
If  DataBD.Recordset.RecordCount  >  0  Then  devolve_qty_bom  =  DataBD.Recordset.Fields("qty")  Else
devolve_qty_bom = ""
End Function

Function devolve_num_maquina_nova()
Dim sql As String
sql = "SELECT DEF_types_table.Machine_cod, Count(*) AS [result] "
sql = sql & "From DEF_types_table "
sql = sql & " GROUP BY  DEF_types_table.machine_cod"
With DataBD
  .RecordSource = sql
  .Refresh
End With
devolve_num_maquina_nova = DataBD.Recordset.RecordCount + 1
End Function

Function devolve_desc_maquinas(tipo_maq As Long)
Dim sql As String
sql = "SELECT DEF_types_table.Machine_Cod, DEF_types_table.Machine_types "
sql = sql & "From DEF_types_table "
sql = sql & "WHERE DEF_types_table.Machine_cod=" & tipo_maq & ""
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
   DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("Machine_types")) Then
   devolve_desc_maquinas = CStr(DataBD.Recordset.Fields("Machine_types"))
   Else
  devolve_desc_maquinas = " "
  End If
 Else
  devolve_desc_maquinas = " "
End If
DataBD.Recordset.Close
End Function

Function devolve_maxmin_maquinas(tipo_maq As Long)
Dim sql As String
Dim res(0, 4)
sql   =   "SELECT   DEF_types_table.Machine_Cod,   DEF_types_table.max_qty,   DEF_types_table.min_qty,
DEF_types_table.max_time, DEF_types_table.min_time "
sql = sql & "From DEF_types_table "
sql = sql & "WHERE DEF_types_table.Machine_cod=" & tipo_maq & ""
With DataBD
  .RecordSource = sql
  .Refresh
End With
If DataBD.Recordset.RecordCount > 0 Then
   DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("Machine_cod")) Then
   res(0, 1) = DataBD.Recordset.Fields("max_qty")
   res(0, 2) = DataBD.Recordset.Fields("min_qty")
   res(0, 3) = DataBD.Recordset.Fields("max_time")
   res(0, 4) = DataBD.Recordset.Fields("min_time")
   devolve_maxmin_maquinas = res
   Else
  devolve_maxmin_maquinas = " "
  End If
 Else
  devolve_maxmin_maquinas = " "
End If
DataBD.Recordset.Close
End Function

Private Sub actualiza_TEMP_maquinas()
```

```
Dim a(), tipos(), maxs(), valores()
linhas = devolve_count_linhas()
ReDim linhasactivas(linhas)
ReDim maqlinha(linhas)
fim = 0
For i = 1 To 40
  qt = devolve_count_maquinas_linha(CLng(i))
  If qt > 0 Then
    linhasactivas(fim + 1) = i
    maqlinha(fim + 1) = qt
    fim = fim + 1
   End If
  If fim = linhas Then Exit For
Next i
nmaq = devolve_num_maquinas()
tipos = devolve_tipo_maquinas(CLng(nmaq))
ReDim maxs(nmaq)
ReDim valores(nmaq, 4)
For i = 1 To nmaq
 maxs(i) = devolve_maxmin_maquinas(CLng(tipos(i)))
 valores(i, 0) = devolve_desc_maquinas(CLng(tipos(i)))
 For t = 1 To 4
  valores(i, t) = maxs(i)(0, t)
 Next t
Next i
'************************apaga*********************************
With Data_limites_maquinas
  .RecordSource = "select TEMP_maquinas.* FROM TEMP_maquinas "
  .Refresh
End With
If Data_limites_maquinas.Recordset.RecordCount > 0 Then
Data_limites_maquinas.Recordset.MoveFirst
Data_limites_maquinas.Recordset.MoveLast
Data_limites_maquinas.Recordset.MoveFirst
While Not Data_limites_maquinas.Recordset.EOF
  Data_limites_maquinas.Recordset.Delete
  Data_limites_maquinas.Recordset.MoveNext
Wend
End If
Data_limites_maquinas.Recordset.Close
'*****************************************************************
With Data_limites_maquinas
  .RecordSource = "Select TEMP_maquinas.* from TEMP_maquinas "
  .Refresh
End With
c = 1
For i = 1 To linhas
  For j = 1 To maqlinha(i)
  Data_limites_maquinas.Recordset.AddNew
  Data_limites_maquinas.Recordset.Fields("Machine") = CStr(c)
  Data_limites_maquinas.Recordset.Fields("line") = CStr(linhasactivas(i))
  Data_limites_maquinas.Recordset.Fields("Description") = CStr(valores(c, 0))
  Data_limites_maquinas.Recordset.Fields("Max_Qty") = CLng(valores(c, 1))
  Data_limites_maquinas.Recordset.Fields("Min_Qty") = CLng(valores(c, 2))
  Data_limites_maquinas.Recordset.Fields("Max_Time") = CLng(valores(c, 3))
  Data_limites_maquinas.Recordset.Fields("Min_Time") = CLng(valores(c, 4))
  Data_limites_maquinas.Recordset.Update
  c = c + 1
  Next j
  Next i
Data_limites_maquinas.Recordset.Close
Call desenha_grelhas_limites
End Sub

Private Sub actualiza_TEMP_produtos()
Dim tipos(), prods(), valores(), num_prod, desc()
num_prod = devolve_num_produtos()
prods = devolve_tipo_produtos(CLng(num_prod))
valores = devolve_maxmin_produtos(CLng(num_prod))
 With data_limites_produtos
    .RecordSource = "Select TEMP_produtos.* from TEMP_produtos "
    .Refresh
 End With
'************************apaga*********************************
With data_limites_produtos
  .RecordSource = "select TEMP_produtos.* FROM TEMP_produtos "
  .Refresh
End With
If data_limites_produtos.Recordset.RecordCount > 0 Then
data_limites_produtos.Recordset.MoveFirst
data_limites_produtos.Recordset.MoveLast
data_limites_produtos.Recordset.MoveFirst
While Not data_limites_produtos.Recordset.EOF
  data_limites_produtos.Recordset.Delete
  data_limites_produtos.Recordset.MoveNext
Wend
End If
data_limites_produtos.Recordset.Close
'*****************************************************************

'************************GRAVA *********************************
With data_limites_produtos
  .RecordSource = "select TEMP_produtos.* FROM TEMP_produtos "
  .Refresh
End With
c = 1
  For i = 1 To num_prod
  data_limites_produtos.Recordset.AddNew
  data_limites_produtos.Recordset.Fields("product") = CStr(prods(i))
  data_limites_produtos.Recordset.Fields("Description") = CStr(valores(c, 0))
  data_limites_produtos.Recordset.Fields("Max_Qty") = CLng(valores(c, 1))
  data_limites_produtos.Recordset.Fields("Min_Qty") = CLng(valores(c, 2))
  data_limites_produtos.Recordset.Fields("Max_Time") = CLng(valores(c, 3))
  data_limites_produtos.Recordset.Fields("Min_Time") = CLng(valores(c, 4))
  data_limites_produtos.Recordset.Update
  c = c + 1
```

```
  Next i
data_limites_produtos.Recordset.Close
Call desenha_grelhas_limites
End Sub

Function devolve_maxmin_produtos(num_prod As Long)
Dim sql As String
Dim res()
ReDim res(num_prod, 4)
sql    =    "SELECT    DEF_product_table.desc,    DEF_product_table.max_qty,    DEF_product_table.min_qty,
DEF_product_table.max_time, DEF_product_table.min_time "
sql = sql & "From DEF_product_table "
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_prod
   res(i, 0) = DataBD.Recordset.Fields("desc")
   res(i, 1) = DataBD.Recordset.Fields("Max_qty")
   res(i, 2) = DataBD.Recordset.Fields("Min_qty")
   res(i, 3) = DataBD.Recordset.Fields("Max_time")
   res(i, 4) = DataBD.Recordset.Fields("Min_time")
  DataBD.Recordset.MoveNext
Next i
devolve_maxmin_produtos = res
DataBD.Recordset.Close
End Function

Sub actualiza_TEMP_componentes()
Dim tipos(), mats(), valores(), num_mat, desc()
num_mat = devolve_num_materiais_usados()
mats = devolve_tipo_materiais(CLng(num_mat))
valores = devolve_maxmin_componentes(CLng(num_mat))
 With data_limites_componentes
    .RecordSource = "Select TEMP_componentes.* from TEMP_componentes "
    .Refresh
 End With
'************************apaga*******************************
With data_limites_componentes
  .RecordSource = "select TEMP_componentes.* FROM TEMP_componentes "
  .Refresh
End With
If data_limites_componentes.Recordset.RecordCount > 0 Then
data_limites_componentes.Recordset.MoveFirst
data_limites_componentes.Recordset.MoveLast
data_limites_componentes.Recordset.MoveFirst
While Not data_limites_componentes.Recordset.EOF
  data_limites_componentes.Recordset.Delete
  data_limites_componentes.Recordset.MoveNext
Wend
End If
data_limites_componentes.Recordset.Close
'*****************************************************************

'************************GRAVA ******************************
With data_limites_componentes
  .RecordSource = "select TEMP_componentes.* FROM TEMP_componentes "
  .Refresh
End With
c = 1
  For i = 1 To num_mat
  data_limites_componentes.Recordset.AddNew
  data_limites_componentes.Recordset.Fields("component") = CStr(mats(i))
  data_limites_componentes.Recordset.Fields("desc") = CStr(valores(c, 0))
  data_limites_componentes.Recordset.Fields("Max_Qty") = CLng(valores(c, 1))
  data_limites_componentes.Recordset.Fields("Min_Qty") = CLng(valores(c, 2))
  data_limites_componentes.Recordset.Update
  c = c + 1
  Next i
data_limites_componentes.Recordset.Close
Call desenha_grelhas_limites
End Sub

Function devolve_maxmin_componentes(num_mat As Long)
Dim sql As String
Dim res()
ReDim res(num_mat, 2)
sql = "SELECT DEF_components.desc, DEF_components.max_qty, DEF_components.min_qty "
sql = sql & "From DEF_components "
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
For i = 1 To num_mat
   res(i, 0) = DataBD.Recordset.Fields("desc")
   res(i, 1) = DataBD.Recordset.Fields("Max_qty")
   res(i, 2) = DataBD.Recordset.Fields("Min_qty")
  DataBD.Recordset.MoveNext
Next i
devolve_maxmin_componentes = res
DataBD.Recordset.Close
End Function

Sub desenha_grelhas_limites()
max_lin = 5
maq = devolve_num_maquinas_lim()
If maq <= max_lin Then
  alt = 240 * (maq + 1)
  larg = 6830
  Else
  alt = 240 * (max_lin + 1)
  larg = 6830
```

```
End If
DBGrid_limites.Visible = True
'DBGrid_limites.Top = 2760 + 480 - 80
'DBGrid_limites.Left = 500 + 100
DBGrid_limites.Height = alt
DBGrid_limites.Width = larg
DBGrid_limites.columns(0).Caption = "Machine"
DBGrid_limites.columns(1).Caption = "Line"
DBGrid_limites.columns(2).Caption = "Description"
DBGrid_limites.columns(3).Caption = "Max Time"
DBGrid_limites.columns(4).Caption = "Min Time"
DBGrid_limites.columns(5).Caption = "Max Qty"
DBGrid_limites.columns(6).Caption = "Min Qty"
DBGrid_limites.columns(0).Width = 700
DBGrid_limites.columns(1).Width = 500
DBGrid_limites.columns(2).Width = 1500
For t = 3 To 6
  DBGrid_limites.columns(t).Width = 900
Next t
For t = 0 To 6
  DBGrid_limites.columns(t).Alignment = dbgCenter
Next t
prod = devolve_num_produtos_lim()
If prod <= max_lin Then
  alt = 240 * (prod + 1)
  larg = 6830
  Else
  alt = 240 * (max_lin + 1)
  larg = 6830
End If
'DBGrid_produtos.Top = 4680 + 840 - 80
'DBGrid_produtos.Left = 500 + 100
DBGrid_produtos.Width = larg
DBGrid_produtos.Height = alt
DBGrid_produtos.columns(0).Caption = "Product"
DBGrid_produtos.columns(1).Caption = "Description"
DBGrid_produtos.columns(2).Caption = "Max Time"
DBGrid_produtos.columns(3).Caption = "Min Time"
DBGrid_produtos.columns(4).Caption = "Max Qty"
DBGrid_produtos.columns(5).Caption = "Min Qty"
DBGrid_produtos.columns(0).Width = 700
DBGrid_produtos.columns(1).Width = 2000
For t = 2 To 5
  DBGrid_produtos.columns(t).Width = 900
Next t
For t = 0 To 5
 DBGrid_produtos.columns(t).Alignment = dbgCenter
Next t
comp = devolve_num_materiais_lim()
If comp <= max_lin Then
  linhas = comp
  alt = 240 * (comp + 1)
  larg = 5040
  Else
  alt = 240 * (max_lin + 1)
  larg = 5040
End If
'DBGrid_componentes.Top = 6600 + 1190 - 80
'DBGrid_componentes.Left = 500 + 100
DBGrid_componentes.Width = larg
DBGrid_componentes.Height = alt
DBGrid_componentes.columns(0).Caption = "Component"
DBGrid_componentes.columns(1).Caption = "Description"
DBGrid_componentes.columns(2).Caption = "Max Qty"
DBGrid_componentes.columns(3).Caption = "Min Qty"
DBGrid_componentes.columns(0).Width = 1000
DBGrid_componentes.columns(1).Width = 1700
DBGrid_componentes.columns(2).Width = 900
DBGrid_componentes.columns(3).Width = 900
For t = 0 To 3
 DBGrid_componentes.columns(t).Alignment = dbgCenter
Next t
DBGrid_componentes.Refresh
DBGrid_limites.Refresh
DBGrid_produtos.Refresh
End Sub

Function devolve_num_maquinas_lim()
Dim sql As String

sql = "SELECT Count(TEMP_maquinas.Machine) AS total "
sql = sql & "From TEMP_maquinas "
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("total")) Then
    devolve_num_maquinas_lim = DataBD.Recordset.Fields("total")
   Else
   devolve_num_maquinas_lim = 0
  End If
 Else
    devolve_num_maquinas_lim = 0
End If
DataBD.Recordset.Close
End Function

Function devolve_num_produtos_lim()
Dim sql As String

sql = "SELECT Count(TEMP_produtos.product) AS total "
sql = sql & "From TEMP_produtos "
With DataBD ' data_limites_produtos
```

```
      .RecordSource = sql
      .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("total")) Then
    devolve_num_produtos_lim = DataBD.Recordset.Fields("total")
   Else
   devolve_num_produtos_lim = 0
  End If
 Else
    devolve_num_produtos_lim = 0
End If
DataBD.Recordset.Close
End Function

Function devolve_num_materiais_lim()
Dim sql As String

sql = "SELECT Count(TEMP_componentes.component) AS total "
sql = sql & "From TEMP_componentes "
With DataBD ' data_limites_componentes
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("total")) Then
    devolve_num_materiais_lim = DataBD.Recordset.Fields("total")
   Else
   devolve_num_materiais_lim = 0
  End If
 Else
    devolve_num_materiais_lim = 0
End If
DataBD.Recordset.Close
End Function

Function escolhe_primeira_maquina(linha_actual As String)
Dim maquinas_tipos()

num_maqs = devolve_count_maquinas_linha(CStr(linha_actual))
ReDim maquinas_tipos(num_maqs)
maquinas_tipos = devolve_tipo_maquinas_linha(CStr(linha_actual))

ReDim dist(num_maqs, 1)
ReDim qtfila(num_maqs, 1)
ReDim veloc(num_maqs, 1)
ReDim tproc(num_maqs, 1)

ReDim dist(num_maqs, 1)
ReDim qtfila(num_maqs, 1)
ReDim veloc(num_maqs, 1)
ReDim tproc(num_maqs, 1)

For num = 1 To num_maqs
  distancia = 0
    For j = 1 To num
    distancia = distancia + devolve_temporota_maquina(CStr(maquinas_tipos(j)))
  Next j
 dist(num, 0) = num
 qtfila(num, 0) = num
 veloc(num, 0) = num
 tproc(num, 0) = num

 dist(num, 1) = distancia
 qtfila(num, 1) = devolve_count_planned_ord_maq(CStr(linha_actual), CStr(num))
 veloc(num, 1) = devolve_velocidade_maquina(CStr(maquinas_tipos(num)))
 tproc(num, 1) = devolve_tempo_processo(CStr(maquinas_tipos(num)))
Next num

 'escolhe a 1ª maquina para o operador
 encontrei = 0
 sucesso = 1
 iteracao = 1
 maqs_paradas = num_maqs

    Call ler_criterios_bd(CLng(linha_actual))
    While sucesso <> 0
      Select Case criterio(iteracao, 0)
      Case "Distance"
        If criterio(iteracao, 1) = 0 Then ordena_larger dist, maqs_paradas
        If criterio(iteracao, 1) = 1 Then ordena_smaller dist, maqs_paradas
        sucesso = dist(0, 0)
      Case "Speed"
        If criterio(iteracao, 1) = 0 Then ordena_larger veloc, maqs_paradas
        If criterio(iteracao, 1) = 1 Then ordena_smaller veloc, maqs_paradas
        sucesso = veloc(0, 0)
      Case "Queue"
        If criterio(iteracao, 1) = 0 Then ordena_larger qtfila, maqs_paradas
        If criterio(iteracao, 1) = 1 Then ordena_smaller qtfila, maqs_paradas
        sucesso = qtfila(0, 0)
      Case "Process Time"
        If criterio(iteracao, 1) = 0 Then ordena_larger tproc, maqs_paradas
        If criterio(iteracao, 1) = 1 Then ordena_smaller tproc, maqs_paradas
        sucesso = tproc(0, 0)
      Case Else ' se não existir criterio: escolhe a maquina que se situa mais perto
        ordena_smaller dist, maqs_paradas
        sucesso = 0
      End Select
      iteracao = iteracao + 1
      maqs_paradas = sucesso
    Wend
    encontrei = 1
```

```
    machine = dist(1, 0)
escolhe_primeira_maquina = machine
'FIM escolhe a 1ª maquina para o operador
End Function

Function ordena_larger(ParamArray lista())
Dim aux, pos, lim As Integer
Dim i As Long
lim = lista(1)
While lim > 1
  pos = 0
  For i = 1 To lim - 1
    If lista(0)(i, 1) < lista(0)(i + 1, 1) Then
      Call trocar_posicoes(i, i + 1)
      pos = i
    End If
    lim = pos
  Next i
Wend
empate = 1
i = 1
For i = 1 To lista(1) - 1
  If lista(0)(i, 1) = lista(0)(i + 1, 1) Then
    empate = empate + 1 ' existe empate
    Else
    Exit For
  End If
Next
If empate = 1 Then
  lista(0)(0, 1) = 1
  empate = 0
End If
lista(0)(0, 0) = empate
End Function

Function trocar_posicoes(X As Long, y As Long)
Dim aux As Long
aux = veloc(X, 0): veloc(X, 0) = veloc(y, 0): veloc(y, 0) = aux
aux = veloc(X, 1): veloc(X, 1) = veloc(y, 1): veloc(y, 1) = aux
aux = tproc(X, 0): tproc(X, 0) = tproc(y, 0): tproc(y, 0) = aux
aux = tproc(X, 1): tproc(X, 1) = tproc(y, 1): tproc(y, 1) = aux
aux = dist(X, 0): dist(X, 0) = dist(y, 0): dist(y, 0) = aux
aux = dist(X, 1): dist(X, 1) = dist(y, 1): dist(y, 1) = aux
aux = qtfila(X, 0): qtfila(X, 0) = qtfila(y, 0): qtfila(y, 0) = aux
aux = qtfila(X, 1): qtfila(X, 1) = qtfila(y, 1): qtfila(y, 1) = aux
End Function

Function nao_usada_devolve_especif_maqs()
Dim tipos()
Dim sql As String

sql = "SELECT DEF_types_table.Machine_Cod, DEF_types_table.proc_time, DEF_types_table.route_time "
sql = sql & "From DEF_types_table "
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
ReDim tipos(DataBD.Recordset.RecordCount, 2)
While Not DataBD.Recordset.EOF
  i = CLng(DataBD.Recordset.Fields("machine_cod"))
  tipos(i, 1) = CLng(DataBD.Recordset.Fields("route_time"))
  tipos(i, 2) = CLng(DataBD.Recordset.Fields("proc_time"))
DataBD.Recordset.MoveNext
Wend
DataBD.Recordset.Close
devolve_especif_maqs = tipos
End Function

Sub ler_criterios_bd(linha As Long)
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim sql As String
Dim encontrei As Boolean
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_opr_table")
encontrei = False
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
While Not rs.EOF And Not encontrei
  If rs.Fields("line") = CLng(linha) Then
    If (rs.Fields("first")) <> "" Then criterio(1, 0) = rs.Fields("first"): criterio(1, 1) =
rs.Fields("first_v")
    If (rs.Fields("second")) <> "" Then criterio(2, 0) = rs.Fields("second"): criterio(2, 1) =
rs.Fields("second_v")
    If (rs.Fields("third")) <> "" Then criterio(3, 0) = rs.Fields("third"): criterio(3, 1) =
rs.Fields("third_v")
    If (rs.Fields("fourth")) <> "" Then criterio(4, 0) = rs.Fields("fourth"): criterio(4, 1) =
rs.Fields("fourth_v")
    encontrei = True
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
End Sub

Function ordena_smaller(ParamArray lista())
Dim aux, pos, lim As Integer
Dim i As Long
lim = lista(1)
N = lista(1)
While lim > 1
```

