UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

SISTEMA DE APOIO AO COORDENADOR DO SIMULADOR DE EMPRESAS VIRTUAL UTILIZANDO A TECNOLOGIA CORBA

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

DOUGLAS THOMAS JACOBSEN

BLUMENAU, NOVEMBRO/2000

2000/1-23

SISTEMA DE APOIO AO COORDENADOR DO SIMULADOR DE EMPRESAS VIRTUAL UTILIZANDO A TECNOLOGIA CORBA

DOUGLAS THOMAS JACOBSEN

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Maurício Capobianco Lopes — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Maurício Capobianco Lopes

_____

Prof. Oscar Dalfovo

_____

Prof. Everaldo Artur Grahl

AGRADECIMENTOS

Agradeço, principalmente, aos meus pais, Lothar e Ilca Jacobsen, que sempre lutaram para proporcionar aos filhos tudo o que eles jamais tiveram para si. Sem o incentivo deles, eu jamais teria iniciado e concluído o curso.

Ao meu orientador, Professor Maurício Capobianco Lopes. A orientação deste meu Mestre foi primordial para meu aprendizado. Agradeço por ter tido o privilégio de ter sido seu orientando.

Ao professor Oscar Dalfovo por ter aberto as portas do conhecimento para mim. Sem ele, com certeza não teria chegado ao fim deste curso que venci com sacrifício, sacrifício este que trouxe uma grande realização profissional e pessoal.

Todos os sonhos podem se tornar realidade se realmente tivermos iniciativa, determinação, coragem e principalmente fé em Deus.

A todos os meus colegas de faculdade e amigos que obtive, que ajudaram a tornar este período de estudos estimulante e alegre.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

AO – Adaptador de Objeto

BOA – *Basic Object Adapter*

CORBA – Common Object Request Broker Architecture

COM – Component Object Model

DCOM – Distributed Component Object Model

DII – Interface de Invocação Dinâmica

DSI – Interface de Esqueleto Dinâmico

ESIOP – *Environment-Specific Inter-ORB Protocol*

GIOP – *General Inter-ORB Protocol*

IDL – *Interface Definition Language*

IIOP – *Internet Inter-ORB Protocol*

IR – Repositório de Interfaces

LAN – *Local Network Area*

MTS – *Microsoft® Transaction Server*

OMG – *Object Management Group*

ORB – *Object Request Broker*

UML – *Unified Modeling Language*

WAN – *Wide Network Area*

RESUMO

Este trabalho tem como objetivo implementar um módulo adicional ao sistema VIRTUAL, que é um sistema simulador do comportamento humano dentro de uma realidade empresarial. Este novo módulo será usado pelo coordenador do jogo. O módulo atua como servidor e se comunica com o sistema atual (dos participantes), que passa a ser o módulo cliente e com isso automatiza todo o controle de dados das equipes. A especificação e desenvolvimento foi realizada baseando-se no padrão CORBA de objetos distribuídos. A especificação do protótipo foi feita em UML - *Unified Modeling Language* utilizando a ferramenta Rational Rose e o desenvolvimento utilizando o ambiente de programação visual Delphi 5.0.

ABSTRACT

This work aims to implement an additional module to the VIRTUAL system to be used by the coordinator of the game. This module acts as a server and it communicates to the current system (of the participants) which becomes the module client and with that it automates the whole data control of the teams. The specification and development were accomplished based on CORBA method of distributed objects. The prototype specification was made in UML - Unified Modeling Language using Rational Rose tool and it was developed in visual programming Delphi 5.0 environment.

1 INTRODUÇÃO

Os atuais sistemas têm se tornado complexos, o que é ainda mais agravado diante dos requisitos solicitados pelas aplicações modernas: alta confiabilidade, alto desempenho, desenvolvimento do mesmo de maneira rápida e barata, entre outros.

Neste sentido, as pessoas que estudam métodos de desenvolvimento de sistemas tem tentado aproximá-los ao máximo da realidade, percebendo que os objetos no mundo real existem e estão dispostos por toda parte. Partindo deste pensamento nasceram os Objetos Distribuídos.

Objetos Distribuídos são peças de código que, uma vez unidas, formam um software distribuído. Essas peças podem ser objetos desenvolvidos de acordo com as regras da orientação a objetos, entretanto, apresentam serviços e propriedades que os difere dos objetos convencionais e os tornam distribuídos. A principal característica dos objetos distribuídos está na possibilidade de executar aplicativos orientados a objetos distribuídos em qualquer plataforma, e interagi-los com outros sistemas distribuídos, ou seja eles vêm solucionar os problemas de heterogeneidade entre softwares.

Além disso, os objetos distribuídos podem estar dispostos pelas estações de uma rede de computadores, seja ela uma *LAN* (rede local*), WAN* (rede metropolitana) ou Internet, possibilitando a concretização de ambientes distribuídos (ambientes caracterizados por terem seus computadores ligados por uma rede, trabalhando juntamente, como se fosse um único grande processador), aplicados principalmente em empresas que precisam disponibilizar seus sistemas 24 horas por dia.