```
  pos = 0
  For i = 1 To lim - 1
    If lista(0)(i, 1) > lista(0)(i + 1, 1) Then
      Call trocar_posicoes(i, i + 1)
      pos = i
    End If
    lim = pos
  Next i
Wend
empate = 1
i = 1
For i = 1 To lista(1) - 1
  If lista(0)(i, 1) = lista(0)(i + 1, 1) Then
    empate = empate + 1 ' existe empate
    Else
     Exit For
  End If
Next
If empate = 1 Then
  lista(0)(0, 1) = 1
  empate = 0
End If
lista(0)(0, 0) = empate
End Function

Function devolve_count_planned_ord_maq(linha As String, maquina As String)
Dim sql As String

maquina = "Machine_" + linha + "_" + maquina

sql = "SELECT Sum(CTRL_planned_orders.qty) AS result "
sql = sql & "From CTRL_planned_orders "
sql = sql & "Where CTRL_planned_orders.machine='" & maquina & "'"
sql = sql & " GROUP BY CTRL_planned_orders.machine"
With DataBD
  .RecordSource = sql
  .Refresh
End With
If Not DataBD.Recordset.EOF Then
  DataBD.Recordset.MoveLast
  DataBD.Recordset.MoveFirst
  If Not IsNull(DataBD.Recordset.Fields("result")) Then
    devolve_count_planned_ord_maq = DataBD.Recordset.Fields("result")
   Else
     devolve_count_planned_ord_maq = 0
    End If
   Else
  devolve_count_planned_ord_maq = 0
End If
DataBD.Recordset.Close
End Function

Function devolve_tipo_maquinas_linha(linha As String)
Dim tipos()
Dim sql As String

sql = "SELECT DEF_machine_line.Machine_Cod, DEF_machine_line.line "
sql = sql & "From DEF_machine_line "
sql = sql & "WHERE DEF_machine_line.line=" & linha

With DataBD
     .RecordSource = sql
      .Refresh
End With

DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
ReDim tipos(DataBD.Recordset.RecordCount)
For i = 1 To DataBD.Recordset.RecordCount
  tipos(i) = CLng(DataBD.Recordset.Fields("Machine_cod"))
  DataBD.Recordset.MoveNext
Next i
devolve_tipo_maquinas_linha = tipos
DataBD.Recordset.Close
End Function

Function devolve_linhas_usadas()
Dim linhas()
Dim sql As String
sql = "SELECT DEF_machine_line.line "
sql = sql & "From DEF_machine_line "
sql = sql & "GROUP BY DEF_machine_line.line "
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst
ReDim tipos(DataBD.Recordset.RecordCount)
For i = 1 To DataBD.Recordset.RecordCount
  tipos(i) = CLng(DataBD.Recordset.Fields("line"))
  DataBD.Recordset.MoveNext
Next i
devolve_linhas_usadas = tipos
DataBD.Recordset.Close
End Function

Function devolve_cod_imagem_operador_linha(linha As String)
Dim sql As String

  With DataBD
    .RecordSource = "Select DEF_line_table.*  from DEF_line_table where line=" & linha
    .Refresh
    devolve_cod_imagem_operador_linha = .Recordset("imagem_op")
  End With
  DataBD.Recordset.Close
```

```
End Function

Sub faz_grafico(texto As String, compon As String)
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTR_mrp", dbOpenTable)
rs.MoveLast
rs.MoveFirst
mrp_grafico.RowCount = devolve_num_periodos()
mrp_grafico.ColumnCount = 1
mrp_grafico.Title = compon + ":  " + texto

While Not rs.EOF
   mrp_grafico.Row = CLng(rs.Fields("period"))
   mrp_grafico.RowLabel = rs.Fields("period")
   mrp_grafico.Data = rs.Fields("qty")
   rs.MoveNext
Wend
rs.Close
bd.Close
End Sub

Sub desenha1_grafico_mrp()
grafico_mrp.RowCount = CLng(mrp_months.Text) + CLng(mrp_days.Text) + CLng(mrp_weeks.Text)
grafico_mrp.ColumnCount = 4
grafico_mrp.Column = 1
grafico_mrp.ColumnLabel = "Gross Requeriments"
grafico_mrp.Column = 2
grafico_mrp.ColumnLabel = "Orders"
grafico_mrp.Column = 3
grafico_mrp.ColumnLabel = "Initial Stock"
grafico_mrp.Column = 4
grafico_mrp.ColumnLabel = "Final Stock"
For t = 1 To 4
  grafico_mrp.SeriesColumn = t
  grid_mrp.Row = t
  For i = 1 To periodos
    grid_mrp.Col = i
    grafico_mrp.Row = i
    grafico_mrp.Column = t
    grafico_mrp.RowLabel = "P" + Trim(str(i))
    grafico_mrp.Data = grid_mrp.Text
  Next i
Next t
End Sub

Sub desenha_grelha_monitoring()

Dim num_maquinas, num_linhas, i, j, max_pro, largura, col_ini
Dim linhasactivas(), maqlinha(), maquinas_tipos()
num_maquinas = devolve_num_maquinas()
num_linhas = devolve_count_linhas()

grid_monit.Visible = True

linhas = devolve_count_linhas()
ReDim linhasactivas(linhas)
ReDim maqlinha(linhas)
fim = 0
For i = 1 To 40
  qt = devolve_count_maquinas_linha(CLng(i))
  If qt > 0 Then
    linhasactivas(fim + 1) = i
    maqlinha(fim + 1) = qt
    fim = fim + 1
   End If
  If fim = linhas Then Exit For
Next i

grid_monit.Cols = 14 + 1
grid_monit.rows = num_maquinas + 1

largura = 800
col_ini = 1500

grid_monit.Col = 0
grid_monit.Row = 0
grid_monit.ColWidth(0) = col_ini
grid_monit.CellAlignment = flexAlignCenterCenter

grid_monit.Text = "Machines"
For i = 1 To num_linhas
  grid_monit.Col = 0
  grid_monit.Text = linhasactivas(i)
  maq = 0
  For j = 1 To devolve_count_maquinas_linha(CLng(linhasactivas(i)))
     grid_monit.Row = i
     grid_monit.Col = j
     grid_monit.CellAlignment = flexAlignCenterCenter
     maq = maq + 1
     grid_monit.Text = maq
  Next j
Next i
End Sub

Private Sub cmdGraphics_Click()
Dim txt, componente As String

If Optionres(0).Value = True Then opcao = 0
If Optionres(1).Value = True Then opcao = 1
If Optionres(2).Value = True Then opcao = 2
If Optionres(3).Value = True Then opcao = 3
If Optionres(4).Value = True Then opcao = 4
If Optionres(5).Value = True Then opcao = 5
If Optionres(6).Value = True Then opcao = 6
If Optionres(7).Value = True Then opcao = 7

res_grafico.Stacking = Check_res_grafico_stack.Value
```

```
If Check_res_grafico_legend.Value Then
res_grafico.Legend.Location.LocationType = VtChLocationTypeBottom
res_grafico.Legend.Location.Visible = True
Else
res_grafico.Legend.Location.Visible = False
End If

Select Case opcao
  Case 0:
    Call res_man1
  Case 1:
    Call res_man2
  Case 2:
    Call res_mat1
  Case 3:
    Call res_war1
  Case 4:
    Call res_opr1
  Case 5:
    Call res_opr2
  Case 6:
    Call res_maq1
  Case 7:
    Call res_opr3
End Select

End Sub

Sub res_color(n_colunas As Long)

For t = 1 To n_colunas
   With res_grafico.Plot.SeriesCollection(t).DataPoints(-1).Brush.FillColor
      .Red = 0
      .Green = 0
      .Blue = 255
   End With
Next t
   End Sub
Sub res_escala()
   ' The ComboBox has three items: Log, Percent,
   ' and Linear (the default scale).
cmbScale = "Linear"
   Select Case cmbScale
   Case "Log"
      res_grafico.Plot.Axis(VtChAxisIdY) _
      .AxisScale.Type = VtChScaleTypeLogarithmic

      ' You must specify a LogBase to be used when
      ' switching the scale to Log. The base can be
      ' set to any value between 2 and 200.
      res_grafico.Plot.Axis(VtChAxisIdY).AxisScale _
      .LogBase = 10

   Case "Percent"
      res_grafico.Plot.Axis(VtChAxisIdY).AxisScale _
      .Type = VtChScaleTypePercent
      ' Set the PercentBasis to one of six types. For
      ' the sake of expediency, only one is shown.
      res_grafico.Plot.Axis(VtChAxisIdY).AxisScale _
      .PercentBasis = VtChPercentAxisBasisMaxChart

   Case "Linear"
     res_grafico.Plot.Axis(VtChAxisIdY).AxisScale _
      .Type = VtChScaleTypeLinear
   End Select
End Sub

Sub res_man1()
On Error GoTo erro_db_nao_existe
Dim sql As String
Dim n_colunas As Long

sql = "SELECT TRN_maintenance.line AS Line, TRN_maintenance.machine AS Machine, Sum([end]-[begin]) AS
[Time_in_Failure], Count(*) AS [Number_of_Failures] "
sql = sql & "From TRN_maintenance "
sql = sql & "GROUP BY TRN_maintenance.line, TRN_maintenance.machine "

With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst

n_colunas = DataBD.Recordset.RecordCount

ReDim tipos(n_colunas, 4)

For i = 1 To n_colunas
  tipos(i, 1) = CLng(DataBD.Recordset.Fields("line"))
  tipos(i, 2) = CLng(DataBD.Recordset.Fields("Machine"))
  tipos(i, 3) = CLng(DataBD.Recordset.Fields("Time_in_Failure"))
  tipos(i, 4) = CLng(DataBD.Recordset.Fields("Number_of_Failures"))
  DataBD.Recordset.MoveNext
Next i
DataBD.Recordset.Close

res_grafico.chartType = CLng(txt_res_grafico.Text)

If  res_grafico.chartType  =  2  Or  res_grafico.chartType  =  3  Or  res_grafico.chartType  =  5  Or
res_grafico.chartType = 7 Or res_grafico.chartType = 9 Or res_grafico.chartType = 16 Then
 res_grafico.RowCount = n_colunas
 res_grafico.ColumnCount = 1
 res_grafico.Title = "DURATION OF FAILUES"
```

```
 For i = 1 To n_colunas
   res_grafico.Row = i
   res_grafico.RowLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Column = 1
   res_grafico.Data = tipos(i, 3)

Next
If res_grafico.chartType <> 16 Then
  res_grafico.Column = 1
  res_grafico.ColumnLabel = "Time in Seconds"
End If

Else

res_grafico.RowCount = 1
res_grafico.ColumnCount = n_colunas
res_grafico.Title = "DURATION OF FAILUES"

For i = 1 To n_colunas
'res_grafico.Plot.SeriesCollection(i).DataPoints(-1).Brush.FillColor.Set 255, 0, 0
  res_grafico.Row = 1
  res_grafico.RowLabel = ""
  res_grafico.Column = i
  res_grafico.ColumnLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
  res_grafico.Data = tipos(i, 3) + 5000
Next
End If

Exit Sub
erro_db_nao_existe:
  MsgBox "Missing Data to Make this Graphic"
End Sub

Sub res_man2()
On Error GoTo erro_db_nao_existe
Dim sql As String
Dim n_colunas As Long

sql = "SELECT TRN_maintenance.line AS Line, TRN_maintenance.machine AS Machine, Sum([end]-[begin]) AS
[Time_in_Failure], Count(*) AS [Number_of_Failures] "
sql = sql & "From TRN_maintenance "
sql = sql & "GROUP BY TRN_maintenance.line, TRN_maintenance.machine "

With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst

n_colunas = DataBD.Recordset.RecordCount

ReDim tipos(n_colunas, 4)

For i = 1 To n_colunas
  tipos(i, 1) = CLng(DataBD.Recordset.Fields("line"))
  tipos(i, 2) = CLng(DataBD.Recordset.Fields("Machine"))
  tipos(i, 3) = CLng(DataBD.Recordset.Fields("Time_in_Failure"))
  tipos(i, 4) = CLng(DataBD.Recordset.Fields("Number_of_Failures"))
  DataBD.Recordset.MoveNext
Next i
DataBD.Recordset.Close

res_grafico.chartType = CLng(txt_res_grafico.Text)
'res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 50, 10, 250

If  res_grafico.chartType  =  2  Or  res_grafico.chartType  =  3  Or  res_grafico.chartType  =  5  Or
res_grafico.chartType = 7 Or res_grafico.chartType = 9 Or res_grafico.chartType = 16 Then
 res_grafico.RowCount = n_colunas
 res_grafico.ColumnCount = 1
 res_grafico.Title = "NUMBER OF FAILURES"
 For i = 1 To n_colunas
   res_grafico.Row = i
   res_grafico.RowLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Column = 1
   res_grafico.Data = tipos(i, 4)
Next
If res_grafico.chartType <> 16 Then
  res_grafico.Column = 1
  res_grafico.ColumnLabel = "Quantity"
End If
Else
res_grafico.RowCount = 1
res_grafico.ColumnCount = n_colunas
res_grafico.Title = "NUMBER OF FAILURES"
For i = 1 To n_colunas
res_grafico.Row = 1
   res_grafico.RowLabel = ""
   res_grafico.Column = i
      res_grafico.ColumnLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Data = tipos(i, 4)
Next
End If

Exit Sub
erro_db_nao_existe:
  MsgBox "Missing Data to Make this Graphic"
End Sub

Sub res_opr1()
On Error GoTo erro_db_nao_existe
Dim sql As String
Dim n_colunas As Long

sql  =  "SELECT  TRN_operator_maq.line  AS  Line,  TRN_operator_maq.next_machine  AS  Machine,
Sum(TRN_operator_maq.distance) AS [Time_in_Distance] "
```

```
sql = sql & "From TRN_operator_maq "
sql = sql & "WHERE TRN_operator_maq.next_machine>'" & "0" & "'"
sql = sql & "GROUP BY TRN_operator_maq.line, TRN_operator_maq.next_machine "

With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst

n_colunas = DataBD.Recordset.RecordCount

ReDim tipos(n_colunas, 3)

For i = 1 To n_colunas
  tipos(i, 1) = CLng(DataBD.Recordset.Fields("line"))
  tipos(i, 2) = CLng(DataBD.Recordset.Fields("Machine"))
  tipos(i, 3) = CLng(DataBD.Recordset.Fields("Time_in_Distance"))
  DataBD.Recordset.MoveNext
Next i
DataBD.Recordset.Close

res_grafico.chartType = CLng(txt_res_grafico.Text)
'res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 50, 10, 250

If  res_grafico.chartType  =  2  Or  res_grafico.chartType  =  3  Or  res_grafico.chartType  =  5  Or
res_grafico.chartType = 7 Or res_grafico.chartType = 9 Or res_grafico.chartType = 16 Then
 res_grafico.RowCount = n_colunas
 res_grafico.ColumnCount = 1
 res_grafico.Title = "OPERATOR ROUTE TIME"
 For i = 1 To n_colunas
   res_grafico.Row = i
   res_grafico.RowLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Column = 1
   res_grafico.Data = tipos(i, 3)
Next
If res_grafico.chartType <> 16 Then
  res_grafico.Column = 1
  res_grafico.ColumnLabel = "Time in Seconds"
End If

Else
res_grafico.RowCount = 1
res_grafico.ColumnCount = n_colunas
res_grafico.Title = "OPERATOR ROUTE TIME"
For i = 1 To n_colunas
res_grafico.Row = 1
   res_grafico.RowLabel = ""
   res_grafico.Column = i
      res_grafico.ColumnLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Data = tipos(i, 3)
Next

End If

Exit Sub
erro_db_nao_existe:
  MsgBox "Missing Data to Make this Graphic"
End Sub

Sub res_opr3()
On Error GoTo erro_db_nao_existe
Dim sql As String
Dim n_colunas As Long

sql = "SELECT TRN_operator_maq.line AS Line, TRN_operator_maq.next_machine AS Machine, Count(*) AS
[Number_of_Deslocations], Last(TRN_operator_maq.working_until) AS terminou "
sql = sql & "From TRN_operator_maq "
sql = sql & "WHERE TRN_operator_maq.next_machine>'" & "0" & "'"
sql = sql & "GROUP BY TRN_operator_maq.line, TRN_operator_maq.next_machine "

With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst

n_colunas = DataBD.Recordset.RecordCount

ReDim tipos(n_colunas + 1, 3)

fim = 0
For i = 1 To n_colunas
  tipos(i, 0) = CLng(DataBD.Recordset.Fields("terminou"))
  tipos(i, 1) = CLng(DataBD.Recordset.Fields("line"))
  tipos(i, 2) = CLng(DataBD.Recordset.Fields("Machine"))
  tipos(i, 3) = CLng(DataBD.Recordset.Fields("Number_of_Deslocations"))
  If tipos(i, 0) > fim Then fim = tipos(i, 0)
  DataBD.Recordset.MoveNext
Next i
DataBD.Recordset.Close

For i = 1 To n_colunas
  tipos(i, 3) = tipos(i, 3) * CLng(devolve_tempo_operador(CStr(tipos(i, 1))))
Next i

res_grafico.chartType = CLng(txt_res_grafico.Text)

If  res_grafico.chartType  =  2  Or  res_grafico.chartType  =  3  Or  res_grafico.chartType  =  5  Or
res_grafico.chartType = 7 Or res_grafico.chartType = 9 Or res_grafico.chartType = 16 Then
'res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 50, 10, 250
res_grafico.Legend.Location.LocationType = VtChLocationTypeBottom
res_grafico.Legend.TextLayout.HorzAlignment = VtHorizontalAlignmentCenter

 res_grafico.RowCount = n_colunas
```

```
 res_grafico.ColumnCount = 1
res_grafico.Title = "OPERATOR EFFECTIVE WORK"
 For i = 1 To n_colunas
   res_grafico.Row = i
   res_grafico.RowLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Column = 1
   res_grafico.Data = tipos(i, 3)

Next
If res_grafico.chartType <> 16 Then
  res_grafico.Column = 1
  res_grafico.ColumnLabel = "Time of Effective Work"
End If

Else
res_grafico.RowCount = 1

res_grafico.ColumnCount = n_colunas
res_grafico.Title = "OPERATOR EFFECTIVE WORK"
For i = 1 To n_colunas
res_grafico.Row = 1
   res_grafico.RowLabel = ""
   res_grafico.Column = i
   res_grafico.ColumnLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Data = tipos(i, 3)
Next
End If

Exit Sub
erro_db_nao_existe:
  MsgBox "Missing Data to Make this Graphic"
End Sub

Sub res_war1()
On Error GoTo erro_db_nao_existe
Dim sql As String
Dim n_colunas As Long

sql     =     "SELECT     TRN_warehouse.component     AS     Component,     Count(TRN_warehouse.qty)     AS
[Number_of_Transactions]"
sql = sql & "From TRN_warehouse "
sql = sql & "GROUP BY TRN_warehouse.component "

With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst

n_colunas = DataBD.Recordset.RecordCount

ReDim tipos(n_colunas, 2)

For i = 1 To n_colunas
  tipos(i, 1) = CStr(DataBD.Recordset.Fields("component"))
  tipos(i, 2) = CLng(DataBD.Recordset.Fields("Number_of_Transactions"))
  DataBD.Recordset.MoveNext
Next i
DataBD.Recordset.Close

res_grafico.chartType = CLng(txt_res_grafico.Text)
'res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 50, 10, 250

If  res_grafico.chartType  =  2  Or  res_grafico.chartType  =  3  Or  res_grafico.chartType  =  5  Or
res_grafico.chartType = 7 Or res_grafico.chartType = 9 Or res_grafico.chartType = 16 Then
 res_grafico.RowCount = n_colunas
 res_grafico.ColumnCount = 1
 res_grafico.Title = "NUMBER OF WAREHOUSE TRANSACTIONS"
 For i = 1 To n_colunas
   res_grafico.Row = i
   res_grafico.RowLabel = CStr(tipos(i, 1))
   res_grafico.Column = 1
   res_grafico.Data = tipos(i, 2)
Next
If res_grafico.chartType <> 16 Then
  res_grafico.Column = 1
  res_grafico.ColumnLabel = "Quantity"
End If

Else
res_grafico.RowCount = 1
res_grafico.ColumnCount = n_colunas
res_grafico.Title = "NUMBER OF WAREHOUSE TRANSACTIONS"
For i = 1 To n_colunas
res_grafico.Row = 1
   res_grafico.RowLabel = ""
   res_grafico.Column = i
      res_grafico.ColumnLabel = CStr(tipos(i, 1))
   res_grafico.Data = tipos(i, 2)
Next
End If

Exit Sub
erro_db_nao_existe:
  MsgBox "Missing Data to Make this Graphic"
End Sub
Sub res_mat1()
On Error GoTo erro_db_nao_existe
Dim sql As String
Dim n_colunas As Long

sql = "SELECT TRN_mrp.Component, Sum(TRN_mrp.qty) AS Quantity "
sql = sql & "From TRN_mrp "
sql = sql & "GROUP BY TRN_mrp.Component "
```

```
With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst

n_colunas = DataBD.Recordset.RecordCount

ReDim tipos(n_colunas, 2)

For i = 1 To n_colunas
  tipos(i, 1) = CStr(DataBD.Recordset.Fields("Component"))
  tipos(i, 2) = CLng(DataBD.Recordset.Fields("Quantity"))
  DataBD.Recordset.MoveNext
Next i
DataBD.Recordset.Close

res_grafico.chartType = CLng(txt_res_grafico.Text)
'res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 50, 10, 250
If  res_grafico.chartType  =  2  Or  res_grafico.chartType  =  3  Or  res_grafico.chartType  =  5  Or
res_grafico.chartType = 7 Or res_grafico.chartType = 9 Or res_grafico.chartType = 16 Then
 res_grafico.RowCount = n_colunas
 res_grafico.ColumnCount = 1
 res_grafico.Title = "QUANTITIES OF RAW MATERIAL"
 For i = 1 To n_colunas
   res_grafico.Row = i
   res_grafico.RowLabel = CStr(tipos(i, 1))
   res_grafico.Column = 1
   res_grafico.Data = tipos(i, 2)
Next
If res_grafico.chartType <> 16 Then
  res_grafico.Column = 1
  res_grafico.ColumnLabel = "Quantity"
End If