Levando em consideração todos os aspectos e problemas citados anteriormente, foi desenvolvido o padrão CORBA (*Common Object Resquest Broker Arquiteture*). O padrão CORBA é um modelo proposto pela OMG (*Object Management Group*), com o propósito de promover a tecnologia dos objetos de forma distribuída. Seu objetivo principal é diminuir os custos, reduzir a complexidade, e proporcionar caminhos para o surgimento de novas aplicações a partir dos conceitos propostos pela OMG. O CORBA, resumidamente, propõe a interoperabilidade local ou remota, entre aplicações, independente das linguagens de programação em que foram desenvolvidas e sobre quais plataformas serão executadas.

Com a evolução desta tecnologia, sua aplicação tem se tornado possível em diversas áreas. Neste trabalho pretende-se apresentar o uso do padrão CORBA no simulador de empresas VIRTUAL.

O simulador de empresas VIRTUAL é um jogo geral, automatizado, que contempla áreas de gestão de produção, finanças e marketing e é utilizado em treinamento de habilidades na gestão de recursos humanos, porque permite, através de simulações, prever o comportamento humano dentro de uma realidade empresarial. Atualmente este sistema disponibiliza o mesmo módulo para os participantes e o coordenador do jogo. Deste modo o coordenador não conta com uma ferramenta que o auxilie no controle e acompanhamento dos resultados dos participantes.

Assim, o presente trabalho visa implementar um módulo adicional ao sistema VIRTUAL a ser usado pelo coordenador do jogo que atuará como servidor e irá se comunicar com o sistema atual (dos participantes, que passará a ser o módulo cliente), através da tecnologia CORBA de objetos distribuídos automatizando todo o controle de dados das equipes.

A especificação do protótipo foi feita em UML - *Unified Modeling Language*. A ferramenta utilizada para esta especificação foi o *Rational Rose*, devido aos recursos disponíveis para aplicar as representações da UML, como o Diagrama de Classes, o Diagrama de Casos de Uso e o Diagrama de Seqüência.

Para a implementação do protótipo foi utilizado o ambiente de programação Borland Delphi 5.0, pois este ambiente já suporta a tecnologia CORBA.

1.1 OBJETIVOS

Este trabalho tem como objetivo principal especificar e implementar um módulo para o software VIRTUAL, utilizando técnicas de processamento distribuído baseado no padrão CORBA de objetos distribuídos.

Os objetivos secundários do trabalho são:

a) facilitar e agilizar o uso do software VIRTUAL, por parte do coordenador;

b) verificar a efetividade do uso do padrão CORBA com *softwares* de automação, tal como o Excel, que é onde estão definidos os modelos e os dados do VIRTUAL.

1.2 ESTRUTURA

O trabalho foi estruturado da seguinte maneira:

O capítulo 2 apresenta a descrição dos Jogos de Empresas, suas características, dinâmicas, classificações e objetivos. Também demonstra o simulador de empresas VIRTUAL ao qual será implementado e adicionado o módulo servidor para controle das equipes.

O capítulo 3 apresenta os Objetos Distribuídos, seus protocolos de comunicação, linguagens que podem ser usadas para a sua implementação, vantagens e desvantagens de sua utilização e em especial o padrão CORBA com toda sua estrutura e componentes.

O capítulo 4 apresenta o desenvolvimento do protótipo do simulador, principalmente no que tange à implementação do módulo coordenador (servidor) utilizando o padrão CORBA para a comunicação entre os módulos clientes e o servidor. Será demonstrada a análise do sistema com os diagramas de caso de uso, seqüências e classes bem como todos os passos utilizados para o desenvolvimento do protótipo através da ferramenta Delphi 5 da Inprise e a tecnologia CORBA suportada por esta ferramenta.

O capítulo 5 apresenta as considerações finais, conclusões, dificuldades encontradas no decorrer do desenvolvimento do trabalho e sugestões para futuras implementações e extensões deste trabalho.

2 JOGOS DE EMPRESAS

Os jogos de empresas são modelos utilizados para treinamento de pessoas e têm como finalidade a formação de profissionais, sendo que permitem o treinamento e o aperfeiçoamento dos mesmos através de simulações que podem acontecer no dia a dia empresarial. As pessoas envolvidas nos jogos de empresas podem oferecer soluções e conhecer os resultados, administrando recursos próprios ou de terceiros.

Após a utilização deste tipo de jogo, nota-se nas corporações que as pessoas que fizeram parte do treinamento apresentam um aumento de conhecimentos e adquirem maior facilidade de utilizarem os conhecimentos obtidos durante a vivência na empresa.

Atualmente, os jogos empresariais são aplicados nos mais diversos campos: para treinamento e desenvolvimento de pessoal, avaliação de potencial, planejamento, tomada de decisão e formação de administradores.