Else
res_grafico.RowCount = 1
res_grafico.ColumnCount = n_colunas
res_grafico.Title = "QUANTITIES OF RAW MATERIAL"
For i = 1 To n_colunas
res_grafico.Row = 1
   res_grafico.RowLabel = ""
   res_grafico.Column = i
      res_grafico.ColumnLabel = CStr(tipos(i, 1))
   res_grafico.Data = tipos(i, 2)
Next
End If

Exit Sub
erro_db_nao_existe:
  MsgBox "Missing Data to Make this Graphic"
End Sub
```

```
Sub res_maq1()
On Error GoTo erro_db_nao_existe
Dim sql As String
Dim n_colunas As Long

sql = "SELECT TRN_machine.line, TRN_machine.machine, TRN_machine.obs, Count(*) AS Count "
sql = sql & "From TRN_machine "
sql = sql & "WHERE TRN_machine.obs<>'" & "TERMINOU" & "'"
sql = sql & "GROUP BY TRN_machine.line, TRN_machine.machine, TRN_machine.obs "

With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst

n_colunas = DataBD.Recordset.RecordCount
valorgrafico(0) = n_colunas

ReDim tipos(n_colunas, 4)

For i = 1 To n_colunas
  tipos(i, 1) = CLng(DataBD.Recordset.Fields("line"))
  tipos(i, 2) = CLng(DataBD.Recordset.Fields("Machine"))
  tipos(i, 3) = CStr(DataBD.Recordset.Fields("Obs"))
  tipos(i, 4) = CLng(DataBD.Recordset.Fields("Count"))
  DataBD.Recordset.MoveNext

Next i
DataBD.Recordset.Close

res_grafico.ColumnCount = 4

res_grafico.Column = 1
res_grafico.ColumnLabel = tipos(1, 3)
res_grafico.Column = 2
res_grafico.ColumnLabel = tipos(2, 3)
res_grafico.Column = 3
res_grafico.ColumnLabel = tipos(3, 3)
res_grafico.Column = 4
res_grafico.ColumnLabel = tipos(4, 3)

res_grafico.chartType = CLng(txt_res_grafico.Text)
res_grafico.RowCount = n_colunas / 4
res_grafico.ColumnCount = 4
res_grafico.Title = "STATUS OF MACHINES"

contador = 0
zx = 1
zxz = 1
For i = 1 To n_colunas
   res_grafico.Row = zxz
```

```
   res_grafico.RowLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Column = zx
   res_grafico.Data = tipos(i, 4)
   contador = contador + 1
   zx = zx + 1
   If zx = 5 Then zx = 1: zxz = zxz + 1
Next
res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 255, 0, 0      ' vermelho
res_grafico.Plot.SeriesCollection(2).DataPoints(-1).Brush.Style = VtBrushStyleHatched
res_grafico.Plot.SeriesCollection(2).DataPoints(-1).Brush.Index = VtBrushPatternInvertedTrellis
res_grafico.Plot.SeriesCollection(2).DataPoints(-1).Brush.FillColor.Set 0, 205, 150      'verde escuro
res_grafico.Plot.SeriesCollection(2).DataPoints(-1).Brush.PatternColor.Set   255,   255,   205   '   riscas
amarelas
res_grafico.Plot.SeriesCollection(3).DataPoints(-1).Brush.FillColor.Set 190, 190, 190  ' cinzento
res_grafico.Plot.SeriesCollection(4).DataPoints(-1).Brush.FillColor.Set 100, 255, 50    ' verde
Exit Sub
erro_db_nao_existe:
  MsgBox "Missing Data to Make this Graphic"
End Sub
Sub res_opr2()
On Error GoTo erro_db_nao_existe
Dim sql As String
Dim n_colunas As Long

sql  =  "SELECT  TRN_operator_maq.line  AS  Line,  TRN_operator_maq.next_machine  AS  Machine,  Count(*)  AS
[Number_of_Deslocations] "
sql = sql & "From TRN_operator_maq "
sql = sql & "WHERE TRN_operator_maq.next_machine>'" & "0" & "'"
sql = sql & "GROUP BY TRN_operator_maq.line, TRN_operator_maq.next_machine "

With DataBD
  .RecordSource = sql
  .Refresh
End With
DataBD.Recordset.MoveLast
DataBD.Recordset.MoveFirst

n_colunas = DataBD.Recordset.RecordCount

ReDim tipos(n_colunas, 3)

For i = 1 To n_colunas
  tipos(i, 1) = CLng(DataBD.Recordset.Fields("line"))
  tipos(i, 2) = CLng(DataBD.Recordset.Fields("Machine"))
  tipos(i, 3) = CLng(DataBD.Recordset.Fields("Number_of_Deslocations"))
  DataBD.Recordset.MoveNext
Next i
DataBD.Recordset.Close

res_grafico.chartType = CLng(txt_res_grafico.Text)

If  res_grafico.chartType  =  2  Or  res_grafico.chartType  =  3  Or  res_grafico.chartType  =  5  Or
res_grafico.chartType = 7 Or res_grafico.chartType = 9 Or res_grafico.chartType = 16 Then
 res_grafico.RowCount = n_colunas
 res_grafico.ColumnCount = 1
res_grafico.Title = "NUMBER OF OPERATOR DESLOCATIONS"
 For i = 1 To n_colunas
   res_grafico.Row = i
   res_grafico.RowLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Column = 1
   res_grafico.Data = tipos(i, 3)
Next
If res_grafico.chartType <> 16 Then
  res_grafico.Column = 1
  res_grafico.ColumnLabel = "Quantity"
End If
'res_grafico.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 50, 10, 250

Else

res_grafico.RowCount = 1
res_grafico.ColumnCount = n_colunas
res_grafico.Title = "NUMBER OF OPERATOR DESLOCATIONS"
For i = 1 To n_colunas
res_grafico.Row = 1
   res_grafico.RowLabel = ""
   res_grafico.Column = i
      res_grafico.ColumnLabel = "M_" & CStr(tipos(i, 1)) & "_" & CStr(tipos(i, 2))
   res_grafico.Data = tipos(i, 3)
Next
End If

Exit Sub
erro_db_nao_existe:
  MsgBox "Missing Data to Make this Graphic"
End Sub

Private Sub cmd_copy_Click()
    res_grafico.EditCopy
End Sub
```

Anexo B

Modelo Base

```
Dim a As Arena.Application
Dim M As Arena.Model
Dim s As Arena.SIMAN
Dim maqlinha(), linhasactivas(), operador_livre(), opr_time()
Dim setupmaq(), velocidade(), TempoRota(), tipos_produto()
Dim dist(), qtfila(), veloc(), tproc(), linhaman(), lote(), rejeitada()
Dim criterio(), bundle, estado_maquinas(), estado_maquinas_old(), Ocupacao_opr()
Dim manutencao(4), maq_func_avaria()

Dim temini, tempfini

Private Sub ModelLogic_RunBeginSimulation()

Call inicializacao
Call apaga_tabelas_temporarias       ' TRN tables
temini = Time & "---" & Date
Call atribuicao_inicial_de_pedidos   ' coloca ordem na fila fio

For linhas = 1 To UBound(linhasactivas, 1)
 For z = 1 To maqlinha(linhasactivas(linhas))
   cod = tipos_produto(ler_variavel(CStr(linhasactivas(linhas)), CStr(z), "TipoProduto"), 0)
   If cod = "" Then cod = "#"
   Call                    maquina_control(CLng(linhasactivas(linhas)),                    CLng(z),
CStr(estado_maquinas_old(linhasactivas(linhas), z)), CStr(cod))

   txt = "Machine_" + CStr(linhasactivas(linhas)) + "_" + CStr(z)
   If Not tem_orders(CStr(txt)) Then
    Call maquina_control(CLng(linhasactivas(linhas)), CLng(z), "TERMINOU", " ")
    rejeitada(CLng(linhasactivas(linhas)), CLng(z)) = -1
   End If
 Next
Next

M.RunSpeed = 100

End Sub

Private Sub VBA_Block_1_Fire()

Dim i, maqx, FF, qty, linhas, maq_paradas As Long
Dim txt, cod, prod_novo, prod_declarado As String

Call gestao_manutencao

For linhas = 1 To UBound(linhasactivas, 1)
  Call alteracao_estados_maquinas(CLng(linhas))

  opmaq = local_operador(CStr(linhasactivas(linhas)))

  If opmaq <> 0 Then
   qty = ler_fila(CStr(linhasactivas(linhas)), CStr(opmaq), "Fila Pacote")
   FF = ler_fila(CStr(linhasactivas(linhas)), CStr(opmaq), "Fila Fio")
   If qty <> 0 And FF = 0 And operador_livre(CLng(linhasactivas(linhas)), 1) <= s.RunCurrentTime Then
     cod = tipos_produto(ler_variavel(CStr(linhasactivas(linhas)), CStr(opmaq), "TipoProduto"), 0)
     qty = ler_fila(CStr(linhasactivas(linhas)), CStr(opmaq), "Fila Pacote")

     prod_declarado = cod
     Call declaracao_producao(CStr(cod), CLng(qty), CLng(lote(linhasactivas(linhas), CLng(opmaq))))
     Call apaga_ordem_declarada(CLng(lote(linhasactivas(linhas), opmaq)))
     Call atribui_pedidos(CLng(linhasactivas(linhas)), CLng(opmaq))

     prod_novo = tipos_produto(ler_variavel(CStr(linhasactivas(linhas)), CLng(opmaq), "TipoProduto"),
0)

     If prod_novo <> prod_declarado Then
       operador_livre(linhasactivas(linhas),   1)   =   operador_livre(linhasactivas(linhas),   1)   +
setupmaq(linhasactivas(linhas))
       Call      operador_control(CStr(linhasactivas(linhas)),      CStr(opmaq),      "      ",      0,
CLng(operador_livre(linhasactivas(linhas), 1)), "Tempo de Setup")
       prod_declarado = prod_novo
     End If
     Call escreve_fila(CStr(linhasactivas(linhas)), CStr(opmaq), "Fila Pacote", "0")
   End If
   txt = "Machine_" + CStr(linhasactivas(linhas)) + "_" + CStr(opmaq)
   If Not tem_orders(CStr(txt)) And rejeitada(CLng(linhasactivas(linhas)), CLng(opmaq)) <> -1 Then
     Call maquina_control(CLng(linhasactivas(linhas)), CLng(opmaq), "TERMINOU", " ")
     rejeitada(CLng(linhasactivas(linhas)), CLng(opmaq)) = -1
   End If

  If operador_livre(CLng(linhasactivas(linhas)), 1) <= s.RunCurrentTime  Then 'And s.RunCurrentTime <> 0
Then 'operador_livre(CLng(linhasactivas(linhas)), 1) = 0 Then
   If Ocupacao_opr(linhasactivas(linhas)) <> "Operator FREE" Then
     Call       operador_control(CStr(linhasactivas(linhas)),       "       ",       "       ",       0,
CLng(operador_livre(CLng(linhasactivas(linhas)), 1)), "Operator FREE")
     Ocupacao_opr(linhasactivas(linhas)) = "Operator FREE"
    End If
   ok = 0
   If tem_orders(CStr(txt)) And ler_variavel(CStr(linhasactivas(linhas)), CStr(opmaq), "TempoSetup") <>
"1" Then
     Call coloca_maquina_apta_para_trabalhar(CLng(linhasactivas(linhas)), CLng(opmaq))
     ok = 1
   End If

   If    maquinas_paradas(CLng(linhasactivas(linhas)))    =    0    And    tem_orders(CStr(txt))    And
estado_maquinas(linhasactivas(linhas), opmaq) = "Parada" Then
     Call envia_opr_fica_na_mesma(CStr(linhasactivas(linhas)), CStr(opmaq))
     If ok = 1 Then
       operador_livre(CLng(linhasactivas(linhas)), 0) = CLng(opmaq)
       operador_livre(CLng(linhasactivas(linhas)),        1)        =        s.RunCurrentTime        +
opr_time(CLng(linhasactivas(linhas)), 1)
       Call     operador_control(CStr(linhasactivas(linhas)),     CStr(opmaq),     CStr(opmaq),     0,
CLng(operador_livre(CLng(linhasactivas(linhas)), 1)), "          Operator")
```

```
      End If
    End If

    maq_paradas = maquinas_paradas(CLng(linhasactivas(linhas)))
    If maq_paradas > 0 Then
      Call gestao_operador(CLng(opmaq), CLng(linhasactivas(linhas)), CLng(maq_paradas))
    End If
  End If

  End If

  If estado_maquinas(CStr(linhasactivas(linhas)), operador_livre(linhasactivas(linhas), 0)) = "Avariada"
Then
    opmaq = operador_livre(CLng(linhasactivas(linhas)), 0)
    operador_livre(CLng(linhasactivas(linhas)), 1) = 0
    Call  operador_control(CStr(linhasactivas(linhas)),  "  ",  "  ",  0,  CLng(operador_livre(1,  1)),
"Operator FREE Maq avariou")
    maq_paradas = maquinas_paradas(CLng(linhasactivas(linhas)))
    If maq_paradas > 0 Then
      Call gestao_operador(CLng(opmaq), CLng(linhasactivas(linhas)), CLng(maq_paradas))
    End If
  End If
Next linhas

If terminou_a_simulacao And s.RunCurrentTime <> 0 And Not existe_planned_orders Then
  Call operador_control(CStr("0"), CStr(opmaq), CStr("0"), 0, CLng(operador_livre(1, 1)), "F I M")
  tempfini = Time & "---" & Date
  Call operador_control(0, 0, 0, 0, 0, CStr(temini))
  Call operador_control(0, 0, 0, 0, 0, CStr(tempfini))
  M.End
End If
End Sub

Sub escreve_variavel(linha As String, maquina As String, variavel As String, valor As String)
Set M = ThisDocument.Model
Set s = M.SIMAN
txt = variavel + "_" + linha + "_" + CStr(maquina)
s.VariableArrayValue(s.SymbolNumber(txt)) = CStr(valor)
End Sub

Sub escreve_expressao(linha As String, maquina As String, expressao As String, valor As String)
Set M = ThisDocument.Model
Set s = M.SIMAN

x = s.ExpressionsMaximum
txt = expressao + "_" + linha + "_" + CStr(maquina)
For t = 1 To x
  If txt = s.ConstructString(smSimanConstructExpressions, t, 1) Then
    Exit For
  End If
Next t
xx = s.ExpressionValue(t)
xxg = s.ExpressionValueWithIndices(t, 0, 0)

End Sub

Sub gestao_operador(opmaq As Long, linhas As Integer, maqs_paradas As Long)
Dim maislenta, tem_fio, tempo As Long
Dim i, encontrei As Long
Dim estado As Boolean, machine As String
encontrei = 0

If s.RunCurrentTime >= operador_livre(linhas, 1) And operador_livre(linhas, 0) <> 0 Then
  operador_livre(linhas, 1) = 0
  operador_livre(linhas, 0) = 0
  If maqs_paradas > 0 Then
    sucesso = 1
    iteracao = 1
    Call ler_criterios_bd(CLng((linhas)))
    While (sucesso <> 0 And iteracao < 5) ' 4 criterios
      Select Case criterio(iteracao, 0)
      Case "Distance"
        If criterio(iteracao, 1) = 0 Then ordena_larger dist, maqs_paradas, 0
        If criterio(iteracao, 1) = 1 Then ordena_smaller dist, maqs_paradas, 0
        sucesso = dist(0, 0)
        seguinte = dist(1, 0)
      Case "Speed"
        If criterio(iteracao, 1) = 0 Then ordena_larger veloc, maqs_paradas, 0
        If criterio(iteracao, 1) = 1 Then ordena_smaller veloc, maqs_paradas, 0
        sucesso = veloc(0, 0)
        seguinte = veloc(1, 0)
      Case "Queue"
        If criterio(iteracao, 1) = 0 Then ordena_larger qtfila, maqs_paradas, 0
        If criterio(iteracao, 1) = 1 Then ordena_smaller qtfila, maqs_paradas, 0
        sucesso = qtfila(0, 0)
        seguinte = qtfila(1, 0)
      Case "Process Time"
        If criterio(iteracao, 1) = 0 Then ordena_larger tproc, maqs_paradas, 0
        If criterio(iteracao, 1) = 1 Then ordena_smaller tproc, maqs_paradas, 0
        sucesso = tproc(0, 0)
        seguinte = tproc(1, 0)
      Case Else ' se não existir criterio: escolhe a maquina mais perto

        ordena_smaller dist, maqs_paradas
        sucesso = dist(0, 1)
        seguinte = dist(1, 0)

      End Select
      iteracao = iteracao + 1
      maqs_paradas = sucesso
    Wend
    encontrei = 1
    machine = seguinte
  End If

  If encontrei Then
    Call envia_operador(CStr((linhas)), CStr(opmaq), CStr(machine))
```

```
    Call operador_control(CStr((linhas)), CStr(opmaq), CStr(machine), CLng(dist(1, 1)),
CLng(operador_livre(1, 1)), "Move Operator")
    Ocupacao_opr(linhas) = "Operator FREE"
  End If
End If
End Sub

Sub aceita_maquina(num As Long, maq_destino As Long, linhas As Long, opmaq As Long)
  dist(num, 0) = maq_destino
  qtfila(num, 0) = maq_destino
  veloc(num, 0) = maq_destino
  tproc(num, 0) = maq_destino
  dist(num, 1) = TempoCaminho(CLng(linhas), CLng(opmaq), CLng(maq_destino))
  qtfila(num, 1) = ler_pedidos_em_espera(CStr(linhas), CStr(maq_destino))
  veloc(num, 1) = velocidade(ler_variavel(CStr(linhas), CStr(maq_destino), "TipoMaquina"))
  tproc(num, 1) = ler_variavel(CStr(linhas), CStr(maq_destino), "TempoProc")
End Sub

Sub insere_nas_filas(maquina As String, qty As Long, tipo_fio As String, tipo_produto As String,
tipo_palete As String)
Dim i, j, z As Long
Dim im As Long
Set M = ThisDocument.Model
Set s = M.SIMAN

fila = maquina + ".Fila Fio"
For i = 1 To M.Queues.Count
  If fila = s.ConstructString(smSimanConstructQueues, i, 1) Then
    j = s.EntityCreate
    Exit For
  End If
Next
For z = 1 To qty
  j = s.EntityCreate
  im = s.SymbolNumber(tipo_fio)
  s.EntitySetPicture j, im
  s.EntityInsertIntoQueue j, i
Next
End Sub

Function ler_variavel(linha As String, maquina As String, variavel As String) As Long
Set M = ThisDocument.Model
Set s = M.SIMAN
  txt = variavel + "_" + linha + "_" + maquina
  ler_variavel = s.VariableArrayValue(s.SymbolNumber(txt))
End Function

Function ler_produzidas(linha As String, maquina As String) As Long
Set M = ThisDocument.Model
Set s = M.SIMAN
 txt = "Maquina_" + linha + "_" + maquina + ".Processa.NumberOut"
 ler_produzidas = s.VariableArrayValue(s.SymbolNumber(txt))
End Function

Sub remover_das_filas(linha As String, maquina As String, fila As String, qty As String)
Dim b, z, i As Long
Set M = ThisDocument.Model
Set s = M.SIMAN
txt = "Maquina" + "_" + linha + "_" + maquina + "." + fila
i = M.Queues.Find(smFindName, txt, 1)
For i = 1 To M.Queues.Count
    If txt = s.ConstructString(smSimanConstructQueues, i, 1) Then
      Exit For
    End If
Next
For b = 1 To CLng(qty)
  z = s.QueueFirstEntity(i)
  s.QueueRemoveEntity z, i
Next
End Sub

Function recurso_livre(linha As String, maquina As String) As Boolean
 txt = "Maquina" + "_" + linha + "_" + maquina + " Recurso"
 nMaxResources = s.ResourcesMaximum
 For i = 1 To nMaxResources
   If txt = s.ConstructString(smSimanConstructResources, i, 1) Then
     Exit For
   End If
 Next i
 recurso_livre = s.ResourceNumberBusy(i)
End Function

Function local_operador(lin As String) As Long
' devolve a maquina em que o operador está
Dim k As Long
For k = 1 To maqlinha(CLng(lin))
  ta_aqui = ler_fila(lin, CStr(k), "Fila Operador")
  If ta_aqui = -1 Then Exit For
  If ta_aqui <> 0 Then
    local_operador = k
    Exit For
  End If
Next k
End Function

Sub atribui_pedidos(linha As Long, maq As Long)
Dim rs As DAO.Recordset
Dim bd As DAO.Database

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_planned_orders")
qtd = 0
maquina = "Machine_" + Trim(Str(linha) + "_" + Trim(Str(maq)))
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
```

```
End If
While Not rs.EOF
  rs.Edit
  If Trim(rs.Fields("machine")) = maquina Then
    qtd = rs.Fields("qty")
    produto = rs.Fields("product")
    lote(linha, maq) = rs.Fields("lot_det")
    rs.MoveLast
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
If qtd > 0 Then
  tx1 = "Maquina_" + CStr(linha) + "_" + CStr(maq)
  tx2 = "fio_" + Trim(CStr(tipos_produto(produto, 2)))
  tx3 = "produto_" + Trim(CStr(tipos_produto(produto, 5)))
  tx4 = "palete_" + Trim(CStr(tipos_produto(produto, 4)))
  tx5 = "Machine_" + CStr(linha) + "_" + CStr(maq)
  Call insere_nas_filas(CStr(tx1), CLng(qtd), CStr(tx2), CStr(tx3), CStr(tx4))
  Call declara_materiais(CStr(tipos_produto(produto, 0)), CLng(qtd))
  Call escreve_variavel(CStr(linha), CStr(maq), "TipoProduto", CStr(produto))
  Call escreve_variavel(CStr(linha), CStr(maq), "Imagemfio", CStr(tipos_produto(produto, 2)))
  Call escreve_variavel(CStr(linha), CStr(maq), "Imagempalete", CStr(tipos_produto(produto, 4)))
  Call escreve_variavel(CStr(linha), CStr(maq), "Imagemproduto", CStr(tipos_produto(produto, 5)))
  Call actualiza_tempo_processamento(CLng(linha), CLng(maq))
End If
End Sub