2.1 OBJETIVOS DOS JOGOS DE EMPRESAS

Os principais objetivos dos jogos empresariais são ([MEN1997]):

a) treinamento: permitem aos integrantes das equipes, através de ambientes simulados, adquirirem habilidades e experiências na tomada de decisões que possam vir a acontecer no dia a dia;

b) didático: permitem transmitir, de modo eficaz, conhecimentos sobre diversas áreas empresariais, tais como, contabilidade, administração de empresas, economia, sistemas de informação, entre outros, proporcionando o modo prático e o experimental;

c) pesquisa: permitem descobrir soluções para problemas empresariais, pesquisar sobre as teorias de administração de empresas, estudar o relacionamento individual e grupal dos integrantes e verificar como as pessoas se comportam sob pressão de incertezas e tempo.

Além disto, eles têm sido utilizados para validar técnicas computacionais aplicadas a ambientes de gestão de negócios.

2.2 CARACTERÍSTICAS GERAIS DOS JOGOS DE EMPRESAS

Embora haja diversidade nos jogos empresariais, algumas características são encontradas em todos os jogos, sendo elas ([MEN1997]):

a) o meio ambiente simulado em que o jogo se desenvolve substitui elementos do sistema real. A possibilidade dos participantes avaliarem suas próprias decisões, é a idéia principal dos jogos empresariais;

b) sua representação é feita sob a forma de relações lógicas e/ou matemáticas.

2.3 DINÂMICA DOS JOGOS DE EMPRESAS

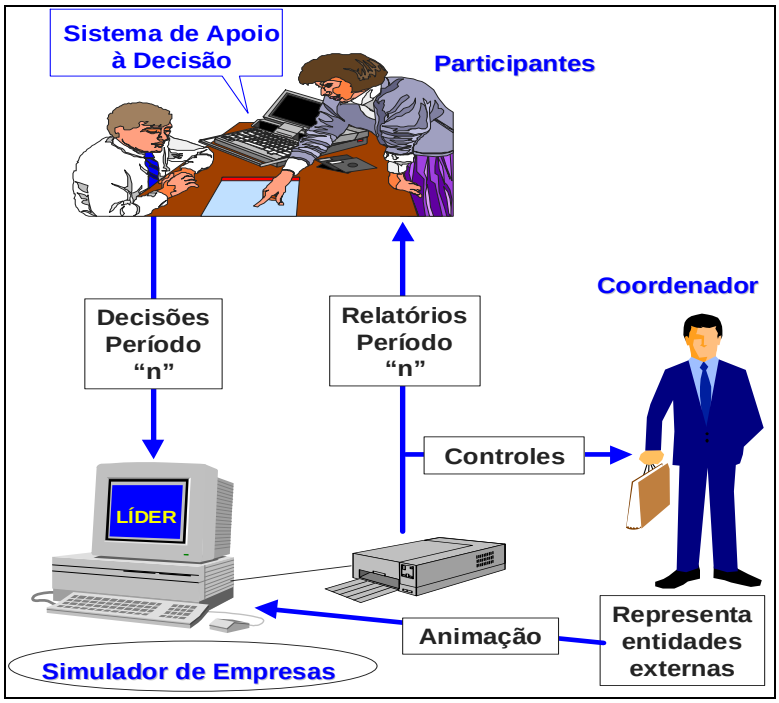
Segundo [MEN1997], “os participantes do jogo, cada um individualmente ou em grupo, administram a empresa (firma) como um todo ou uma parte dela, através de decisões administrativas por períodos sucessivos”.

Os jogos empresariais dividem-se em três fases denominadas de: fase preparatória, ciclo repetitivo e encerramento.

Na fase preparatória os participantes são distribuídos em equipes, normalmente de 3, 4 ou 5 participantes, dependo das características do jogo. Nesta etapa, sob a orientação de um coordenador, eles familiarizam-se com o meio ambiente simulado, analisam dados passados e desenvolvem um plano estratégico. Para isto lhe são fornecidos manuais e relatórios com informações completas sobre o jogo e suas regras de funcionamento.

A fase de ciclo repetitivo que está representada na figura 1, os participantes tomam suas decisões baseados em seus relatórios gerenciais ou, em alguns casos, em sistemas de apoio à decisão, desenvolvidos através de meios computacionais. As decisões são encaminhadas ao simulador que as processa e emite o resultado. O resultado pode ser fornecido através de relatórios gerenciais, ou através de arquivos de dados. Nesta etapa, o coordenador representa as entidades externas, como por exemplo, fornecedores, bancos, governo, acionistas, importadores, etc., cujas decisões também afetam os resultados das equipes. Este ciclo é repetido tantas vezes quanto o coordenador da simulação achar necessário, para um bom aproveitamento e aprendizagem do jogo.

Figura 1– Ciclo repetitivo



Na fase de encerramento, o coordenador e os participantes do jogo realizam uma assembléia onde procedem à avaliação do jogo, comentando estratégias e revendo seus erros e acertos ([MEN1997]).

2.4 CLASSIFICAÇÃO DOS JOGOS DE EMPRESAS

Os Jogos de Empresas podem ser classificados de acordo com sua natureza básica:

a) jogos sistêmicos: dão ênfase no funcionamento do sistema;

b) jogos humanos: visam tratar os problemas das variáveis humanas presentes nas negociações;

c) jogos mistos: intervêm componentes sistêmicos e humanos.

Para [KOP1989] o que é levado em conta nos Jogos de Empresas são o tipo de simulação:

a) jogos gerais: simulam uma empresa como um todo;

b) jogos específicos: feitos a partir da modelagem de uma empresa particular;

c) jogos setoriais: simulam empresas de um setor da economia;

d) jogos funcionais: voltados a uma área específica dentro da empresa.

Para [TAN1977] os Jogos de Empresas podem ser classificados segundo:

a) meio de apuração dos resultados;

b) a interação entre as equipes.

Os Jogos de Empresas também podem ser classificados pelas habilidades envolvidas ([GRA1993]):

a) jogos de comportamento: enfatizam habilidades comportamentais;

b) jogos de processo: enfatizam habilidades técnicas;

c) jogos de mercado: enfatizam habilidades técnicas e de mercado.

2.5 SIMULADOR DE EMPRESAS VIRTUAL

O simulador de empresas VIRTUAL, desenvolvido pelos professores Maurício Capobianco Lopes e Pedro Paulo Hugo Wilhelm na Universidade Regional de Blumenau, é um jogo geral, totalmente computadorizado, que contempla áreas de gestão de produção, finanças e marketing.

O cenário onde ocorre a simulação é uma empresa que opera em condições normais, podendo produzir três tipos de produtos (A, B e C) a partir de três matérias primas (I, II e III). Internamente, existem atividades, tais como, controlar o fluxo de produção, estoques e finanças. Externamente ocorre as relações entre fornecedores, banco, governo, consultores e importadores.

O processo decisório envolve um amplo conjunto de atividades, entre elas destacam-se:

a) produção:

compras e estoques de matérias-primas;

programação das horas de atividades de produção para o mês;

contratação e demissão de pessoal;

política salarial;

gastos com manutenção de equipamentos;

b) mercado:

formação de preços e prazos de parcelamento;

gastos com propagandas;

propostas de exportação;

c) financeiro:

política de juros e descontos nas vendas;

política de prazo nas compras;

empréstimos de curto e longo prazo;

aplicações em renda fixa ou fundo de ações;

d) geral:

metas globais de rentabilidade e vendas;

novos investimentos.

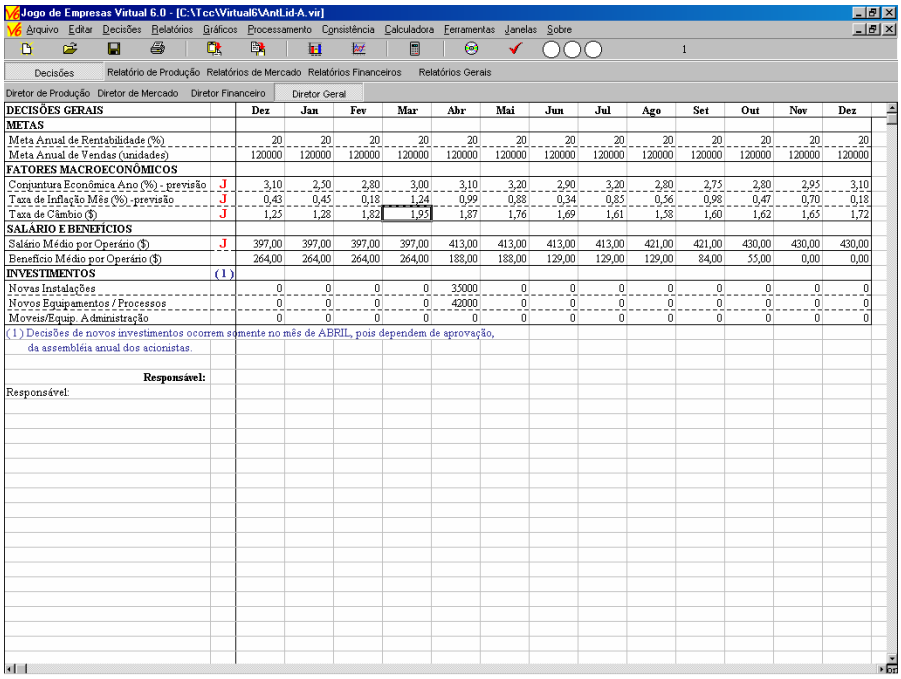
Cada uma destas atividades (produção, mercado, financeiro e geral) é de responsabilidade de um diretor, que são os participantes do jogo. Também cabe aos diretores simular cenários otimistas ou pessimistas, definindo possíveis sazonalidades, inflação, crescimento econômico, etc., visando reduzir o impacto de situações inesperadas sobre a empresa simulada.

A interface do simulador de empresas VIRTUAL foi desenvolvida no ambiente de programação Delphi. Os modelos de processamento e o banco de dados deste sistema são feitos através da planilha Excel.

A figura 2 mostra a tela principal do VIRTUAL onde pode ser observada a forma como o simulador apresenta os dados aos participantes. Os participantes analisam o problema e metas a serem alcançadas e alteram os valores propostos inicialmente pelo coordenador baseados nas suas próprias decisões.

No exemplo da figura 2 está sendo mostrada a tela onde o diretor geral toma suas decisões estabelecendo as metas a serem alcançadas e verificando os fatores macroeconômicos, salários e investimentos a serem realizados.