Sub nao_usada_valor_setupmaq()
'guardar os valores iniciais da variavel TempoSetup
For i = 1 To UBound(linhasactivas, 1)
    maq = maqlinha(i)
    For j = 1 To maq
      setupmaq(linhasactivas(i), j) = ler_variavel(CStr(linhasactivas(i)), CStr(j), "TempoSetup")
      Call escreve_variavel(CStr(linhasactivas(i)), CStr(j), "TempoSetup", "1") ' para poder trabalhar
sem a presença do operador
      TempoRota(linhasactivas(i), j) = ler_variavel(CStr(linhasactivas(i)), CStr(j), "TempRotaMaq")
      velocidade(linhasactivas(i), j) = ler_variavel(CStr(linhasactivas(i)), CStr(j), "TempoProc")
    Next j
Next i
End Sub

Function TempoCaminho(linha As Long, maqactual As Long, proxima As Long)
If maqactual > proxima Then
  aux = proxima
  proxima = maqactual
  maqactual = aux
End If
t = 0
For k = maqactual To proxima - 1
  t = t + TempoRota(k)
Next k
TempoCaminho = t
End Function

Sub envia_operador(linhactual As String, maqactual As String, maq As String)
'envia o operador da maqactual para a maq
sinal = 0
Call escreve_variavel(linhactual, maqactual, "TempRotaMaq", CStr(dist(1, 1)))
Call escreve_variavel(linhactual, maqactual, "prox", CStr(maq))
sinal = devolve_sinal(CLng(linhactual), CLng(maqactual))
s.SignalSend (sinal)
operador_livre(CLng(linhactual), 0) = CLng(maq)
operador_livre(CLng(linhactual), 1) = s.RunCurrentTime + opr_time(CLng(linhactual), 1) + dist(1, 1)
End Sub

Sub envia_opr_fica_na_mesma(linhactual As String, maqactual As String)
sinal = 0
Call escreve_variavel(linhactual, maqactual, "TempRotaMaq", "0")
Call escreve_variavel(linhactual, maqactual, "prox", CStr(maqactual))
sinal = devolve_sinal(CLng(linhactual), CLng(maqactual))
s.SignalSend (sinal)
End Sub

Function ler_fila(linha As String, maquina As String, fila As String) As Long
Set M = ThisDocument.Model
Set s = M.SIMAN
txt = "Maquina" + "_" + linha + "_" + maquina + "." + fila
ler_fila = s.QueueNumberOfEntities(s.SymbolNumber(txt))
End Function

Function escreve_fila(linha As String, maquina As String, fila As String, valor As String) As Long
Set M = ThisDocument.Model
Set s = M.SIMAN
txt = "Maquina" + "_" + linha + "_" + maquina + "." + fila
's.QueueNumberOfEntities(s.SymbolNumber(txt)) = CStr(valor)
End Function

Function ordena_larger(ParamArray lista())
Dim aux, pos, lim As Integer
Dim i As Long
lim = lista(1)
While lim > 1
  pos = 0
  For i = 1 To lim - 1
    If lista(0)(i, 1) < lista(0)(i + 1, 1) Then
      If lista(2) Then
        Call trocar_posicoes_man(i, i + 1)
       Else
        Call trocar_posicoes(i, i + 1)
      End If
      pos = i
    End If
    lim = pos
  Next i
```

```
Wend
empate = 1
For i = 1 To lista(1) - 1
  If lista(0)(i, 1) = lista(0)(i + 1, 1) Then
    empate = empate + 1 ' existe empate
    Else
    Exit For
  End If
Next
If empate = 1 Then
  lista(0)(0, 1) = 1
  empate = 0
End If
lista(0)(0, 0) = empate
End Function

Function terminou_a_simulacao() As Boolean
Dim x, z As Long

terminou_a_simulacao = True
x = 0
z = 0
For linha = 1 To UBound(linhasactivas, 1)
  For maq = 1 To maqlinha(linha)
    If rejeitada(linha, maq) = -1 Then
    x = x + 1
    Else
      x = -1
      terminou_a_simulacao = False
      Exit For
     End If
  Next maq
  If x = -1 Then Exit For
  If x = maqlinha(linha) Then z = z + 1
Next linha
If z = UBound(linhasactivas, 1) Then terminou_a_simulacao = True
End Function

Function ordena_smaller(ParamArray lista())
Dim aux, pos, lim As Integer
Dim i As Long
lim = lista(1)
n = lista(1)
While lim > 1
  pos = 0
  For i = 1 To lim - 1
    If lista(0)(i, 1) > lista(0)(i + 1, 1) Then
      If lista(2) Then
        Call trocar_posicoes_man(i, i + 1)
       Else
        Call trocar_posicoes(i, i + 1)
      End If
      pos = i
    End If
    lim = pos
  Next i
Wend
empate = 1
For i = 1 To lista(1) - 1
  If lista(0)(i, 1) = lista(0)(i + 1, 1) Then
    empate = empate + 1 ' existe empate
    Else
    Exit For
  End If
Next
If empate = 1 Then
  lista(0)(0, 1) = 1
  empate = 0
End If
lista(0)(0, 0) = empate
End Function

Function trocar_posicoes(x As Long, y As Long)
Dim aux As Long
aux = veloc(x, 0): veloc(x, 0) = veloc(y, 0): veloc(y, 0) = aux
aux = veloc(x, 1): veloc(x, 1) = veloc(y, 1): veloc(y, 1) = aux
aux = tproc(x, 0): tproc(x, 0) = tproc(y, 0): tproc(y, 0) = aux
aux = tproc(x, 1): tproc(x, 1) = tproc(y, 1): tproc(y, 1) = aux
aux = dist(x, 0): dist(x, 0) = dist(y, 0): dist(y, 0) = aux
aux = dist(x, 1): dist(x, 1) = dist(y, 1): dist(y, 1) = aux
aux = qtfila(x, 0): qtfila(x, 0) = qtfila(y, 0): qtfila(y, 0) = aux
aux = qtfila(x, 1): qtfila(x, 1) = qtfila(y, 1): qtfila(y, 1) = aux
End Function

Sub ler_criterios_bd(linha As Long)
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim sql As String
Dim encontrei As Boolean
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_opr_table")
encontrei = False
ReDim criterio(4, 1)
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
While Not rs.EOF And Not encontrei
  If rs.Fields("line") = CLng(linha) Then
    If (rs.Fields("first")) <> "" Then criterio(1, 0) = rs.Fields("first"): criterio(1, 1) =
rs.Fields("first_v")
    If (rs.Fields("second")) <> "" Then criterio(2, 0) = rs.Fields("second"): criterio(2, 1) =
rs.Fields("second_v")
    If (rs.Fields("third")) <> "" Then criterio(3, 0) = rs.Fields("third"): criterio(3, 1) =
rs.Fields("third_v")
    If (rs.Fields("fourth")) <> "" Then criterio(4, 0) = rs.Fields("fourth"): criterio(4, 1) =
```

```
rs.Fields("fourth_v")
    encontrei = True
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
End Sub

Sub escrever_na_bd()
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim sql As String
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_opr_table")
linha = 3
With rs
  rs.index = "PrimaryKey"
  If Not rs.EOF Then
    rs.MoveLast
    rs.MoveFirst
  End If
  rs.Seek "=", linha
   If Not .NoMatch Then
      rs.Edit
      !first = "X"
      !first_v = 5
      rs.Update
     End If
   End With
rs.Close
bd.Close
End Sub

Sub ler_pedidos()
Dim rs As DAO.Recordset
Dim bd As DAO.Database
For k = 1 To UBound(tipos_produto, 1)
tipos_produto(k, 1) = 0
Next k
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_manufacturing")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
While Not rs.EOF
  For k = 1 To UBound(tipos_produto, 1)
    If Trim(rs.Fields("cod")) = tipos_produto(k, 0) Then
      tipos_produto(k, 1) = rs.Fields("qty_inicial") - rs.Fields("qty_prod") + tipos_produto(k, 1)
      Exit For
    End If
  Next k
  rs.MoveNext
Wend
rs.Close
bd.Close
End Sub

Sub coloca_maquina_apta_para_trabalhar(linha As Long, maquina As Long)
Call escreve_variavel(CStr(linha), CStr(maquina), "TempoSetup", "1")
End Sub

Sub inicializacao()
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim linhas_usadas(40)
Dim nMaxResources As Long
Dim nIndex As Long

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_machine_line")

If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
j = 0
i = rs.Fields("line")
While Not rs.EOF
  If rs.Fields("line") <> linhas_usadas(rs.Fields("line")) Then
    linhas_usadas(rs.Fields("line")) = rs.Fields("line")
    j = j + 1
  End If
  If rs.Fields("line") > i Then
    i = rs.Fields("line")              ' ultima linha a ser usada para dimensinar opr_time
  End If
  rs.MoveNext
Wend
rs.Close
ReDim linhasactivas(j)
ReDim operador_livre(i, 1)
ReDim opr_time(i, 1)
For k = 1 To i
  operador_livre(k, 0) = 0  ' livre
  operador_livre(k, 1) = 0  ' tempo
Next k
Set rs = bd.OpenRecordset("DEF_product_table")
n_prod = rs.RecordCount
ReDim tipos_produto(n_prod, 5)
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
For k = 1 To n_prod
  tipos_produto(k, 0) = Trim(rs.Fields("cod"))
```

```
  tipos_produto(k, 2) = rs.Fields("wire_image")
  tipos_produto(k, 3) = rs.Fields("time")
  tipos_produto(k, 4) = rs.Fields("batch_image")
  tipos_produto(k, 5) = rs.Fields("prod_image")
  rs.MoveNext
Next k
rs.Close

x = 0
For i = 1 To 40
  If linhas_usadas(i) > 0 Then
    x = x + 1
    linhasactivas(x) = i
    If x = j Then Exit For
  End If
Next i

Set rs = bd.OpenRecordset("DEF_types_table")
n_maqs = rs.RecordCount
ReDim setupmaq(n_maqs)
ReDim TempoRota(n_maqs)
ReDim velocidade(n_maqs)
ReDim maq_func_avaria(n_maqs)

If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
For i = 1 To n_maqs
  setupmaq(rs.Fields("Machine_cod")) = rs.Fields("Setup_time")
  TempoRota(rs.Fields("Machine_cod")) = rs.Fields("Route_time")
  velocidade(rs.Fields("Machine_cod")) = rs.Fields("Proc_time")
  qty_bundle = rs.Fields("qtd_batch")
  maq_func_avaria(rs.Fields("Machine_cod")) = rs.Fields("Fail_time")
  rs.MoveNext
Next i
rs.Close
Set rs = bd.OpenRecordset("DEF_line_table")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
For i = 1 To UBound(linhasactivas, 1)
  opr_time(linhasactivas(i), 0) = linhasactivas(i)
  opr_time(linhasactivas(i), 1) = tempo_operador_da_linha(rs.Fields("line"))
  rs.MoveNext
Next
rs.Close
bd.Close

ReDim maqlinha(linhasactivas(UBound(linhasactivas, 1)))
For i = 1 To UBound(linhasactivas, 1)
maqlinha(linhasactivas(i)) = num_maquinas_linha(CStr(linhasactivas(i)))
Next i

Set a = GetObject("", "Arena.Application")
Set M = a.ActiveModel
Set s = M.SIMAN
With s
  nMaxResources = s.ResourcesMaximum
  ReDim estado_maquinas_old(linhasactivas(UBound(linhasactivas, 1)), nMaxResources)
  ReDim estado_maquinas(linhasactivas(UBound(linhasactivas, 1)), nMaxResources)
  For nIndex = 1 To nMaxResources
    recurso = .ConstructString(smSimanConstructResources, nIndex, 1)
    maq = Val(Mid$(recurso, 11, 2))
    lin = Val(Mid$(recurso, 9, 2))
    estado_maquinas_old(lin,      maq)     =     .ConstructString(smSimanConstructStateSets,     nIndex,
.ResourceState(nIndex))
    estado_maquinas(lin,      maq)      =      .ConstructString(smSimanConstructStateSets,      nIndex,
.ResourceState(nIndex))
  Next nIndex
End With
ReDim lote(linhasactivas(UBound(linhasactivas, 1)), nMaxResources)
ReDim rejeitada(linhasactivas(UBound(linhasactivas, 1)), nMaxResources)
ReDim Ocupacao_opr(linhasactivas(UBound(linhasactivas, 1)))

For i = 1 To UBound(linhasactivas, 1)
  For k = 1 To maqlinha(linhasactivas(i))
    rejeitada(linhasactivas(i), k) = 0

    Call escreve_variavel(CStr(linhasactivas(i)), CStr(k), "Quantidade", CLng(qty_bundle))

  Next k
Next i

manutencao(0) = 0  'linha
manutencao(1) = 0  'maq
manutencao(2) = -1 'tempo inicio
manutencao(3) = -1 'tempo fim

End Sub

Sub operador_control(linha As String, maq_actual As String, maq_seguinte As String, distancia As Long,
work As Long, comments As String)
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim sql As String
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("TRN_operator_maq")

If rs.RecordCount Then rs.MoveLast

With rs
  rs.AddNew
  !Time = s.RunCurrentTime
  !line = linha
```

```
  !from_machine = maq_actual
  !next_machine = maq_seguinte
  !Distance = distancia
  !working_until = work
  !obs = comments
  rs.Update
End With
rs.Close
bd.Close
End Sub

Sub actualiza_tempo_processamento(linhas As Long, maquina As Long)
Dim vl, tp, time_processa As Long
vl = velocidade(ler_variavel(CStr(linhas), CStr(maquina), "TipoMaquina"))
tp = tipos_produto(ler_variavel(CStr(linhas), CStr(maquina), "TipoProduto"), 3)
time_processa = Int(tp / vl)
Call escreve_variavel(CStr(linhas), CStr(maquina), "TempoProc", CStr(time_processa))
End Sub

Sub atribuicao_inicial_de_pedidos()
Dim tempo As Long
Dim opmaq, i, encontrei As Long
Dim machine As String
Set M = ThisDocument.Model
Set s = M.SIMAN

For linhas = 1 To UBound(linhasactivas, 1)
 For maq = 1 To maqlinha(linhasactivas(linhas))
     Call atribui_pedidos(CStr(linhasactivas(linhas)), CStr(maq))
 Next maq
Next linhas

'escolhe a 1ª maquina para o operador
   For linhas = 1 To UBound(linhasactivas, 1)
    encontrei = 0
    sucesso = 1
    iteracao = 1
    maqs_paradas = maquinas_paradas(CLng(linhasactivas(linhas)))
    Call ler_criterios_bd(CLng(linhasactivas(linhas)))
    While sucesso <> 0
      Select Case criterio(iteracao, 0)
      Case "Distance"
        If criterio(iteracao, 1) = 0 Then ordena_larger dist, maqs_paradas, 0
        If criterio(iteracao, 1) = 1 Then ordena_smaller dist, maqs_paradas, 0
        sucesso = dist(0, 0)
      Case "Speed"
        If criterio(iteracao, 1) = 0 Then ordena_larger veloc, maqs_paradas, 0
        If criterio(iteracao, 1) = 1 Then ordena_smaller veloc, maqs_paradas, 0
        sucesso = veloc(0, 0)
      Case "Queue"
        If criterio(iteracao, 1) = 0 Then ordena_larger qtfila, maqs_paradas, 0
        If criterio(iteracao, 1) = 1 Then ordena_smaller qtfila, maqs_paradas, 0
        sucesso = qtfila(0, 0)
      Case "Process Time"
        If criterio(iteracao, 1) = 0 Then ordena_larger tproc, maqs_paradas, 0
        If criterio(iteracao, 1) = 1 Then ordena_smaller tproc, maqs_paradas, 0
        sucesso = tproc(0, 0)
      Case Else ' se não existir criterio: escolhe a maquina que se situa mais perto
        ordena_smaller dist, maqs_paradas
        sucesso = 0
      End Select
      iteracao = iteracao + 1
      maqs_paradas = sucesso
    Wend
    encontrei = 1
    machine = dist(1, 0)
  If encontrei Then
    Call escreve_variavel(CStr(linhasactivas(linhas)), "1", "prox", CStr(machine))
    sinal = devolve_sinal(CLng(linhasactivas(linhas)), CLng(machine))
    s.SignalSend (sinal)
    operador_livre(CLng(CLng(linhasactivas(linhas))), 0) = CLng(machine)
    txt = "TempRotaOp_" + CStr(linhasactivas(linhas))
    tempo = s.VariableArrayValue(s.SymbolNumber(txt))
    operador_livre(CLng(CLng(linhasactivas(linhas))),  1)  =  opr_time(CLng(linhasactivas(linhas)), 1) +
tempo
    Call    operador_control(CStr(linhasactivas(linhas)),    "Start",    CStr(machine),    CLng(tempo),
CLng(operador_livre(linhasactivas(linhas), 1)), "Move Operator")
  End If
Next linhas
End Sub

Function tempo_operador_da_linha(linha As Long)
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim encontrei As Boolean
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_line_table")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
encontrei = False
While (Not rs.EOF And encontrei = False)
  If rs.Fields("line") = linha Then
  tempo_operador_da_linha = rs.Fields("opr_time")
  encontrei = True
  End If
rs.MoveNext
Wend
rs.Close
bd.Close
End Function

Sub apaga_tabelas_temporarias()
Dim rs As DAO.Recordset
Dim bd As DAO.Database
```

```
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("TRN_operator_maq")
While Not rs.EOF
  rs.Delete
  rs.MoveNext
Wend
rs.Close
Set rs = bd.OpenRecordset("TRN_maintenance")
While Not rs.EOF
  rs.Delete
  rs.MoveNext
Wend
rs.Close
Set rs = bd.OpenRecordset("TRN_mrp")
While Not rs.EOF
  rs.Delete
  rs.MoveNext
Wend
rs.Close
Set rs = bd.OpenRecordset("TRN_warehouse")
While Not rs.EOF
  rs.Delete
  rs.MoveNext
Wend
rs.Close
Set rs = bd.OpenRecordset("TRN_machine")
While Not rs.EOF
  rs.Delete
  rs.MoveNext
Wend
rs.Close
bd.Close
End Sub

Sub apaga_ordem_declarada(nlot As Long)
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_planned_orders")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
While Not rs.EOF
If rs.Fields("lot_det") = nlot Then
  rs.Delete
  rs.MoveLast
  Exit Sub
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
End Sub
```

```
Sub actualiza_qty_prod(ordem As String, qty As Long)
Dim rs As DAO.Recordset
Dim bd As DAO.Database

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_orders")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
conta = 0
While Not rs.EOF
If Trim(rs.Fields("n_order")) = Trim(ordem) Then
  rs.Edit
  rs.Fields("qty_prod") = CLng(qty)
  rs.Update
  rs.MoveLast
End If
rs.MoveNext
Wend
rs.Close
bd.Close
End Sub

Sub declaracao_producao(cod As String, qty As Long, nlote As Long)
Dim rs As DAO.Recordset
Dim bd As DAO.Database

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_orders")
If (qty) <= 0 Then
  Exit Sub
End If
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If

qty_aux = qty
qty_bd_aux = 0
qty_bd_falta_aux = 0
n_order = devolve_num_ordem(CStr(cod), CLng(qty))
n_order = devolve_num_ordem(CStr(cod), CLng(qty))
While Not rs.EOF And qty_aux > 0
  If rs.Fields("n_order") = n_order Then
    If IsNull(rs.Fields("qty_prod")) Then
      qty_bd_aux = 0
      qty_bd_falta_aux = CLng(rs.Fields("qty_inicial"))
    Else
     qty_bd_aux = CLng(rs.Fields("qty_prod"))
     qty_bd_falta_aux = CLng(rs.Fields("qty_inicial")) - qty_bd_aux
    End If
```

```
    If qty_aux <= qty_bd_falta_aux Then
      Call actualiza_qty_prod(CStr(rs.Fields("n_order")), CLng(qty_aux + qty_bd_aux))
      qty_aux = 0
    Else
     Call actualiza_qty_prod(CStr(rs.Fields("n_order")), CLng(qty_bd_aux + qty_bd_falta_aux))
     qty_aux = qty_aux - qty_bd_falta_aux
     n_order = devolve_num_ordem(CStr(cod), CLng(qty))
    End If
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
End Sub

Sub declara_materiais(cod As String, qty As Long)
Dim rs, rrs As DAO.Recordset
Dim bd As DAO.Database

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_bom")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
While Not rs.EOF
  If Trim(rs.Fields("bom_cod")) = cod Then
    rs.Edit
    material = rs.Fields("component")
    qtd = rs.Fields("qty")
    Set rrs = bd.OpenRecordset("TRN_mrp")
    With rrs
      rrs.AddNew
      !Time = s.RunCurrentTime
      !product = cod
      !component = material
      !qty = qtd * qty
      rrs.Update
    End With
    Call declara_transfer_component(CStr(cod), CStr(material), CLng(qtd * qty))
    Call declara_receipts_component(CStr(cod), CStr(material), CLng(qtd * qty))
    rrs.Close
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
End Sub

Function devolve_num_ordem(cod As String, qty As Long) As Long
Dim rs As DAO.Recordset
Dim bd As DAO.Database

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_orders")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
While Not rs.EOF
If rs.Fields("qty_inicial") > rs.Fields("qty_prod") And Trim(rs.Fields("cod")) = cod Then
  n_order = rs.Fields("n_order")
  rs.MoveLast
 End If
 rs.MoveNext
Wend
rs.Close
bd.Close
devolve_num_ordem = n_order
End Function