Figura 2 – Tela principal do simulador de empresas VIRTUAL



DECISÕES GERAIS		Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
METAS														
Meta Anual de Rentabilidade (%)		20	20	20	20	20	20	20	20	20	20	20	20	20
Meta Anual de Vendas (unidades)		120000	120000	120000	120000	120000	120000	120000	120000	120000	120000	120000	120000	120000
FATORES MACROECONÔMICOS														
Conjuntura Econômica Ano (%) - previsão	J	3,10	2,50	2,80	3,00	3,10	3,20	2,90	3,20	2,80	2,75	2,80	2,95	3,10
Taxa de Inflação Mês (%) -previsão	J	0,43	0,45	0,18	1,24	0,99	0,88	0,34	0,85	0,56	0,98	0,47	0,70	0,18
Taxa de Câmbio ($)	J	1,25	1,28	1,82	1,95	1,87	1,76	1,69	1,61	1,58	1,60	1,62	1,65	1,72
SALÁRIO E BENEFÍCIOS														
Salário Médio por Operário ($)	J	397,00	397,00	397,00	397,00	413,00	413,00	413,00	413,00	421,00	421,00	430,00	430,00	430,00
Benefício Médio por Operário ($)		264,00	264,00	264,00	264,00	188,00	188,00	129,00	129,00	129,00	84,00	55,00	0,00	0,00
INVESTIMENTOS	(1)													
Novas Instalações		0	0	0	0	35000	0	0	0	0	0	0	0	0
Novos Equipamentos / Processos		0	0	0	0	42000	0	0	0	0	0	0	0	0
Moveis/Equip. Administração		0	0	0	0	0	0	0	0	0	0	0	0	0

(1) Decisões de novos investimentos ocorrem somente no mês de ABRIL, pois dependem de aprovação, da assembléia anual dos acionistas.

Responsável:

Responsável:

A seguir serão demonstradas todas as opções do simulador de empresas VIRTUAL disponíveis no menu do sistema:

a) Arquivo

Novo: inicializa e cria um novo jogo;

Abrir: abre um novo jogo;

Salvar: salva a planilha atual;

Salvar como... : salva a planilha atual com outro nome;

Fechar: fecha a planilha atual;

Imprimir ...: imprime a planilha atual;

Processar: processa as informações do mês atual. (com o novo módulo, após o processamento as informações serão enviadas para o servidor);

Sair: fecha o sistema.

b) Editar

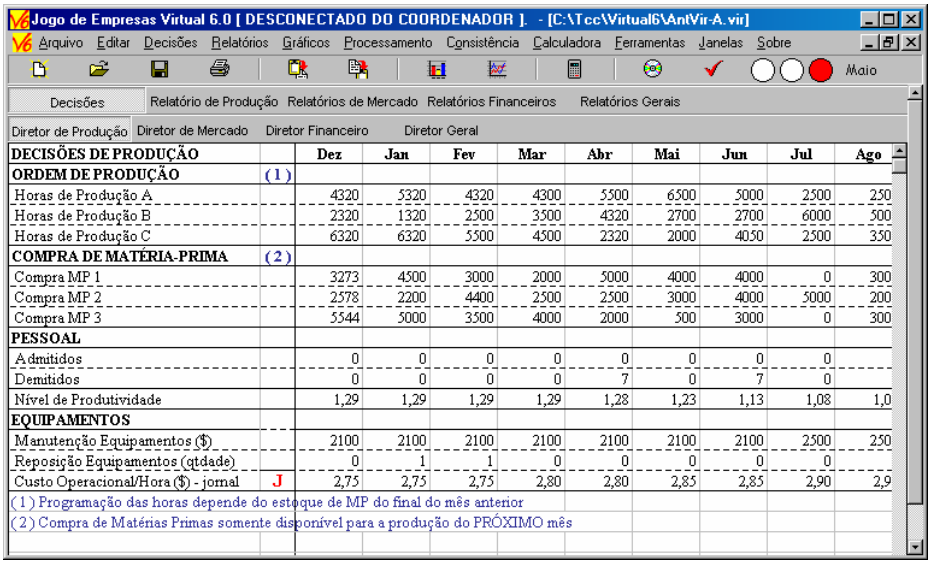
Copiar: copia as informações selecionadas na planilha;

Colar: restaura as informações copiadas na planilha.

c) Decisões

Produção: abre a planilha do diretor de produção (figura 3) para acesso às decisões referentes a produção (ordem de produção, matéria prima, pessoal e equipamentos);