Sub declara_transfer_component(prod As String, component As String, qty As Long)
Dim tipo As String
Dim rs As DAO.Recordset
Dim bd As DAO.Database

tipo = "Transfer"
If CLng(qty) <= 0 Then
  Exit Sub
End If

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("TRN_warehouse ")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
transaction = CStr(devolve_nova_recepcao())
  rs.AddNew
  rs.Fields("trnbr") = CLng(transaction)
  rs.Fields("qty") = CLng(qty)
  rs.Fields("type") = tipo
  rs.Fields("component") = component
  rs.Fields("time") = s.RunCurrentTime
  rs.Fields("product") = CStr(prod)
  rs.Update
rs.Close
bd.Close
End Sub

Sub declara_receipts_component(prod As String, component As String, qty As Long)
Dim tipo As String
Dim rs As DAO.Recordset
Dim bd As DAO.Database

tipo = "Reception"
```

```
If CLng(qty) <= 0 Then
  Exit Sub
End If

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("TRN_warehouse ")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If

trnbr = CStr(devolve_nova_recepcao())
  rs.AddNew
  rs.Fields("trnbr") = CLng(trnbr)
  rs.Fields("qty") = CLng(qty)
  rs.Fields("type") = tipo
  rs.Fields("component") = component
  rs.Fields("time") = s.RunCurrentTime
  rs.Fields("product") = CStr(prod)
  rs.Update
rs.Close
bd.Close
End Sub

Function devolve_nova_recepcao()
Dim rs As DAO.Recordset
Dim bd As DAO.Database

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("TRN_warehouse ")

If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If

If Not rs.EOF Then
rs.MoveLast

  If Not IsNull(rs.Fields("trnbr")) Then
    devolve_nova_recepcao = CLng(rs.Fields("trnbr")) + 1
   Else
     devolve_nova_recepcao = 1
  End If
 Else
   devolve_nova_recepcao = 1
End If
rs.Close
bd.Close
End Function

Function existe_planned_orders() As Boolean
Dim rs As DAO.Recordset
Dim bd As DAO.Database

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_planned_orders")

If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If

If Not rs.EOF Then
rs.MoveLast
  If Not IsNull(rs.Fields("machine")) Then
    existe_planned_orders = True
   Else
     existe_planned_orders = False
  End If
 Else
  existe_planned_orders = False
End If
rs.Close
bd.Close
End Function

Function tem_orders(maquina As String) As Boolean
Dim rs As DAO.Recordset
Dim bd As DAO.Database

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_planned_orders")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
tem_orders = False
While Not rs.EOF
If rs.Fields("machine") = maquina Then
  tem_orders = True
  rs.MoveLast
 End If
 rs.MoveNext
Wend
rs.Close
bd.Close
End Function

Function maquinas_paradas(linhas As Long)
Dim maq As Long
n_maqs = maqlinha(linhas)
ReDim dist(n_maqs, 1)
ReDim qtfila(n_maqs, 1)
ReDim veloc(n_maqs, 1)
ReDim tproc(n_maqs, 1)
```

```
maqs_paradas = 0
' verifica se existem maquinas paradas
For maq = 1 To maqlinha(linhas)
  opmaq = local_operador(CStr(linhas))
  'estado = recurso_livre(CStr(linhas), CStr(maq))
  'maquina = "Machine_" + CStr(linhas) + "_" + CStr(maq)
  'activada = ler_variavel(CStr(linhas), CStr(maq), "TempoSetup")

  If estado_maquinas(linhas, maq) = "Parada" And maq <> opmaq And rejeitada(CLng(linhas), CLng(maq)) <>
-1 Then
  'End If
  'If estado = False And tem_orders(CStr(maquina)) And activada <> 1 Then
    'If rejeitada(CLng(linhas), CLng(maq)) <> -1 Then
      maqs_paradas = maqs_paradas + 1
     ' opmaq = local_operador(CStr(linhas))
      Call aceita_maquina(CLng(maqs_paradas), CLng(maq), CLng(linhas), CLng(opmaq))
    'End If
  End If
Next maq
maquinas_paradas = maqs_paradas
End Function

Function ler_pedidos_em_espera(linha As String, maquina As String)
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim txt As String
Dim qty As Long

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_planned_orders")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
txt = "Machine_" + linha + "_" + maquina
qty = 0
While Not rs.EOF
  If Trim(rs.Fields("machine")) = txt Then
    qty = rs.Fields("qty") + qty
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
ler_pedidos_em_espera = qty
End Function

Function num_maquinas_linha(linha As String)
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim txt As String
Dim qty As Long

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_machine_line")
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
qty = 0
While Not rs.EOF
  If Trim(rs.Fields("line")) = linha Then
    qty = qty + 1
  End If
  rs.MoveNext
Wend
rs.Close
bd.Close
num_maquinas_linha = qty
End Function

Sub Estados_Maquinas()
Dim nMaxResources As Long
Dim nIndex, i As Long

Set a = GetObject("", "Arena.Application")
Set M = a.ActiveModel
Set s = M.SIMAN
With s
  nMaxResources = s.ResourcesMaximum
  For nIndex = 1 To nMaxResources
    recurso = .ConstructString(smSimanConstructResources, nIndex, 1)
    maq = Val(Mid$(recurso, 11, 2))
    lin = Val(Mid$(recurso, 9, 2))
    estado_maquinas(lin,      maq)      =      .ConstructString(smSimanConstructStateSets,      nIndex,
.ResourceState(nIndex))
  Next nIndex
End With
End Sub

Sub maquina_control(linhas As Long, maq As Long, comments As String, prod As String)
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim sql As String
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("TRN_machine")
With rs
  rs.AddNew
  !Time = s.RunCurrentTime
  !line = linhas
  !machine = maq
  !product = prod
  !obs = comments
  rs.Update
End With
```

```
rs.Close
bd.Close
End Sub

Function devolve_sinal(linha As Long, maq As Long)

sinal = 0
If linha = linhasactivas(1) Then
  sinal = sinal + maq
 Else
   For i = 1 To UBound(linhasactivas, 1) - 1
     If linhasactivas(i) = linha Then Exit For
     sinal = sinal + maqlinha(linhasactivas(i))
   Next i
   sinal = sinal + maq
End If
devolve_sinal = sinal
End Function

Sub guarda_informacao_maquina(linhas As Long, maqx As Long)
End Sub

Sub ins_fila_operador(maquina As String)
Dim i, j As Long
fila = maquina + ".Fila Operador"
For i = 1 To M.Queues.Count
  If fila = s.ConstructString(smSimanConstructQueues, i, 1) Then
    j = s.EntityCreate
    s.EntityInsertIntoQueue j, i
    Exit For
  End If
Next
End Sub

Sub alteracao_estados_maquinas(linhas As Long)
Dim maqx As Long
Call Estados_Maquinas
For maqx = 1 To maqlinha(linhasactivas(linhas))
  If  estado_maquinas(linhasactivas(linhas),  maqx)  <>  estado_maquinas_old(linhasactivas(linhas),  maqx)
Then
    cod = tipos_produto(ler_variavel(CStr(linhasactivas(linhas)), CStr(maqx), "TipoProduto"), 0)
    Call                    maquina_control(CLng(linhasactivas(linhas)),                    CLng(maqx),
CStr(estado_maquinas(linhasactivas(linhas), maqx)), CStr(cod))
    estado_maquinas_old(linhasactivas(linhas), maqx) = estado_maquinas(linhasactivas(linhas), maqx)
    If estado_maquinas(linhasactivas(linhas), maqx) = "Parada" Then
      Call escreve_variavel(CStr(linhasactivas(linhas)), CStr(maqx), "TempoSetup", "0")
    End If
  End If
Next maqx
End Sub

Sub  manutencao_control(inicio  As String, fim  As String, linha  As String, maq  As String, comments  As
String)

Dim rs As DAO.Recordset
Dim bd As DAO.Database
Dim sql As String
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("TRN_maintenance")
With rs
  rs.AddNew
  !begin = inicio
  !End = fim
  !machine = maq
  !line = linha
  !status = comments
  rs.Update
End With
rs.Close
bd.Close
End Sub

Function maquinas_avariadas()
n_maqs = s.ResourcesMaximum
ReDim dist(n_maqs, 1)
ReDim qtfila(n_maqs, 1)
ReDim veloc(n_maqs, 1)
ReDim tproc(n_maqs, 1)
ReDim linhaman(n_maqs, 1)
maqs_avariadas = 0
' verifica se existem maquinas avariadas
For linha = 1 To UBound(linhasactivas, 1)
 For maq = 1 To maqlinha(linhasactivas(linha))
    If rejeitada(CLng(linhasactivas(linha)), CLng(maq)) <> -1 And estado_maquinas(linhasactivas(linha),
maq) = "Avariada" Then
      maqs_avariadas = maqs_avariadas + 1
      Call      aceita_maquina_man(CLng(maqs_avariadas),      CLng(maq),      CLng(linhasactivas(linha)),
CLng(manutencao(0)), CLng(manutencao(1)))
      Call escreve_variavel(CStr(linhasactivas(linha)), CStr(maq), "TempoSetup", "0")
    End If
  Next maq
Next linha
maquinas_avariadas = maqs_avariadas
End Function

Function trocar_posicoes_man(x As Long, y As Long)
Dim aux As Long
aux = veloc(x, 0): veloc(x, 0) = veloc(y, 0): veloc(y, 0) = aux
aux = veloc(x, 1): veloc(x, 1) = veloc(y, 1): veloc(y, 1) = aux
aux = tproc(x, 0): tproc(x, 0) = tproc(y, 0): tproc(y, 0) = aux
aux = tproc(x, 1): tproc(x, 1) = tproc(y, 1): tproc(y, 1) = aux
aux = dist(x, 0): dist(x, 0) = dist(y, 0): dist(y, 0) = aux
aux = dist(x, 1): dist(x, 1) = dist(y, 1): dist(y, 1) = aux
aux = qtfila(x, 0): qtfila(x, 0) = qtfila(y, 0): qtfila(y, 0) = aux
aux = qtfila(x, 1): qtfila(x, 1) = qtfila(y, 1): qtfila(y, 1) = aux
aux = linhaman(x, 0): linhaman(x, 0) = linhaman(y, 0): linhaman(y, 0) = aux
aux = linhaman(x, 1): linhaman(x, 1) = linhaman(y, 1): linhaman(y, 1) = aux
```

```
End Function

Sub ler_criterios_bd_man()
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_man_table")
ReDim criterio(5, 1)
If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
While Not rs.EOF
    If (rs.Fields("first")) <> "" Then criterio(1, 0) = rs.Fields("first"): criterio(1, 1) =
rs.Fields("first_v")
    If (rs.Fields("second")) <> "" Then criterio(2, 0) = rs.Fields("second"): criterio(2, 1) =
rs.Fields("second_v")
    If (rs.Fields("third")) <> "" Then criterio(3, 0) = rs.Fields("third"): criterio(3, 1) =
rs.Fields("third_v")
    If (rs.Fields("fourth")) <> "" Then criterio(4, 0) = rs.Fields("fourth"): criterio(4, 1) =
rs.Fields("fourth_v")
    If (rs.Fields("fifth")) <> "" Then criterio(5, 0) = rs.Fields("fifth"): criterio(1, 1) =
rs.Fields("fifth_v")
  rs.MoveNext
Wend
rs.Close
bd.Close
End Sub

Sub aceita_maquina_man(num As Long, maq_destino As Long, linha_destino As Long, linha_actual As Long,
maq_actual As Long)
  dist(num, 0) = maq_destino
  qtfila(num, 0) = maq_destino
  veloc(num, 0) = maq_destino
  tproc(num, 0) = maq_destino

  linhaman(num, 0) = linha_destino

  dist(num, 1) = TempoCaminho_man(CLng(linha_actual), CLng(linha_destino), CLng(maq_actual),
CLng(maq_destino))
  qtfila(num, 1) = ler_pedidos_em_espera(CStr(linha_destino), CStr(maq_destino))
  veloc(num, 1) = velocidade(ler_variavel(CStr(linha_destino), CStr(maq_destino), "TipoMaquina"))
  tproc(num, 1) = ler_variavel(CStr(linha_destino), CStr(maq_destino), "TempoProc")

  linhaman(num, 1) = linha_destino
End Sub

Function TempoCaminho_man(linha_a As Long, linha_d As Long, maqactual As Long, proxima As Long)
If maqactual > proxima Then
  aux = proxima
  proxima = maqactual
  maqactual = aux
End If
If linha_a > linha_d Then
  aux = linha_a
  linha_d = maqactual
  linha_a = aux
End If
t = 0
For k = maqactual To proxima - 1
  t = t + TempoRota(k)
Next k
For k = linha_a To linha_d - 1
  t = t + TempoRota(k)
Next k

TempoCaminho_man = t
End Function

Sub gestao_manutencao()
'********************manutenção************************************************************
  maqs_avariadas = maquinas_avariadas()
  If maqs_avariadas > 0 And manutencao(0) = 0 Then
    sucesso = 1
    iteracao = 1
    Call ler_criterios_bd_man
    While (sucesso <> 0 And iteracao < 6) ' 5 criterios
      Select Case criterio(iteracao, 0)
      Case "Line"
        If criterio(iteracao, 1) = 0 Then ordena_larger linhaman, maqs_avariadas, 1
        If criterio(iteracao, 1) = 1 Then ordena_smaller linhaman, maqs_avariadas, 1
        sucesso = linhaman(0, 0)
        seguinte = dist(1, 0)
      Case "Distance"
        If criterio(iteracao, 1) = 0 Then ordena_larger dist, maqs_avariadas, 1
        If criterio(iteracao, 1) = 1 Then ordena_smaller dist, maqs_avariadas, 1
        sucesso = dist(0, 0)
        seguinte = dist(1, 0)
      Case "Speed"
        If criterio(iteracao, 1) = 0 Then ordena_larger veloc, maqs_avariadas, 1
        If criterio(iteracao, 1) = 1 Then ordena_smaller veloc, maqs_avariadas, 1
        sucesso = veloc(0, 0)
        seguinte = veloc(1, 0)
      Case "Queue"
        If criterio(iteracao, 1) = 0 Then ordena_larger qtfila, maqs_avariadas, 1
        If criterio(iteracao, 1) = 1 Then ordena_smaller qtfila, maqs_avariadas, 1
        sucesso = qtfila(0, 0)
        seguinte = qtfila(1, 0)
      Case "Process Time"
        If criterio(iteracao, 1) = 0 Then ordena_larger tproc, maqs_avariadas, 1
        If criterio(iteracao, 1) = 1 Then ordena_smaller tproc, maqs_avariadas, 1
        sucesso = tproc(0, 0)
        seguinte = tproc(1, 0)
      Case Else ' se não existir criterio: escolhe a maquina mais perto
        ordena_smaller dist, maqs_avariadas
        sucesso = 0
```

```
        seguinte = dist(1, 0)
      End Select
      iteracao = iteracao + 1
      maqs_avariadas = sucesso
    Wend
    encontrei = 1
    machine = seguinte

  If estado_maquinas(linhaman(1, 0), machine) = "Avariada" And manutencao(3) < s.RunCurrentTime Then
      nMaxResources = s.ResourcesMaximum
      With s
        For nIndex = 1 To nMaxResources
          recurso = .ConstructString(smSimanConstructResources, nIndex, 1)
          maq = Val(Mid$(recurso, 11, 2))
          lin = Val(Mid$(recurso, 9, 2))
          If machine = maq And lin = linhaman(1, 0) Then
              manutencao(0) = linhaman(1, 0)
              manutencao(1) = machine
              manutencao(2) = s.RunCurrentTime + 45      ' rota=45 deslocacao da manutencao
              manutencao(3)   =   CLng(s.RunCurrentTime   +   ler_expressao(CStr(lin),   CStr(machine),
"TempoAvaria"))
              manutencao(4) = nIndex
              estado = .ConstructString(smSimanConstructStateSets, nIndex, .ResourceState(nIndex))
              Call  manutencao_control(CStr(manutencao(2)),  CStr(manutencao(3)),  CStr(linhaman(1,  0)),
CStr(machine), "Envia Manutencao")
              Exit For
            End If
          Next nIndex
      End With
    End If
  End If

If manutencao(2) = s.RunCurrentTime Then
  s.ResourceState(manutencao(4)) = 4 'em manutenção
End If

If manutencao(3) = s.RunCurrentTime Then
  somatot     =     ler_fila(CStr(manutencao(0)),     CStr(manutencao(1)),     "Fila     Pacote")     +
ler_fila(CStr(manutencao(0)),  CStr(manutencao(1)),  "Processa.Queue")  +  ler_fila(CStr(manutencao(0)),
CStr(manutencao(1)), "Fila Fio") + ler_fila(CStr(manutencao(0)), CStr(manutencao(1)), "Fila Operador") +
ler_fila(CStr(manutencao(0)), CStr(manutencao(1)), "Batch.Queue")
  If soma <> 0 Then
    tx1 = "Maquina_" + CStr(manutencao(0)) + "_" + CStr(manutencao(1))
    tx3 = "produto_" + CStr(ler_variavel(CStr(manutencao(0)), CStr(manutencao(1)), "Imagemproduto"))
    tx2 = "fio_" + CStr(ler_variavel(CStr(manutencao(0)), CStr(manutencao(1)), "Imagemfio"))
    tx4 = "palete_" + CStr(ler_variavel(CStr(manutencao(0)), CStr(manutencao(1)), "Imagempalete"))

    soma = ler_fila(CStr(manutencao(0)), CStr(manutencao(1)), "Fila Pacote")
    If  soma > 0  Then Call  remover_das_filas(CStr(manutencao(0)),  CStr(manutencao(1)),  "Fila Pacote",
CLng(soma))
    soma = ler_fila(CStr(manutencao(0)), CStr(manutencao(1)), "Processa.Queue")
    If soma > 0 Then Call remover_das_filas(CStr(manutencao(0)), CStr(manutencao(1)), "Processa.Queue",
CLng(soma))
    soma = ler_fila(CStr(manutencao(0)), CStr(manutencao(1)), "Fila Fio")
    If  soma > 0  Then Call  remover_das_filas(CStr(manutencao(0)),  CStr(manutencao(1)),  "Fila Fio",
CLng(soma))
    soma = ler_fila(CStr(manutencao(0)), CStr(manutencao(1)), "Fila Operador")
    If soma > 0 Then Call remover_das_filas(CStr(manutencao(0)), CStr(manutencao(1)), "Fila Operador",
CLng(soma))
    soma = ler_fila(CStr(manutencao(0)), CStr(manutencao(1)), "Batch.Queue")
    If  soma > 0  Then Call  remover_das_filas(CStr(manutencao(0)),  CStr(manutencao(1)),  "Batch.Queue",
CLng(soma))

    Call insere_nas_filas(CStr(tx1), CLng(somatot), CStr(tx2), CStr(tx3), CStr(tx4))
  End If
    s.ResourceState(manutencao(4)) = 1 'manutenção coloca a maquina "Parada" espera o operador
  manutencao(0) = 0
  manutencao(1) = 0
  manutencao(2) = 0
  manutencao(3) = 0
End If
'*******************fim da manutenção***********************************************
End Sub

Function devolve_tipo_maquinas_linha(linha As String, num_maq As String)
Dim tipo As Long
Dim rs As DAO.Recordset
Dim bd As DAO.Database
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("DEF_machine_line")

If Not rs.EOF Then
  rs.MoveLast
  rs.MoveFirst
End If
tipo = 0
For i = 1 To rs.RecordCount
  If rs.Fields("line") = linha Then
    tipo = tipo + 1
    If tipo = CLng(num_maq) Then
      tipo = rs.Fields("Machine_cod")
      Exit For
    End If
  End If
  rs.MoveNext
Next i
devolve_tipo_maquinas_linha = tipo
rs.Close
bd.Close
End Function

Function ler_expressao(linha As String, maquina As String, expressao As String)
Set s = M.SIMAN

x = s.ExpressionsMaximum
txt = expressao + "_" + linha + "_" + CStr(maquina)
```

```
For t = 1 To x
  If txt = s.ConstructString(smSimanConstructExpressions, t, 1) Then
    Exit For
  End If
Next t
```

```
ler_expressao = s.ExpressionValue(t)
End Function
```

Anexo C

Escalonamento

```
Dim timeprocess(), z(), profit(), bi(), bk(), bj(), r()
Dim a(), max_comp(), max_prod(), tempo_prod()
Dim m1, m2, m3, M, N, tempo_max, maxmaq, linhas, colunas
Dim sinais_equacoes(), contador_linhas, si
Dim estrategias As String
Dim linhasactivas(), maqlinha(), lm(1)

Sub faz_matriz(ParamArray lista())
Dim X#(), ICASE&, IZROV&(), IPOSV&()
Dim i&, j&, M&, MP&, NP&, MinMax#
Dim aux()

linhas = CLng(lista(0))
colunas = CLng(lista(1))
r = lista(2)
bi = lista(3)
timeprocess = lista(4)
tempo_max = lista(5)
maxmaq = lista(6)
profit = lista(7)
max_comp = lista(8)
max_prod = lista(9)
tempo_prod = lista(10)

m1 = linhas
m2 = 0
m3 = 0
estrategias = ""

If Form_Main.chk_profit.Value = 1 Then
estrategias = estrategias + "_FWP"
End If

If Form_Main.chk_qtd_iguais.Value = 1 Then
m3 = m3 + colunas - 1
estrategias = estrategias + "_MEQ"
End If

If Form_Main.chk_quantidades.Value = 1 Then
m1 = m1 + colunas
m2 = m2 + colunas
estrategias = estrategias + "_MQL"
End If

If Form_Main.chk_tempos_iguais.Value = 1 Then
m3 = m3 + colunas - 1
estrategias = estrategias + "_MET"
End If

If Form_Main.chk_tempos.Value = 1 Then
m1 = m1 + colunas
m3 = m3 + colunas
estrategias = estrategias + "_MTL"
End If

If Form_Main.opt_maq_lucro_igual.Value = 1 Then
m3 = m3 + colunas - 1
estrategias = estrategias + "_MEP"
End If

If Form_Main.opt_p_qty_iguais.Value = 1 Then
m3 = m3 + linhas - 1
estrategias = estrategias + "_PEQ"
End If

If Form_Main.opt_p_qty.Value = 1 Then
m1 = m1 + linhas
m2 = m2 + linhas
estrategias = estrategias + "_PQL"
End If

If Form_Main.opt_p_tempos_iguais.Value = 1 Then
m3 = m3 + linhas - 1
estrategias = estrategias + "_PET"
End If

If Form_Main.opt_p_time.Value = 1 Then
m1 = m1 + linhas
m2 = m2 + linhas
estrategias = estrategias + "_PTL"
End If

If Form_Main.opt_p_lucro_igual.Value = 1 Then
m3 = m3 + linhas - 1
estrategias = estrategias + "_PEP"
End If

If Form_Main.chk_p_compo.Value = 1 Then
m1 = m1 + UBound(max_comp) / 2
m2 = m2 + UBound(max_comp) / 2
estrategias = estrategias + "_PCL"
End If

N = linhas * colunas

'**************************não escolhida*************************
'If Form_Main.opt_1prod_por_maquina.Value = 1 Then
'm1 = m1 + linhas * colunas + colunas
'N = linhas * colunas * 2 + 1
```

```
'aux = r
'ReDim r(linhas, colunas * 2)
'For i = 1 To linhas
' For j = 1 To colunas * 2
'  If j <= colunas Then r(i, j) = aux(i, j) Else r(i, j) = 0
'  Next j
'Next i
'Else
'N = linhas * colunas
'End If
'***************************não escolhida*************************

si = 1
M = m1 + m2 + m3    ' N. de equacoes
MP = M + 2          ' Dimensao necessaria
NP = N + 1          ' Matriz a().