Figura 3 – Decisões do diretor de produção

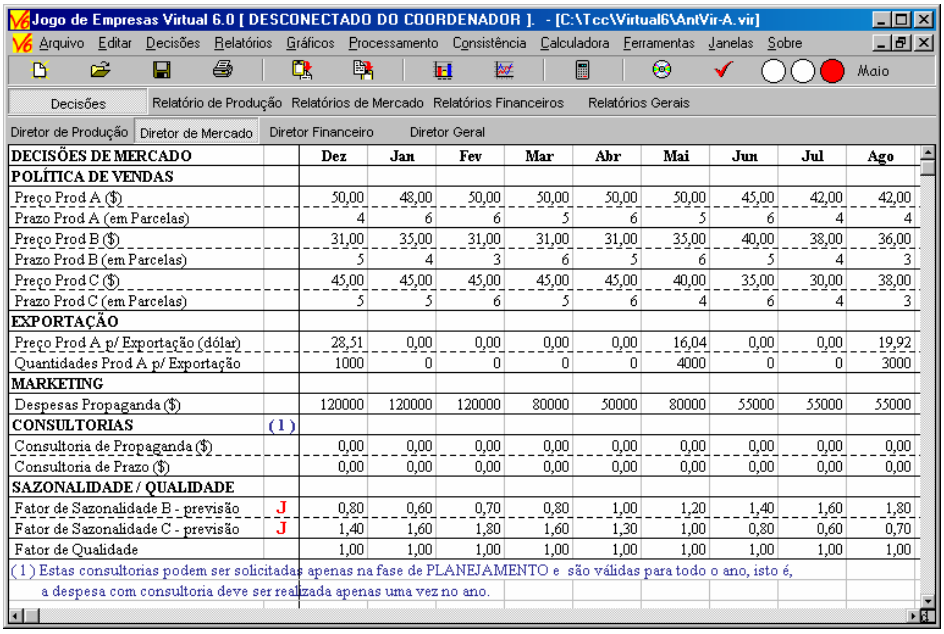


DECISÕES DE PRODUÇÃO		Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago
ORDEM DE PRODUÇÃO	(1)									
Horas de Produção A		4320	5320	4320	4300	5500	6500	5000	2500	250
Horas de Produção B		2320	1320	2500	3500	4320	2700	2700	6000	500
Horas de Produção C		6320	6320	5500	4500	2320	2000	4050	2500	350
COMPRA DE MATÉRIA-PRIMA	(2)									
Compra MP 1		3273	4500	3000	2000	5000	4000	4000	0	300
Compra MP 2		2578	2200	4400	2500	2500	3000	4000	5000	200
Compra MP 3		5544	5000	3500	4000	2000	500	3000	0	300
PESSOAL										
Admitidos		0	0	0	0	0	0	0	0	
Demitidos		0	0	0	0	7	0	7	0	
Nível de Produtividade		1,29	1,29	1,29	1,29	1,28	1,23	1,13	1,08	1,0
EQUIPAMENTOS										
Manutenção Equipamentos ($)		2100	2100	2100	2100	2100	2100	2100	2500	250
Reposição Equipamentos (qtdade)		0	1	1	0	0	0	0	0	
Custo Operacional/Hora ($) - jornal	J	2,75	2,75	2,75	2,80	2,80	2,85	2,85	2,90	2,9

(1) Programação das horas depende do estoque de MP do final do mês anterior
(2) Compra de Matérias Primas somente disponível para a produção do PRÓXIMO mês

Mercado: abre a planilha do diretor de mercado (figura 4) para acesso às decisões referentes ao mercado (vendas, exportação, marketing, consultorias e sazonalidade);