ReDim a(1 To MP, 1 To NP) '
ReDim sinais_equacoes(1 To MP)

Call F1_funcao_objectivo
Call F2_limites_orders

'If Form_Main.opt_1prod_por_maquina.Value = 1 Then
'Call MP_1prod_por_maquina
'End If

If Form_Main.chk_tempos.Value = 1 Then
Call M4_m_imites_de_tempos
End If

If Form_Main.chk_quantidades.Value = 1 Then
Call M2_m_limites_de_quantidades
End If

If Form_Main.chk_tempos_iguais.Value = 1 Then
Call M3_m_tempos_iguais
End If

If Form_Main.chk_qtd_iguais.Value = 1 Then
Call M1_m_quantidades_iguais
End If

If Form_Main.opt_maq_lucro_igual.Value = 1 Then
Call M5_m_mesmo_lucro
End If

If Form_Main.opt_p_time.Value = 1 Then
Call P4_p_limites_tempos
End If

If Form_Main.opt_p_qty.Value = 1 Then
Call P2_p_limites_quantidades
End If

If Form_Main.opt_p_qty_iguais.Value = 1 Then
Call P1_p_quantidades_iguais
End If

If Form_Main.opt_p_tempos_iguais.Value = 1 Then
Call P3_p_tempos_iguais
End If

If Form_Main.opt_p_lucro_igual.Value = 1 Then
Call P5_p_mesmo_lucro
End If

If Form_Main.chk_p_compo.Value = 1 Then
Call P6_p_limites_componentes
End If

N = linhas * colunas
If Form_Main.opt_inteira.Value Then
Call calcula_inteira(a, linhas, colunas, M, sinais_equacoes, estrategias)
Else
  Simplx m1, m2, m3, ICASE, IZROV(), IPOSV()
  Call escreve_solucao_normal(ICASE, N, a, IPOSV, lista(0), lista(1), False)
End If

End Sub

Sub Simplx(ByVal m1&, ByVal m2&, ByVal m3& _
, ICASE&, IZROV&(), IPOSV&(), Optional ByVal eps# = 0.000001)
'
' Simplex method for linear programming. Input parameters a(), m1, m2, and m3,
' and output parameters a(), ICASE, IZROV(), and IPOSV() are described in Dati.
' Parameters:EPS is the absolute precision, which should be adjusted to
' the scale of your variables.
'
Dim i&, IP&, IX&, k&, KH&, KP&, M&, N&, NL1&
Dim bmax#, q1#
'
M = UBound(a, 1) - 2
N = UBound(a, 2) - 1
ReDim IPOSV(1 To M), IZROV(1 To N)
ReDim L1&(1 To N), L3&(1 To M)

If (M <> m1 + m2 + m3) Then
  MsgBox "Bad input constraint counts in Simplx.", vbCritical, " Simplx"
  ICASE = -2
  Exit Sub
End If

NL1 = N
For k = 1 To N
  L1(k) = k ' Initialize index list of columns admissible for exchange.
```

```
  IZROV(k) = k ' Initially make all variables right-hand.
Next k
For i = 1 To M
  If (a(i + 1, 1) < 0#) Then ' Constants bi must be non-negative.
    MsgBox "Bad input tableau in Simplx.", vbCritical, " Simplx"
    ICASE = -2
    Exit Sub
  End If
  IPOSV(i) = N + i
  ' Initial left-hand variables.
  ' m1 type constraints are represented by having their slack variable initially
  'left-hand, with no artificial variable.
  ' m2 type constraints have their slack variable initially left-hand, with a
  'minus sign, and their artificial variable handled implicitly during their
  'first exchange.
  ' m3 type constraints have their artificial variable initially left-hand.
Next i
If (m2 + m3 = 0) Then GoTo 30 ' The origin is a feasible starting solution.

' Go to phase two.

For i = 1 To m2 ' Initialize list of m2 constraints whose slack
  L3(i) = 1 ' variables have never been exchanged out of the
Next i ' initial basis.

For k = 1 To N + 1 ' Compute the auxiliary objective function.
  q1 = 0#
  For i = m1 + 1 To M
    q1 = q1 + a(i + 1, k)
  Next i
  a(M + 2, k) = -q1
Next k

10 Simp1 M + 1, L1(), NL1, 0, KP, bmax ' Find max. coeff. of auxiliary
' objective fn.
If (bmax <= eps) And (a(M + 2, 1) < -eps) Then
  ICASE = -1 ' Auxiliary objective function is
  Exit Sub ' still negative and can't be im-
 ElseIf (bmax <= eps) And (a(M + 2, 1) <= eps) Then
  For IP = m1 + m2 + 1 To M
    If IPOSV(IP) = IP + N Then ' Found an artificial variable for
    ' an equality constraint.
      Simp1 IP, L1(), NL1, 1, KP, bmax
      If (bmax > eps) Then GoTo 1 ' Exchange with column corresponding
       ' to maximum pivot element in row.
    End If
  Next IP
  For i = m1 + 1 To m1 + m2 ' Change sign of row for any m2
  ' constraints still present from
  ' the initial basis.
    If L3(i - m1) = 1 Then
      For k = 1 To N + 1
        a(i + 1, k) = -a(i + 1, k)
      Next k
    End If
  Next i
  GoTo 30 ' Go to phase two.
End If

Simp2 M, N, IP, KP, eps ' Locate a pivot element (phase one).
If IP = 0 Then ' Maximum of auxiliary objective
  ICASE = -1 ' function is unbounded, so no
  Exit Sub ' feasible solution exists.
End If

1 Simp3 M + 1, N, IP, KP
' Exchange a left- and a right-hand variable (phase one), then update lists.
If IPOSV(IP) >= (N + m1 + m2 + 1) Then ' Exchanged out an artificial variable
  ' for an equality constraint. Make
  ' sure it stays out by removing it
  ' from the l1 list.
  For k = 1 To NL1
    If L1(k) = KP Then GoTo 2
  Next k
2   NL1 = NL1 - 1
  For IX = k To NL1
    L1(IX) = L1(IX + 1)
  Next IX
 Else
  KH = IPOSV(IP) - m1 - N
  If KH >= 1 Then ' Exchanged out an m2 type constraint.
    If L3(KH) <> 0 Then ' If it's the first time, correct
      ' the pivot column for the minus
      ' sign and the implicit artificial
      ' variable.
      L3(KH) = 0
      a(M + 2, KP + 1) = a(M + 2, KP + 1) + 1#
      For i = 1 To M + 2
        a(i, KP + 1) = -a(i, KP + 1)
      Next i
    End If
  End If
End If
IX = IZROV(KP) ' Update lists of left- and
' right-hand variables.
IZROV(KP) = IPOSV(IP)
IPOSV(IP) = IX
GoTo 10 ' Still in phase one, go back to 10.
' End of phase one code for finding an initial feasible solution.

' Now, in phase two, optimize it.
30 Simp1 0, L1(), NL1, 0, KP, bmax ' Test the z-row for doneness.
If bmax <= eps Then ' Done. Solution found.
  ICASE = 0 ' Return with the good news.
  Exit Sub
End If
```

```
Simp2 M, N, IP, KP, eps ' Locate a pivot element (phase two).
If IP = 0 Then ' Objective function is unbounded.
  ICASE = 1 ' Report and return.
  Exit Sub
End If

Simp3 M, N, IP, KP ' Exchange a left- and a right-hand
' variable (phase two),
IX = IZROV(KP) ' update lists of left- and
' right-hand variables,
IZROV(KP) = IPOSV(IP)
IPOSV(IP) = IX
GoTo 30 ' and return for another iteration.

End Sub

Sub Simp1(ByVal MM&, LL&(), ByVal NLL&, ByVal IABF&, KP&, bmax#)
'
' Determines the maximum of those elements whose index is contained in the
' supplied list LL(), either with or without taking the absolute value, as
' flagged by IABF.
'
Dim k&, test#

If NLL <= 0 Then ' No eligible columns.
  bmax = 0#
 Else
  KP = LL(1)
  bmax = a(MM + 1, KP + 1)
  For k = 2 To NLL
    If IABF = 0 Then
      test = a(MM + 1, LL(k) + 1) - bmax
     Else
       test = Abs(a(MM + 1, LL(k) + 1)) - Abs(bmax)
    End If
    If test > 0# Then
      bmax = a(MM + 1, LL(k) + 1)
      KP = LL(k)
    End If
  Next k
End If
End Sub

Sub Simp2(ByVal M&, ByVal N&, IP&, ByVal KP&, ByVal eps#)
'
' Locate a pivot element, taking degeneracy into account.
'
Dim i&, k&
Dim q#, q0#, q1#, qp#
'Const eps = 0.000001

IP = 0
For i = 1 To M
  If (a(i + 1, KP + 1) < -eps) Then GoTo 1
Next i
MsgBox "No possible pivots.", vbInformation, " Simplx"
Exit Sub ' Return with message.

1 q1 = -a(i + 1, 1) / a(i + 1, KP + 1)
IP = i
For i = IP + 1 To M
  If (a(i + 1, KP + 1) < -eps) Then
    q = -a(i + 1, 1) / a(i + 1, KP + 1)
    If q < q1 Then
      IP = i
      q1 = q
     ElseIf q = q1 Then ' We have a degeneracy.
       For k = 1 To N
         qp = -a(IP + 1, k + 1) / a(IP + 1, KP + 1)
         q0 = -a(i + 1, k + 1) / a(i + 1, KP + 1)
         If (q0 <> qp) Then GoTo 2
       Next k
2   If (q0 < qp) Then IP = i
    End If
  End If
Next i

End Sub

Sub Simp3(ByVal I1&, ByVal K1&, ByVal IP&, ByVal KP&)
'
' Matrix operations to exchange a left-hand and right-hand variable (see text).
'
Dim II&, KK&
Dim piv#

piv = 1# / a(IP + 1, KP + 1)
For II = 1 To I1 + 1
  If II - 1 <> IP Then
    a(II, KP + 1) = a(II, KP + 1) * piv
    For KK = 1 To K1 + 1
      If KK - 1 <> KP Then
        a(II, KK) = a(II, KK) - a(IP + 1, KK) * a(II, KP + 1)
      End If
    Next KK
  End If
Next II

For KK = 1 To K1 + 1
  If (KK - 1 <> KP) Then a(IP + 1, KK) = -a(IP + 1, KK) * piv
Next KK
a(IP + 1, KP + 1) = piv
End Sub

Sub Simpinfo()
```

```
' Forma do problema de programacao linear.
'
' Maximizar a:
' z = a01 * x1 + a02 * x2 + ... + a0n * xn con x1 >= 0, x2 >= 0, ..., xn >= 0,
'
' sujeita as retricoes:
' m1 inequacoes do tipo 1:
' ai1 * x1 + ai2 * x2 + ... + ain * xn <= bi com i = 1, ..., m1
'
' m2 inequacoes do tipo 2:
' aj1 * x1 + aj2 * x2 + ... + ajn * xn >= bj com j = m1 + 1, ..., m1 + m2
'
' m3 inequacoes do tipo 3:
' ak1 * x1 + ak2 * x2 + ... + akn * xn = bk com k = m1 + m2 + 1, ..., m1 + m2 + m3
'
' e com todos b >= 0.
'
' Os dados do problema, passam para Simplx a(), m1, m2, m3, ...,
' devem ser organizados como:
'
' N = N. de incognitas.
'
' m1 = N. de inequacoes do tipo 1:
' m2 = N. de inequacoes do tipo 2:
' m3 = N. de inequacoes do tipo 3:
' M = m1 + m2 + m3
'
' MP = M + 2: NP = N + 1
' ReDim a#(1 To MP, 1 To NP), IPOSV&(1 To M), IZROV&(1 To N)
'
' O valor da matriz a() entra na Simplx como:
'
' 0. coeficientes da equacao de maximizacao:
' a(1, 1) = 0: a(1, 2) = a01: a(1, 2) = a01: ...: a(1, N + 1) = a0n
'
' 1. m1 coeficientes da inequacao do tipo <= bi :
' a(2, 1) = bi: a(2, 2) = -ai1: a(2, 3) = -ai2: ...: a(2, N + 1) = -ain
' ......
'
' 2. m2 coeficientes da inequaca do tipo >= bj :
' a(f, 1) = bj: a(f, 2) = -aj1: a(f, 3) = -aj2: ...: a(f, N + 1) = -ajn
' ......
' con f = m1 + 1
'
' 3. m3 coeficientes da equacao do tipo = bk :
' a(g, 1) = bk: a(g, 2) = -ak1: a(g, 3) = -ak2: ...: a(g, N + 1) = -akn
' ......
' con g = m1 + m2 + 1
'
'
' A solucao do problema e devolvida pela funcao Simplx a(), ..., ICASE, IZROV(), IPOSV()
'
' ICASE = 0 se e devolvida uma solucao finita.
' = +1 se a solucao e ilimitada
' = -1 se as restriccoes não podem ser cumpridas.
' = -2 se existe erros nos paramentros ao passar para a funcao Simplx.
'
' Se ICASE = 0 posso conhecer o valor x da solucao con:
'
' Redim X#(1 To N) ' Vector da solucao.
' For L = 1 To M
' If IPOSV(L) <= N Then
' X(IPOSV(L)) = a(L + 1, 1)
' End If
' Next L
' a(1, 1) = Valor de z(X(1), X(2), ..., X(N))
End Sub

Sub F1_funcao_objectivo()
si = 1

'N = linhas * colunas * 2 + 1

' escreve a matriz a zero
For i = 1 To m1 + m2 + m3 + 1
  For j = 1 To linhas * colunas + 1
    a(i, j) = 0
  Next j
Next i

'Coeficiente da equacao fa funcao objectivo : 1 maximizar ; -1 minimizar
' equacao da funcao objectivo
MinMax = 1 ' maximizar
bx = 1
For i = 1 To linhas
  For j = 1 To colunas
    a(1, bx + 1) = MinMax * r(i, j)
    'If Form_Main.opt_1prod_por_maquina.Value Then
    '  a(1, bx + 1 + colunas * linhas) = r(i, j)
    '  a(1, bx + 1) = r(i, j)
    'End If
    If Form_Main.chk_profit.Value Then a(1, bx + 1) = MinMax * r(i, j) * profit(i)
    bx = bx + 1
  Next j
Next i
a(1, 1) = 0#
contador_linhas = 1
End Sub

Sub F2_limites_orders()
l = 1: c = 1
For i = 1 To linhas
  For j = 1 To linhas * colunas
    a(i + 1, j + 1) = -r(l, c)
    If (j Mod colunas) = 0 Then
      l = l + 1: i = i + 1: c = 0
```

```
    End If
    c = c + 1
  Next j
  c = colunas + c

Next i
' valores de bi
For i = 1 To linhas
  a(i + 1, 1) = bi(i)
  sinais_equacoes(si) = 0 ' LE
  si = si + 1
Next i
contador_linhas = contador_linhas + linhas
End Sub
Sub M1_m_quantidades_iguais()
c = 1: v = 0
For j = 1 To colunas - 1
  For i = 1 To linhas
    a(j + contador_linhas, c + 1 + v) = -1 * r(i, j)
    a(j + contador_linhas, c + v + 2) = r(i, j + 1)
    c = c + 2
    v = v + colunas - 2
  Next i
  c = 1
  v = j
Next j
' valores de bk
For i = 1 To colunas - 1
  a(i + contador_linhas, 1) = 0 ' quantidades iguais em todas as maquinas (colunas)
  sinais_equacoes(si) = 1 ' EQ
  si = si + 1
Next i
contador_linhas = contador_linhas + colunas - 1
End Sub

Sub M3_m_tempos_iguais()
c = 1
v = 0
For j = 1 To colunas - 1
  For i = 1 To linhas
    a(j + contador_linhas, c + 1 + v) = -1 * r(i, j) * timeprocess(i, j)
    a(j + contador_linhas, c + v + 2) = r(i, j + 1) * timeprocess(i, j + 1)
    c = c + 2
    v = v + colunas - 2
  Next i
  c = 1
  v = j
  sinais_equacoes(si) = 1 ' EQ
  si = si + 1
Next j
' valores de bk
For i = 1 To colunas - 1
  a(i + contador_linhas, 1) = 0 ' para o valor das colunas serem iguais
Next i
contador_linhas = contador_linhas + colunas - 1
End Sub

Sub M4_m_imites_de_tempos()
c = 0
l = 1
For j = 1 To colunas
  For i = 1 To linhas
    a(l + contador_linhas, j + 1 + c) = -r(i, j) * timeprocess(i, j)
    c = c + colunas
  Next i
  l = l + 1
  c = 0
Next j
For i = 1 To colunas
  a(i + contador_linhas, 1) = tempo_max(i + colunas)
  sinais_equacoes(si) = 0 ' LE
  si = si + 1
Next i
' ********************************************************************
contador_linhas = contador_linhas + colunas
c = 0
l = 1
For j = 1 To colunas
  For i = 1 To linhas
    a(l + contador_linhas, j + 1 + c) = -r(i, j) * timeprocess(i, j)
    c = c + colunas
  Next i
  l = l + 1
  c = 0
Next j
For i = 1 To colunas
  a(i + contador_linhas, 1) = tempo_max(i)
  sinais_equacoes(si) = 2 ' GE
  si = si + 1
Next i
contador_linhas = contador_linhas + colunas
End Sub

Sub M2_m_limites_de_quantidades()
c = 0
l = 1
For j = 1 To colunas
  For i = 1 To linhas
    a(l + contador_linhas, j + 1 + c) = -r(i, j)
    c = c + colunas
  Next i
  l = l + 1
  c = 0
Next j
For i = 1 To colunas
  a(i + contador_linhas, 1) = maxmaq(i + colunas)
```

```
  sinais_equacoes(si) = 0 ' LE
  si = si + 1
Next i
contador_linhas = contador_linhas + colunas

' ********************************************************************
l = 1
c = 0
For j = 1 To colunas
  For i = 1 To linhas
    a(l + contador_linhas, j + 1 + c) = -r(i, j)
    c = c + colunas
  Next i
  l = l + 1
  c = 0
Next j
For i = 1 To colunas
  a(i + contador_linhas, 1) = maxmaq(i)
  sinais_equacoes(si) = 2 ' GE
  si = si + 1
Next i
contador_linhas = contador_linhas + colunas
End Sub

Sub P2_p_limites_quantidades()
c = 1
cl = 0
l = 1
For i = 1 To linhas
  For j = 1 To colunas
    a(i + contador_linhas, j + cl + 1) = -r(l, c)
    c = c + 1
  Next j
  c = 1
  l = l + 1
  cl = cl + colunas
Next i
For i = 1 To linhas
  a(i + contador_linhas, 1) = max_prod(i + linhas) 'filipe                'tp_maq(i + colunas)
  sinais_equacoes(si) = 0 ' LE
  si = si + 1
Next i
contador_linhas = contador_linhas + linhas
cl = 0
l = 1
For i = 1 To linhas
  For j = 1 To colunas
    a(i + contador_linhas, j + cl + 1) = -r(l, c)
    c = c + 1
  Next j
  c = 1
  l = l + 1
  cl = cl + colunas
Next i

For i = 1 To linhas
  a(i + contador_linhas, 1) = max_prod(i)
  sinais_equacoes(si) = 2 ' GE
  si = si + 1
Next i
contador_linhas = contador_linhas + linhas
End Sub

Sub P1_p_quantidades_iguais()
c = 0
For i = 1 To linhas - 1
  For j = 1 To colunas
    a(i + contador_linhas, c + j + 1) = -r(i, j)
    a(i + contador_linhas, j + c + colunas + 1) = r(i + 1, j)
  Next j
c = colunas
Next i
For i = 1 To linhas - 1
  a(i + contador_linhas, 1) = 0
  sinais_equacoes(si) = 1 ' EQ
  si = si + 1
Next i
contador_linhas = contador_linhas + linhas - 1
End Sub