Figura 4 – Decisões do diretor de mercado

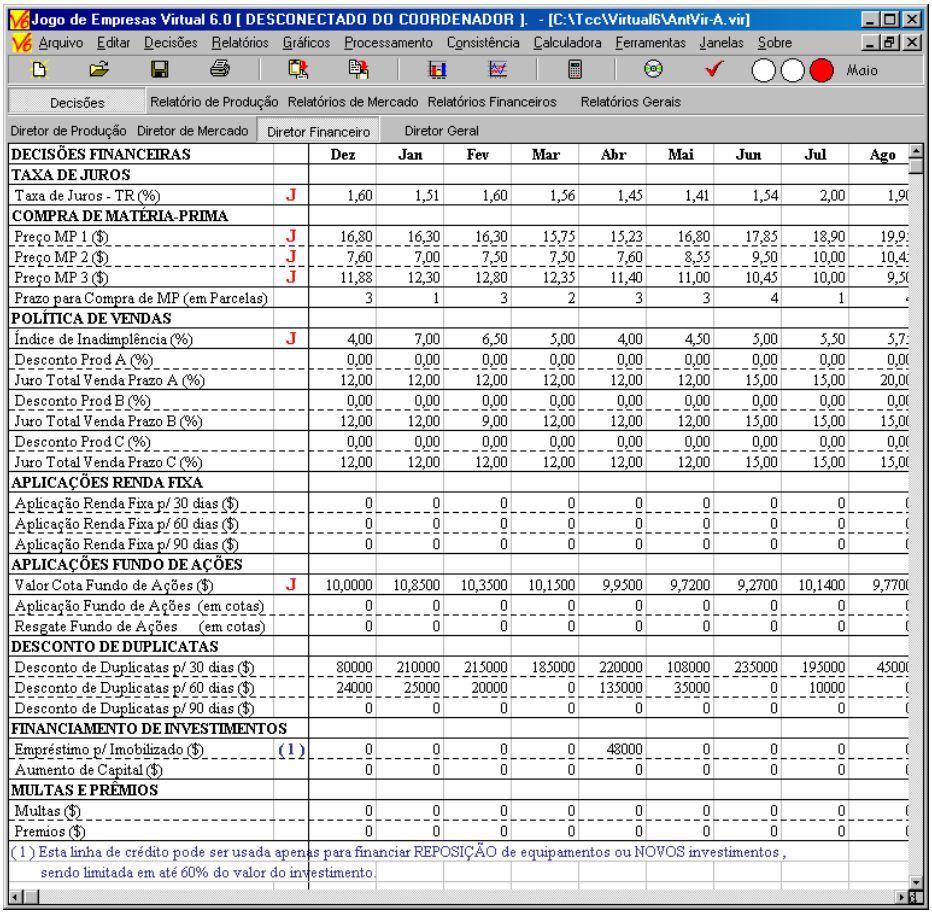


DECISÕES DE MERCADO		Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago
POLÍTICA DE VENDAS										
Preço Prod A ($)		50,00	48,00	50,00	50,00	50,00	50,00	45,00	42,00	42,00
Prazo Prod A (em Parcelas)		4	6	6	5	6	5	6	4	4
Preço Prod B ($)		31,00	35,00	31,00	31,00	31,00	35,00	40,00	38,00	36,00
Prazo Prod B (em Parcelas)		5	4	3	6	5	6	5	4	3
Preço Prod C ($)		45,00	45,00	45,00	45,00	45,00	40,00	35,00	30,00	38,00
Prazo Prod C (em Parcelas)		5	5	6	5	6	4	6	4	3
EXPORTAÇÃO										
Preço Prod A p/ Exportação (dólar)		28,51	0,00	0,00	0,00	0,00	16,04	0,00	0,00	19,92
Quantidades Prod A p/ Exportação		1000	0	0	0	0	4000	0	0	3000
MARKETING										
Despesas Propaganda ($)		120000	120000	120000	80000	50000	80000	55000	55000	55000
CONSULTORIAS	(1)									
Consultoria de Propaganda ($)		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Consultoria de Prazo ($)		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
SAZONALIDADE / QUALIDADE										
Fator de Sazonalidade B - previsão	J	0,80	0,60	0,70	0,80	1,00	1,20	1,40	1,60	1,80
Fator de Sazonalidade C - previsão	J	1,40	1,60	1,80	1,60	1,30	1,00	0,80	0,60	0,70
Fator de Qualidade		1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

(1) Estas consultorias podem ser solicitadas apenas na fase de PLANEJAMENTO e são válidas para todo o ano, isto é, a despesa com consultoria deve ser realizada apenas uma vez no ano.

Financeiro: abre a planilha do diretor financeiro (figura 5) para acesso às decisões financeiras a serem tomadas (taxa de juros praticada, compra de matéria prima, política de vendas, aplicações de renda fixa, ações, desconto de duplicatas, financiamentos, multas e prêmios);

Figura 5 – Decisões do diretor financeiro



Jogo de Empresas Virtual 6.0 [DESCONECTADO DO COORDENADOR]. - [C:\Tcc\Virtual6\AntVir-A.vir]

Arquivo Editar Decisões Relatórios Gráficos Processamento Consistência Calculadora Ferramentas Janelas Sobre

Maio

Decisões | Relatório de Produção | Relatórios de Mercado | Relatórios Financeiros | Relatórios Gerais

Diretor de Produção | Diretor de Mercado | Diretor Financeiro | Diretor Geral

DECISÕES FINANCEIRAS		Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago
TAXA DE JUROS										
Taxa de Juros - TR (%)	J	1,60	1,51	1,60	1,56	1,45	1,41	1,54	2,00	1,9(
COMPRA DE MATÉRIA-PRIMA										
Preço MP 1 ($)	J	16,80	16,30	16,30	15,75	15,23	16,80	17,85	18,90	19,9:
Preço MP 2 ($)	J	7,60	7,00	7,50	7,50	7,60	8,55	9,50	10,00	10,4:
Preço MP 3 ($)	J	11,88	12,30	12,80	12,35	11,40	11,00	10,45	10,00	9,5(
Prazo para Compra de MP (em Parcelas)		3	1	3	2	3	3	4	1	
POLÍTICA DE VENDAS										
Índice de Inadimplência (%)	J	4,00	7,00	6,50	5,00	4,00	4,50	5,00	5,50	5,7:
Desconto Prod A (%)		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,0(
Juro Total Venda Prazo A (%)		12,00	12,00	12,00	12,00	12,00	12,00	15,00	15,00	20,0(
Desconto Prod B (%)		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,0(
Juro Total Venda Prazo B (%)		12,00	12,00	9,00	12,00	12,00	12,00	15,00	15,00	15,0(
Desconto Prod C (%)		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,0(
Juro Total Venda Prazo C (%)		12,00	12,00	12,00	12,00	12,00	12,00	15,00	15,00	15,0(
APLICAÇÕES RENDA FIXA										
Aplicação Renda Fixa p/ 30 dias ($)		0	0	0	0	0	0	0	0	(
Aplicação Renda Fixa p/ 60 dias ($)		0	0	0	0	0	0	0	0	(
Aplicação Renda Fixa p/ 90 dias ($)		0	0	0	0	0	0	0	0	(
APLICAÇÕES FUNDO DE AÇÕES										
Valor Cota Fundo de Ações ($)	J	10,0000	10,8500	10,3500	10,1500	9,9500	9,7200	9,2700	10,1400	9,770(
Aplicação Fundo de Ações (em cotas)		0	0	0	0	0	0	0	0	(
Resgate Fundo de Ações (em cotas)		0	0	0	0	0	0	0	0	(
DESCONTO DE DUPLICATAS										
Desconto de Duplicatas p/ 30 dias ($)		80000	210000	215000	185000	220000	108000	235000	195000	4500(
Desconto de Duplicatas p/ 60 dias ($)		24000	25000	20000	0	135000	35000	0	10000	(
Desconto de Duplicatas p/ 90 dias ($)		0	0	0	0	0	0	0	0	(
FINANCIAMENTO DE INVESTIMENTOS										
Empréstimo p/ Imobilizado ($)	(1)	0	0	0	0	48000	0	0	0	(
Aumento de Capital ($)		0	0	0	0	0	0	0	0	(
MULTAS E PRÊMIOS										
Multas ($)		0	0	0	0	0	0	0	0	(
Premios ($)		0	0	0	0	0	0	0	0	(