Sub P3_p_tempos_iguais()
c = 0
For i = 1 To linhas - 1
  For j = 1 To colunas
    a(i + contador_linhas, j + 1 + c) = r(i, j) * timeprocess(i, j)
    a(i + contador_linhas, j + 1 + colunas + c) = -r(i + 1, j) * timeprocess(i + 1, j)
  Next j
  c = c + colunas
Next i
For i = 1 To linhas - 1
  a(i + contador_linhas, 1) = 0
  sinais_equacoes(si) = 1 ' EQ
  si = si + 1
Next i
contador_linhas = contador_linhas + linhas - 1

End Sub

Sub P4_p_limites_tempos()
c = 1
cl = 0
l = 1
For i = 1 To linhas
  For j = 1 To colunas
    a(i + contador_linhas, j + cl + 1) = -r(l, c) * timeprocess(i, j)
    c = c + 1
```

```
  Next j
  c = 1
  l = l + 1
  cl = cl + colunas
Next i
For i = 1 To linhas
  a(i + contador_linhas, 1) = tempo_prod(i + linhas)
  sinais_equacoes(si) = 0 ' LE
  si = si + 1
Next i
contador_linhas = contador_linhas + linhas
cl = 0
l = 1
For i = 1 To linhas
  For j = 1 To colunas
    a(i + contador_linhas, j + cl + 1) = -r(l, c) * timeprocess(i, j)
    c = c + 1
  Next j
  c = 1
  l = l + 1
  cl = cl + colunas
Next i
For i = 1 To linhas
  a(i + contador_linhas, 1) = tempo_prod(i)
  sinais_equacoes(si) = 2 ' GE
  si = si + 1
Next i
contador_linhas = contador_linhas + linhas
End Sub

Sub M5_m_mesmo_lucro()
c = 1
v = 0
For j = 1 To colunas - 1
  For i = 1 To linhas
    a(j + contador_linhas, c + 1 + v) = -1 * r(i, j) * profit(i)
    a(j + contador_linhas, c + v + 2) = r(i, j + 1) * profit(i)
    c = c + 2
    v = v + colunas - 2
  Next i
  c = 1
  v = j
  sinais_equacoes(si) = 1 ' EQ
  si = si + 1
Next j
' valores de bk
For i = 1 To colunas - 1
  a(i + contador_linhas, 1) = 0 ' para o valor das colunas serem iguais
Next i
contador_linhas = contador_linhas + colunas - 1
End Sub

Sub MP_1prod_por_maquina()
vc = 0
vy = linhas * colunas
LL = 0
For i = 1 To linhas
 For j = 1 To colunas
  a(i + contador_linhas + LL, j + 1 + vc) = -r(i, j)
  a(i + contador_linhas + LL, j + 1 + vy + vc) = bi(i)
   LL = LL + 1
 Next j
  LL = LL - 1
  vc = vc + colunas
Next i
For i = 1 To linhas * colunas
  a(i + contador_linhas, 1) = 0
  sinais_equacoes(si) = 0 ' LE
  si = si + 1
 Next i
'*****************************
contador_linhas = contador_linhas + linhas * colunas
vy = linhas * colunas
vz = 0
LL = l + colunas - 1
For j = 1 To colunas
  For i = 1 To linhas
    a(j + contador_linhas, vz + vy + i + 1) = -r(i, j)
    vz = vz + colunas - 1
  Next i
  vz = j
  vx = 1
Next j
' valores de bk
For i = 1 To colunas
  a(i + contador_linhas, 1) = 1
  sinais_equacoes(si) = 0 'LE '1 ' EQ
  si = si + 1
Next i
contador_linhas = contador_linhas + colunas
End Sub

Sub P5_p_mesmo_lucro()
c = 0
For i = 1 To linhas - 1
  For j = 1 To colunas
    a(i + contador_linhas, j + 1 + c) = r(i, j) * profit(i)
    a(i + contador_linhas, j + 1 + colunas + c) = -r(i + 1, j) * profit(i + 1)
  Next j
  c = c + colunas
Next i
For i = 1 To linhas - 1
  a(i + contador_linhas, 1) = 0
  sinais_equacoes(si) = 1 ' EQ
  si = si + 1
Next i
```

```
contador_linhas = contador_linhas + linhas - 1
End Sub

Sub P6_p_limites_componentes()
Dim matt(), prodt(), prods()

num_maqs = Form_Main.devolve_num_maquinas()
num_prod = Form_Main.devolve_num_produtos()
num_comp = Form_Main.devolve_num_materiais_usados()
matt = Form_Main.devolve_tipo_materiais_usados()
prods = Form_Main.devolve_tipo_produtos(CLng(num_prod))

c = 1
For i = 1 To num_comp
  For j = 1 To num_prod
   For k = 1 To num_maqs
      qty = Val(Form_Main.devolve_qty_bom(Trim(prods(j)), Trim(matt(i))))
      a(i + contador_linhas, k + c) = qty * r(j, k) * (-1)
   Next k
   c = c + k - 1
 Next j
 c = 1
Next i

' valores de bk
  For j = 1 To num_comp
    a(j + contador_linhas, 1) = max_comp(j + num_comp)
   sinais_equacoes(si) = 0 ' LE
   si = si + 1
Next j
contador_linhas = contador_linhas + num_comp

c = 1
For i = 1 To num_comp
  For j = 1 To num_prod
   For k = 1 To num_maqs
      qty = Val(Form_Main.devolve_qty_bom(Trim(prods(j)), Trim(matt(i))))
      a(i + contador_linhas, k + c) = qty * r(j, k) * (-1)
   Next k
   c = c + k - 1
 Next j
 c = 1
Next i

' valores de bk
  For j = 1 To num_comp
    a(j + contador_linhas, 1) = max_comp(j)
   sinais_equacoes(si) = 2 ' GE
   si = si + 1
Next j
contador_linhas = contador_linhas + num_comp

End Sub
```

```
Sub nao_usada_escreve_solucao_inteira(ParamArray lista())
' escreve a solucao do problema
Dim rs As Recordset
Dim bd As Database
Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_manufacturing", dbOpenTable)
  c = 1: l = 1:  pp = 1
  For j = 0 To lista(0) * lista(1) - 1
      With rs
        If lista(2)(j) > 0 Then
          rs.AddNew
          rs.Fields("line_prod") = l
          rs.Fields("number") = j
          rs.Fields("qty") = lista(2)(j)
          rs.Fields("product") = pp
          rs.Update
        End If
      End With
    c = c + 1
    If c > lista(0) Then
      c = 1: l = l + 1: pp = pp + 1
    End If
  Next j
  rs.Close

End Sub

Sub desenha_valores_grelha_inteira(ParamArray lista())

Dim maq, prod, c As Integer
Dim resultados() As Double

Form_Main.grid_result.Visible = True

maq = lista(0)
prod = lista(1)
ReDim resultados(1 To maq * prod)
resultados = lista(2)

c = 0

For i = 1 To prod
  For j = 1 To maq
    resultados(c) = Int(resultados(c) + 0.001)
    Form_Main.grid_result.Row = i
    Form_Main.grid_result.Col = j
    Form_Main.grid_result.CellAlignment = flexAlignCenterCenter
    Form_Main.grid_result.Text = "---"
    If resultados(c) <> 0 Then Form_Main.grid_result.Text = resultados(c)
    c = c + 1
  Next
Next
```

```
End Sub

Sub escreve_solucao_normal(ParamArray lista())
' escreve a solucao do problema
Dim rs As Recordset
Dim bd As Database
Dim aux(), M, prod_nome()

Set bd = OpenDatabase(".\db.mdb")
Set rs = bd.OpenRecordset("CTRL_manufacturing", dbOpenTable)
ICASE = lista(0)

produtos = CLng(lista(4))
maquinas = CLng(lista(5))
prod_nome = Form_Main.devolve_tipo_produtos(CLng(produtos))
N = lista(1)

'If Form_Main.opt_1prod_por_maquina.Value = 1 Then
'colunas = colunas / 2 'reajustar o numero de colunas com o numero de maquinas
'maquinas = colunas
'N = N / 2
'End If

ReDim X(0 To N) As Double
ReDim aux(N)

linhas = Form_Main.devolve_count_linhas()
ReDim linhasactivas(linhas)
ReDim maqlinha(linhas)
fim = 0

For i = 1 To 40
  qt = Form_Main.devolve_count_maquinas_linha(CLng(i))
  If qt > 0 Then
    linhasactivas(fim + 1) = i
    maqlinha(fim + 1) = qt
    fim = fim + 1
   End If
  If fim = linhas Then Exit For
Next i

KM = 0
KL = 1
M = linhasactivas(1)

For i = 1 To 0 ' produtos * maquinas
  If M > maqlinha(KL) Then
    KL = KL + 1
    M = linhasactivas(1)
    If KL > UBound(linhasactivas) Then
      KL = 1
    End If
   End If
   aux(KM) = KL 'linhasactivas(KL)
   M = M + 1
   KM = KM + 1
Next i

LL = LL
If ICASE = 0 Then
  ' Prepara o vector da solucao:

   If lista(6) Then
     X = lista(2)
     inicio = 0
     N = N - 1
    Else
     inicio = 1

      For j = 1 To UBound(lista(3))
       If lista(3)(j) <= N Then
          X(lista(3)(j)) = lista(2)(j + 1, 1)
        End If
     Next j
   End If

  p = 1
  c = 1
  i = 0

  For j = inicio To N
      With rs
        If X(j) > 0 Then
          If lista(6) Then X(j) = Int(X(j) + 0.0001)

          If j - 1 > maquinas Then
            M = j Mod maquinas
          End If
          If M = 0 Then
            M = maquinas
          End If
          gato = X(j) * 100
          gatoi = Int(gato)
          gato = Abs(gatoi - gato)
          If gato > 0.5 Then
          gato = gatoi / 100 + 0.01
          Else
          gato = gatoi / 100
          End If

          If gato <> 0 Then
            rs.AddNew
            rs.Fields("number") = c
            rs.Fields("qty") = gato
            rs.Fields("product") = p
            rs.Fields("number_silva") = M
```

```
            lnha = devolve_posicoes(CLng(j), CLng(produtos))
            rs.Fields("line_prod") = lnha
            rs.Fields("machine") = "Machine_" + Trim(str(lm(0))) + "_" + Trim(str(lm(1)))
            rs.Fields("desc") = prod_nome(p)
            rs.Update
          End If
          i = i + 1 ' saber se funcionou bem o modelo simplex
        End If
      End With
     c = c + 1
    If c > maquinas Then
      p = p + 1
      c = 1
    End If
  Next j
  If i = 0 Then MsgBox "No solution. Please confirm the model.", vbExclamation
  txt_solucao = txt_solucao & vbNewLine
 ElseIf ICASE = 1 Then
  MsgBox "The solution is ilimited.", vbExclamation
 ElseIf ICASE = -1 Then
   MsgBox "The problem is not possible." & vbNewLine _
    & "The resticions are incompatible.", vbExclamation
End If

If Form_Main.existe_solucao Then
rs.MoveFirst
While Not rs.EOF
  With rs
    rs.Edit
    Form_Main.grid_result.Row = rs.Fields("product")
    Form_Main.grid_result.Col = rs.Fields("number")
    Form_Main.grid_result.CellAlignment = flexAlignCenterCenter
    Form_Main.grid_result.Text = rs.Fields("qty")
End With
rs.Update
rs.MoveNext
Wend
Form_Main.Frame(40).Visible = True
Form_Main.grid_result.Visible = True
Form_Main.Label(66).Visible = True
Form_Main.Label(67).Visible = True
Form_Main.Label(68).Visible = True
Form_Main.Label(69).Visible = True

End If
End Sub

Function devolve_posicoes(nmaq As Long, produtos As Long)
  maxl = UBound(linhasactivas, 1)
  xx = 0
  lin = 1
  For t = 1 To maxl
    xx = maqlinha(t) + xx
  Next t
  If nmaq > xx Then
    nmaq = nmaq Mod xx
  End If
  i = linhasactivas(1)
  c = 1
  maq = maqlinha(1)
  ln = i
  mq = nmaq
  w = 0
  While maq < nmaq
    If c + 1 <= maxl Then
      c = c + 1
      i = linhasactivas(c)
      maq = maq + maqlinha(c)
      ln = i
      w = maqlinha(c - 1) + w
      mq = Abs(w - nmaq)
    End If
  Wend
lm(0) = ln
lm(1) = mq
devolve_posicoes = c
If nmaq = 0 Then
  lm(0) = linhasactivas(maxl)
  lm(1) = maqlinha(maxl)
  devolve_posicoes = maxl
End If
End Function
```

Anexo D

Geração Automática de Modelos

```
' ************************************************************
' Routine to Open Arena
' ************************************************************
Public Sub Open_Arena()
On Error GoTo Arena_error

    ' Opens the Arena Software
    Set a = GetObject("", "Arena.Application")
    a.Visible = True

    ' Activates the Arena Software
    a.Activate
    a.Refresh

    Exit Sub

Arena_error:
    MsgBox "Problem Encountered Trying to Retrieve Arena Information." & vbCr & _
        "Be Sure That Arena is Installed in your Computer", vbOKOnly, "Error Encountered"

End Sub

' ************************************************************
' Routine to Load Arena
' ************************************************************
Public Sub Load_Arena(filename As String)
On Error GoTo Arena_error

    ' Opens the Arena Software
    Set a = GetObject("", "Arena.Application")
    a.Visible = True

    ' Activates the Arena Software
    a.Activate
    a.Refresh

    If filename = "" Then
        Set M = a.Models.Add
        M.ActiveView.ZoomAll
    Else
        Set M = a.Models.Open(filename)
        M.ActiveView.ZoomAll
    End If

    Set s = M.SIMAN

    ' Attachs several panels to the Arena Software
    a.Panels.Attach "BasicProcess"
    a.Panels.Attach "AdvancedProcess"
    a.Panels.Attach "AdvancedTransfer"
    a.Panels.Attach "Blocks"
    a.Panels.Attach "Elements"

    ' Attachs the TAF panel to the Arena Software
    a.Panels.Attach CurrentDir + "\" + "TAF Template"

    Exit Sub

Arena_error:
    MsgBox "Problem Encountered Trying to Retrieve Arena Information." & vbCr & _
        "Be Sure That Arena is Installed in your Computer", vbOKOnly, "Error Encountered"

End Sub

' ************************************************************
' Routine to Create a SubModel
' ************************************************************
Public Sub  Create_SubModel(Name As String, EntryPoints  As Long, ExitPoints  As Long, CoordX  As Long,
CoordY As Long)

    Dim SubMod As Arena.submodel

    Set SubMod = M.Submodels.Create(Name, CoordX, CoordY)
    SubMod.NumEntryPoints = EntryPoints
    SubMod.NumExitPoints = ExitPoints

End Sub

' ************************************************************
' Routine to Enter a SubModel
' ************************************************************
Public Sub Enter_SubModel(SubModelName As String)

    Dim indice As Long

    indice = M.Submodels.Find(smFindName, SubModelName)
    M.Submodels(indice).Model.Show
  '  If (Form_Main.Text2.Text < 9) And (Form_Main.Text1.Text < 9) Then
  '      m.Submodels(indice).Model.ActiveView.ZoomAll
  '  End If

End Sub

' ************************************************************
' Function to Return a SubModel Indice
' ************************************************************
Public Function SubModel_Index(SubModelName As String) As Long

    SubModel_Index = M.Submodels.Find(smFindName, SubModelName)

End Function
```

```
' ************************************************************
' Function to Create a Station Module
' ************************************************************
Public Function Create_Station(ModuleName As String, StationName As String, CoordX As Long, CoordY As Long) As Arena.Module

    Set Create_Station = M.Modules.Create("AdvancedTransfer", "Station", CoordX, CoordY)
    Create_Station.Data("Name") = ModuleName
    Create_Station.Data("Statn") = StationName
    Create_Station.UpdateShapes
    Create_Station.Shape.Selected = False

End Function

' ************************************************************
' Function to Create a Station Module in a SubModel
' ************************************************************
Public Function Create_Station_SubModel(SubModelName As String, ModuleName As String, StationName As String, CoordX As Long, CoordY As Long) As Arena.Module

    Call Enter_SubModel(SubModelName)
    Set Create_Station_SubModel = M.Submodels(SubModel_Index(SubModelName)).Model.Modules.Create("AdvancedTransfer", "Station", CoordX, CoordY)
    Create_Station_SubModel.Data("Name") = ModuleName
    Create_Station_SubModel.Data("Statn") = StationName
    Create_Station_SubModel.UpdateShapes
    Create_Station_SubModel.Shape.Selected = False
    M.Show

End Function

' ************************************************************
' Routine to Create an Animation of a Station
' ************************************************************
Public Sub Create_Station_Animation(StationName As String, CoordX As Long, CoordY As Long)
    Dim Station As Arena.Station

    Set Station = M.Stations.Create(CoordX, CoordY)
    Station.Identifier = StationName

End Sub

' ************************************************************
' Routine to Connect Animations of Stations
' ************************************************************
Public Sub Connect_Station_Animation(StationName1 As String, StationName2 As String)
    Dim Estacao1 As Arena.Station
    Dim Estacao2 As Arena.Station
    Dim Rota As Arena.route
    Dim e1 As Long
    Dim e2 As Long

    e1 = M.Stations.Find(smFindName, StationName1)
    e2 = M.Stations.Find(smFindName, StationName2)
    Set Estacao1 = M.Stations.Item(e1)
    Set Estacao2 = M.Stations.Item(e2)
    Set Rota = M.Routes.Create(Estacao1, Estacao2, True, True)
    Rota.Shape.Selected = False

End Sub

' ************************************************************
' Routine to Connect Animations of Stations in a Submodel
' ************************************************************
Public Sub Connect_Station_Animation_SubModel(SubModelName As String, StationName1 As String, StationName2 As String)
    Dim Estacao1 As Arena.Station
    Dim Estacao2 As Arena.Station
    Dim Rota As Arena.route
    Dim e1 As Long
    Dim e2 As Long

    e1 = M.Submodels(SubModel_Index(SubModelName)).Model.Stations.Find(smFindName, StationName1)
    e2 = M.Submodels(SubModel_Index(SubModelName)).Model.Stations.Find(smFindName, StationName2)
    Set Estacao1 = M.Submodels(SubModel_Index(SubModelName)).Model.Stations.Item(e1)
    Set Estacao2 = M.Submodels(SubModel_Index(SubModelName)).Model.Stations.Item(e2)
    Set Rota = M.Submodels(SubModel_Index(SubModelName)).Model.Routes.Create(Estacao1, Estacao2, True, True)
    Rota.Shape.Selected = False

End Sub

' ************************************************************
' Routine to Find a Shape in a Module (Systems Modeling)
' ************************************************************
Public Function FindShapeInModule(ByVal mInst As Arena.Module, ByVal objType As Long, ByVal Index As Long) As Object
    ' Return the <index>th object of type <objType> in module <mInst>, or Nothing if the appropriate
    '  match isn't found
    Dim i As Long, theshapes As Arena.Shapes, numshapes As Long, test As Arena.Shape, numFound As Long

    numFound = 0
    Set FindShapeInModule = Nothing
    Set theshapes = mInst.Shapes
    numshapes = theshapes.count
    For i = 1 To numshapes
        Set test = theshapes(i)
        If test.Type = objType Then
            numFound = numFound + 1
            If numFound = Index Then
                Set FindShapeInModule = test
                Exit For
            End If
        End If
    Next i
```

```
End Function

' ************************************************************
' Routine to Create the TAF Model
' ************************************************************
Public Sub Create_TAF_Model()
  Call Load_Arena(Model_FileName)
  Modelo_Criado = Modelo_Criado + 1
End Sub
' ************************************************************
' Routine to Create the TAF Model
' ************************************************************
Public Sub Create_TAF_linha()
'command_model_export
End Sub
' ************************************************************
' Routine to Create Variables
' ************************************************************
Public Sub Create_Variables(VarName As String, VarValue As String)
  Dim Mod1 As Arena.Module

  Set Mod1 = M.Modules.Create("BasicProcess", "Variable", 100, 100)
  Mod1.Data("Name") = VarName
  Mod1.Data("Initial Value") = VarValue
End Sub
' ************************************************************
' Routine to Create Expressions
' ************************************************************
Public Sub Create_Expression(ExpName As String, ExpValue As String)
  Dim Mod1 As Arena.Module

  Set Mod1 = M.Modules.Create("AdvancedProcess", "Expression", 100, 100)
  Mod1.Data("Name") = ExpName
  Mod1.Data("Value") = ExpValue
End Sub
' ************************************************************
' Routine to Connect The Animation Stations of the TAF Model
' ************************************************************
Public Sub Connect_Stations_TAF(NumMaquinas As Long, linha As String)
  Dim txt As String
  Dim txt2 As String
  Dim txt3 As String

  txt = linha
  Call Connect_Station_Animation("EstacaoOperador_" + txt, "Maquina_" + txt + "_" + "1")
  For i = 1 To NumMaquinas - 1
    txt2 = i
    For j = 2 To NumMaquinas
      txt3 = j
      Call Connect_Station_Animation("Maquina_" + txt + "_" + txt2, "Maquina_" + txt + "_" + txt3)
    Next
  Next
End Sub

' ************************************************************
' Routine to Define the Resource Pictures
' ************************************************************
Public Sub Resource_Pictures(nome As String, X As Long, y As Long, image As Long)
  Dim Resource_Pic As Arena.resourcePicture

  Set Resource_Pic = M.ResourcePictures.Create(X, y)
  Resource_Pic.Identifier = nome
  Resource_Pic.DeletePicture ("Idle")
  Resource_Pic.DeletePicture ("Busy")
  Resource_Pic.DeletePicture ("Inactive")
  Resource_Pic.DeletePicture ("Failed")
  Resource_Pic.SetPicture "Trabalhar", CurrentDir + "\" + "TAF_IMA_maq.plb", 1 + (image - 1) * 4
  Resource_Pic.SetPicture "Avariada", CurrentDir + "\" + "TAF_IMA_maq.plb", 2 + (image - 1) * 4
  Resource_Pic.SetPicture "Parada", CurrentDir + "\" + "TAF_IMA_maq.plb", 3 + (image - 1) * 4
  Resource_Pic.SetPicture "Manutencao", CurrentDir + "\" + "TAF_IMA_maq.plb", 4 + (image - 1) * 4
End Sub
Public Sub Entity_operador_Pictures(nome As String, X As Long, y As Long, image As Long)
Dim ent_img As Arena.entityPicture
Dim h, b As Long
'Set ent_img = m.EntityPictures.Create(X, Y)
'ent_img.DeletePicture "Default"
End Sub
Public Sub Entity_Pictures_create()
  Dim Entity_Pic As Arena.entityPicture
  Dim Mod1 As Arena.Module

  Set Entity_Pic = M.EntityPictures.Create(-28500, -31500)
  Entity_Pic.DeletePicture ("Default")
  For i = 0 To 11
    txt = "operador_" + Trim(str(i))
    Entity_Pic.SetPicture txt, CurrentDir + "\TAF_IMA_opr.plb", i + 1
  Next i
  For i = 1 To 16
    txt = "fio_" + Trim(str(i))
    Entity_Pic.SetPicture txt, CurrentDir + "\TAF_IMA_fios.plb", i
  Next i
  For i = 1 To 15
    txt = "palete_" + Trim(str(i))
    Entity_Pic.SetPicture txt, CurrentDir + "\TAF_IMA_pale.plb", i
  Next i

  For i = 1 To 12
    txt = "produto_" + Trim(str(i))
    Entity_Pic.SetPicture txt, CurrentDir + "\TAF_IMA_prod.plb", i
  Next i

  Set Mod1 = M.Modules.Create("BasicProcess", "Set", 100, 100)
  Mod1.Data("Name") = "imagem"
  Mod1.Data("Type") = "Entity Picture"
  For i = 1 To 11
    txt = "Picture Name(" + Trim(str(i)) + ")"
    Mod1.Data(txt) = "operador_" + Trim(str(i))
```

```
  Next i
  Set Mod1 = M.Modules.Create("BasicProcess", "Set", 100, 100)
  Mod1.Data("Name") = "imagemfio"
  Mod1.Data("Type") = "Entity Picture"
  For i = 1 To 16
    txt = "Picture Name(" + Trim(str(i)) + ")"
    Mod1.Data(txt) = "fio_" + Trim(str(i))
  Next i
  Set Mod1 = M.Modules.Create("BasicProcess", "Set", 100, 100)
  Mod1.Data("Name") = "imagempalete"
  Mod1.Data("Type") = "Entity Picture"
  For i = 1 To 15
    txt = "Picture Name(" + Trim(str(i)) + ")"
    Mod1.Data(txt) = "palete_" + Trim(str(i))
  Next i

  Set Mod1 = M.Modules.Create("BasicProcess", "Set", 100, 100)
  Mod1.Data("Name") = "imagemproduto"
  Mod1.Data("Type") = "Entity Picture"
  For i = 1 To 12
    txt = "Picture Name(" + Trim(str(i)) + ")"
    Mod1.Data(txt) = "produto_" + Trim(str(i))
  Next i
End Sub

' ************************************************************
' Function to Create an Operador Module
' ************************************************************
Public Function Create_Operador(NumOperadores As String, ImagemOperador As String, NomeEstacao As
String, TempoRota As String, EstacaoDestino As String, CoordX As Long, CoordY As Long) As Arena.Module
  Dim Resource_Pic As Arena.entityPicture

  Set Create_Operador = M.Modules.Create("TAF template", "Entrada Operador", CoordX, CoordY)
  Create_Operador.Data("Numero Operadores") = NumOperadores
  Create_Operador.Data("imagem") = ImagemOperador
  Create_Operador.Data("Nome Estacao") = NomeEstacao
  Create_Operador.Data("Tempo Rota") = TempoRota
  Create_Operador.Data("Estacao Destino") = EstacaoDestino
  Create_Operador.Shape.Selected = False

End Function
' ************************************************************
' Function to Create an Operador Module in a SubModel
' ************************************************************
Public Function Create_Operador_SubModel(SubModelName As String, NumOperadores As String, ImagemOperador
As String, NomeEstacao As String, TempoRota As String, EstacaoDestino As String, CoordX As Long, CoordY
As Long) As Arena.Module
  Call Enter_SubModel(SubModelName)
  Set  Create_Operador_SubModel  =  M.Submodels(SubModel_Index(SubModelName)).Model.Modules.Create("TAF
Template", "Entrada Operador", CoordX, CoordY)
  Create_Operador_SubModel.Data("Numero Operadores") = NumOperadores
  Create_Operador_SubModel.Data("imagem_operador") = ImagemOperador
  Create_Operador_SubModel.Data("Nome Estacao") = NomeEstacao
  Create_Operador_SubModel.Data("Tempo Rota") = TempoRota
  Create_Operador_SubModel.Data("Estacao Destino") = EstacaoDestino
  Create_Operador_SubModel.Shape.Selected = False
  M.Show
End Function
' ************************************************************
' Function to Create a Maquina Module
' ************************************************************
Public Function Create_Maquina(NomeEstacao As String, Sinal As String, TempoRota As String,
EstacaoDestino As String, Recurso As String, TempoProc As String, Quantidade As String, imagem As
String, TempoSetup As String, TempoFunc As String, TempoAvaria As String, Tipo As String, CoordX As
Long, CoordY As Long, imagemproduto As String, imagemfio As String, imagempalete As String) As
Arena.Module
  Set Create_Maquina = M.Modules.Create("TAF Template", "Maquina", CoordX, CoordY)
  Create_Maquina.Data("Nome Estacao") = NomeEstacao
  Create_Maquina.Data("Sinal") = Sinal
  Create_Maquina.Data("Tempo Rota") = TempoRota
  Create_Maquina.Data("Estacao Destino") = EstacaoDestino
  Create_Maquina.Data("Recurso") = Recurso
  Create_Maquina.Data("Tempo de Processamento") = TempoProc
  Create_Maquina.Data("Quantidade") = Quantidade
  Create_Maquina.Data("Tempo_Setup") = TempoSetup
  Create_Maquina.Data("Imagem") = imagem
  Create_Maquina.Data("Tempo_Func") = TempoFunc
  Create_Maquina.Data("Tempo_Avaria") = TempoAvaria

  Create_Maquina.Data("Imagemfio") = imagemfio
  Create_Maquina.Data("Imagempalete") = imagempalete
  Create_Maquina.Data("Imagemproduto") = imagemproduto

  Create_Maquina.Shape.Selected = False
End Function
' ************************************************************
' Function to Create a Maquina Module in a SubModel
' ************************************************************
Public Function Create_Maquina_SubModel(SubModelName As String, NomeEstacao As String, Sinal As String,
TempoRota As String, EstacaoDestino As String, Recurso As String, TempoProc As String, Quantidade As
String, imagem As String, TempoSetup As String, TempoFunc As String, TempoAvaria As String, CoordX As
Long, CoordY As Long) As Arena.Module
  Call Enter_SubModel(SubModelName)
  Set  Create_Maquina_SubModel  =  M.Submodels(SubModel_Index(SubModelName)).Model.Modules.Create("TAF
Template", "Maquina", CoordX, CoordY)
  Create_Maquina_SubModel.Data("Nome Estacao") = NomeEstacao
  Create_Maquina_SubModel.Data("Sinal") = Sinal
  Create_Maquina_SubModel.Data("Tempo Rota") = TempoRota
  Create_Maquina_SubModel.Data("Estacao Destino") = EstacaoDestino
  Create_Maquina_SubModel.Data("Recurso") = Recurso
  Create_Maquina_SubModel.Data("Tempo de Processamento") = TempoProc
  Create_Maquina_SubModel.Data("Quantidade") = Quantidade
  Create_Maquina_SubModel.Data("Tempo_Setup") = TempoSetup
  Create_Maquina_SubModel.Data("Imagem") = imagem

  Create_Maquina_SubModel.Data("Imagemfio") = imagemfio
```

```
Create_Maquina_SubModel.Data("Imagempalete") = imagempalete
Create_Maquina_SubModel.Data("Imagemproduto") = imagemproduto

Create_Maquina_SubModel.Data("Tempo_Func") = TempoFunc
Create_Maquina_SubModel.Data("Tempo_Avaria") = TempoAvaria
Create_Maquina_SubModel.Shape.Selected = False
M.Show
End Function
```

Bibliografia

Referências

[ANNI81] ANNIMO, J. S. Russel, E.C. "The Seven Most Frequent Causes of Simulation Analysis Failure – and How to Avoid them", Interfaces, 1981.

[AUDS90] AUDSLEY, N., A. Burns, "Real-Time System Scheduling", on Firts Year Report Task B of the Espirit BRA Project 3092: Predictably Dependable Computing Systems, Chapter 2, Vol 2 of 3, May, 1990.

[BALL96] BALL, P., "Introduction to Discrete Event Simulation", University of Strathclyde, 1996.

[BANK96] BANKS, J., Carson II, J. S., Barry, L. N., "Discret-Event System Simulation", 2ª ed., PrenticeHall, 1996.

[BILS94] BILSTEIN, João, M- L- Sequeira, "Gestão da Produção", Instituto de Apoio a pequenas e médias empresas, Abril, 1994.

[CHAS89] CHASE, R.B. & Aquilano, N.J., "Production and Operations Management", 5th ed., Irwin,1989.

[COST02] COSTA, Miguel Bueno, "Simulação de Sistemas", UFSCAR, 2002.

[DOUK87] DOUKIDIS, G. I., "An Anthology on the Homology of Simulation with Artificial Intelligence", Journal of the Operational Research Society 38, Nº 8, 1987.

[EDWA92] EDWARDS, G., Sankar, R. "Modeling and Simulatiion of Networks Using CSIM", Simulation, V58, nº2, pag. 131-136, Fevereiro, 1992.

[ENGL97] ENGLANDER, Robert. "Developing Java Beans", O'Reilly & Associates, 1997.

[FISH95] FISHEICK, P. "Simulation Model Design & Execution: Building Digital Worlds", Prentice Hall International,1995.

[GARC79] GARCEY, M. R., D. S. Johnson, "Computer and Intractabiliy: a Guide to the Theory of the NP-Completeness", W. H. Freeman and Company, 1979.

[GORD78] GORDON, G., "System Simulation", Prentice Hall, 2nd ed., 1978.

[HILL88] HILLIER, F. & Liebermann, "Introduction to operations research", Mc Graw Hill, 1988.

[KELT02] KELTON, W. David, Randall P. Sadowski, Deborah A. Sadowski, "Simulation With ARENA", 2nd ed, McGraw-hill, 2002.

[KNUT69] KNUTH, D.E., "The art of Computer Programming: Seminumerical Algorithms", vol.1. Reading, MA, Wesley, 1969.

[JERS88] JERSINE, Richard J., “Principles of Inventory and Materials Management”, Prentice-Hall, 1998.

[LAW91] LAW, A. M., Kelton, W. D., "Simulation Modeling & Analysis", 2nd ed., McGraw-Hill, 1991.

[LEEM99] LEEMIS, Lawrence. "Simulation Input Modeling", Proceedings of the 1999 Winter Simulation Conference P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

[LOBA95] LOBÃO, E.C.,"Uma Técnica de Modelagem para Simulação de Máquinas que leva a um Bom Produto", Revista Máquinas e Metais, pag.22-34, Porto, A.J.V., 1995.

[MACD75] MACDOUGALL, M. H. "System Level Simulation in Digital System Design Automation: Languages, Simulation & Data Base", Computer Science Press, 1975.

[MACD87] MACDOUGALL, M. H. "Simulating Computer Systems Techniques and Tools", MIT Press, 1987.

[MARY80] MARYANSKY, F. J., "Digital Computer Simulation", Haydess Book Company, 1980.

[NAYL66] NAYLOR, T.H., "Computer Simulation Tecniques", J. Wiley, 1966.

[PEGD90] PEGDEN, C.D. et al, "Introduction to Simulation Using SIMAN" Mc Graw Hill,1990

[PEGD91] PEGDEN, C. D. et al, "Introduction to Simulation Using Siman", McGraw-Hill, March 1991.

[PINT94] PINTO, Vitor, M., "Gestão da Manutenção", IAPMEI, Julho 1994.

[POOR89] POORTE, J.K., JOHSON, M.E., "A Hierarcal Aproach to Computer Animation in the Simulation Models”. Simulation 1989.

[REDD86] REDDY, R., "Epistemology of Knowledge Based Simulation", Simulation, 1987.

[RODR00] RODRIGUES, Nuno, João, Miguel, Machado, Ricardo, "Levatamento dos Requisitos da TAF", IDITE Minho, 2000.

[RODR96] RODRIGUES, A. Guimarães, "Simulação", Sebenta, Universidade do Minho, 1996.

[SANT94] SANTANA, R. H. C., Santana, M. J. Orlandi, R. C. G. S., SPOLON, R. Calônego, N., "Técnicas para avaliação de desempenho de sistemas computacionais", Notas didacticas, São Carlos, ICMSC-USP, 1994.

[SAYD85] SAYDAM, T. "Process Oriented Languages", Simuleter, vol 12 (2), Abril, 1995.

[SEQU94] SEQUEIRA, João M. Bilstein M. L., "Gestão de aprovisionamentos", IAPMEI, Junho, 1994.

[SHAN75] SHANNON, Robert E., "Systems Simulation: The Art and Science", Prentice-Hall, 1975.

[SHAN81] SHANNON, Robert E. "Operation Research Methodologics in Industrial Engeneering: a Survey", AIIE Transations, vol. 12, nº. 4, pag. 364-367, 1981.

[SHAR88] SHARMA, R. And Rose, L.I., " Modular Design for Simulation", Software Pratice and Experience, vol. 18, 1998.

[SOAR92] SOARES, L. F. G., "Modelagem em Simulação Discreta de Sistemas", Editora Campus Ltd, 1992.

[SOUZ92] SOUZA, R.C.G., "Desenvolvimento de uma Extensão Funcional em C para a Construção de um Ambiente de Simulação Orientado a Processo", Relatório de Iniciação Cientifica, ICMSC-USP,1992.

[SPOL92] SPOLON, R. "Uma Extensão Funcional de Modula-2 para Simulação de Sistemas Discretos", Relatório de Iniciação Cientifica, ICMSC-USP, 1992.

[SPOL94] SPOLON, R. "Um Editor Grafico para um Ambiente de Simulação Automático", ICMSC-USP, 1994.

[TAKU97] TAKUS, David A., David M. Profozich, "ARENA Software Tutorial", in Proceedings of 1997 Winter Simulation Conference, eds. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson.

[VIEI02] VIEIRA, Pedro, “Gerador Automático de Modelos de Simulação”, Relatório de Estágio da Licenciatura em Engenharia de Sistemas e Informática, Universidade do Minho, 2002.

[WAGN00] WAGNER, Flávio R., "Modelagem e Simulação de Sistemas Discretos", Universidade Federal do Rio Grande do Sul, 2000.

[WAGN00A] WAGNER, Marcus Vinícius da Silva. "Especificação de Componentes para Simulação de Redes Tcp/Ip", Campina Grande-PB, Brasil, Agosto de 2000.

Leituras Adicionais

Adriana Salvador Zanini
"Introdução a Modelagem e Simulação de Sistemas"
Sebenta de "Simulação de Sistemas", Universidade do Sul de Santa Catarina, 2004

Andrew F. Seila
"Spreadsheet Simulation"
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

Antonio Diaz-Calderon, Christiaan J. J. Paredis, Pradeep K. Khosla
"Organization And Selection Of Reconfigurable Models"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Arvind Mehta
"Smart Modeling? Basic Methodology And Advanced Tools"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Averill M. Law, Michael G. McComas
"How To Build Valid And Credible Simulation Models"
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

Averill M. Law, Michael G. McComas
"Simulation Of Manufacturing Systems"
Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

Berna Dengiz, Cigdem Alabas
"Simulation Optimization Using Tabu Search"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Boon Ping Gan, Li Liu, Sanjay Jain, Stephen J. Turner, Wentong Cai
"Distributed Supply Chain Simulation Across Enterprise Boundaries"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Candace L. Conwell, Rosemary Enright, Marcia A. Stutzman
"Capability Maturity Models Support Of Modeling And Simulation Verification, Validation, And Accreditation"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Christos Alexopoulos,Andrew F. Seila
"Output Analysis For Simulations"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

David Burnett, Todd LeBaron
"Efficiently Modeling Warehouse Systems "
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

David Goldsman, Bruce W. Schmeiser
"Computational Efficiency Of Batching Methods"
Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

David Goldsman, Gamze Tokol
"Output Analysis Procedures For Computer Simulations"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

David H. Withers
"Software Engineering Best Practices Applied To The Modeling Process"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Dirk Brade
"Enhancing Modeling And Simulation Accreditation By Structuring Verification And Validation Results"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Don Caughlin
"An Integrated Approach To Verification, Validation, And Accredition Of Models And Simulations"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Erwin Rybin, Felix Breitenecker
"Simulation Of A Production Plant In The Brick Industry"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Graça Bressan
"Modelagem e Simulação de Sistemas Computacionais"
EPUSP, 2002

Graça Bressan
"Modelos de Simulação de Acontecimentos Discretos"
EPUSP, 2002

Halim Damerdji, Shane G. Henderson
"Computational Efficiency Evaluation In Output Analysis"
Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

Henk de Swaan Arons, Eelco van Asperen
"Computer Assistance For Model Definition"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Hessam S. Sarjoughian, Bernard P. Zeigler
"Models And Representation Of Their Ownership"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Hui Zhao
"Simulation And Analysis Of Dealers' Returns Distribution Strategy"
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

J. Michael Knoll, Joseph A. Heim
"Ensuring The Successful Adoption Of Discrete Event Simulation In A Manufacturing Environment"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

James R. Swisher, Paul D. Hyden
"A Survey Of Simulation Optimization Techniques And Procedures"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

James Ritchie-Dunham, Douglas J. Morrice , Judy Scott, Edward G. Anderson
"A Strategic Supply Chain Simulation Model"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Jennifer Chew, Cindy Sullivan
"Verification, Validation, And Accreditation In The Life Cycle Of Models And Simulations"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

João Moura Pires
"Geração de números aleatórios e Simulação"
Sebenta de Métodos Quantitativos, Universidade nova de Lisboa, 2002/2003

John M. Charnes
"Using Simulation For Option Pricing"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

John R. Clymer
"Optimizing Production Work Flow Using Opemcss"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Julie N. Ehrlich, William R. Lilegdon
"Making Better Manufacturing Decisions With Aim"
Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

K. Guus C. de Ruiter, Joost M. Sluijs, Wilbert B. Stoutjesdijk
"Simulation For Recurring Decisions"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Leonardo Chwif, Marcos Ribeiro Pereira Barretto, Ray J. Paul
"On Simulation Model Complexity"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Martha A. Centeno,Manuel Carrillo
"Challenges Of Introducing Simulation As A Decision Making Tool"
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

Marvin S. Seppanen
"Developing Industrial Strength Simulation Models Using Visual Basic For Applications (VBA)"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Michael C. Ferris, Todd S. Munson, Krung Sinapiromsaran
"A Practical Approach To Sample-Path Simulation Optimization"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Michael C. Fu
"Simulation Optimization"
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

Neil H. Robertson, Terrence Perera
"Feasibility For Automatic Data Collection"
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

Osman Balci, William F. Ormsby, John T. Carr, III, Said D. Saadi
"Planning For Verification, Validation, And Accreditation Of Modeling And Simulation Applications"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Peter Mullarkey, Srinagesh Gavirneni
"Dynamic Output Analysis For Simulations Of Manufacturing Environments"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Ralph V. Rogers
"What Makes A Modeling And Simulation Professional?:The Consensus View From One Workshop"
Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

Richard P. Marek, Debra A. Elkins, Donald R. Smith
"Understanding The Fundamentals Of Kanban And Conwip Pull Systems Using Simulation"
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

Robert Entriken, Siegfried Vössner
"Genetic Algorithms With Cluster Analysis For Production Simulation"
Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

Robert G. Sargent
"Verification, Validation, And Accreditation Of Simulation Models"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Robert G. Sargent, Priscilla A. Glasow, Jack P.C. Kleijnen
"Strategic Directions In Verification, Validation, And Accreditation Research"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Rockwell Software
"Arena Basic User's Guide"
Rockwell Software 2000

Rockwell Software
"Arena Standard User's Guide"
Rockwell Software 2000

Rockwell Software
"Variables Guide"
Rockwell Software 2000

Roderick J. Swets, Glenn R. Drake
"The Arena Product Family: Enterprise Modeling Solutions"
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

Sarita Mazzini Bruschi
"Extensão do ASIA para Simulação de Arquitetura de Computadores"
São Carlos (SP), 1997

Scott Miller,Dennis Pegden
"Introduction To Manufacturing Simulation"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Shane G. Henderson
"Mathematics For Simulation"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Stewart Robinson, Thanos Alifantis, Robert Hurrion, John Edwards
"Modelling And Improving Human Decision Making With Simulation"
Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

Tarek M. Zayed, Daniel W. Halpin
"Simulation As A Tool For Resource Mana gement"
Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

Tony Dean
"A Method For Achieving Stable Distributions Of Wireless Mobile Location In Motion Simulations"
Proceedings of the 2000 Winter Simul tion Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

William S. Keezer
"Simulation Of Computer Systems And Applications"
Proceedings of the 1997 Winter Simulation Conference
eds. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

Young Hae Lee, Sook Han Kim
"Optimal Production-Distribution Planning In Supply Chain Management Using A Hybrid Simulation-Analytic Approach"
Proceedings of the 2000 Winter Simulation Conference