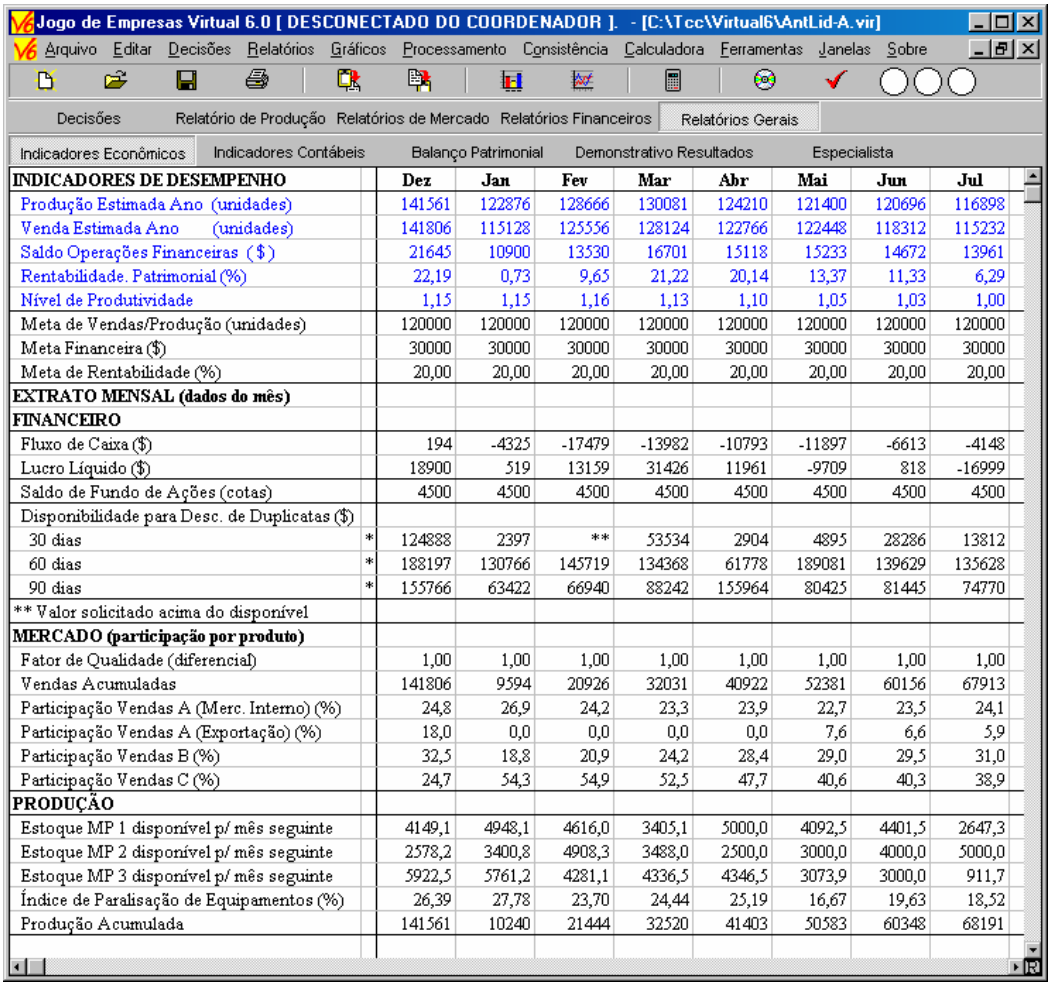
(1) Esta linha de crédito pode ser usada apenas para financiar REPOSIÇÃO de equipamentos ou NOVOS investimentos, sendo limitada em até 60% do valor do investimento.

Geral: abre a planilha do diretor geral para acesso às decisões gerais (metas, fatores macroeconômicos, salários, benefícios e investimentos) conforme mostrado anteriormente na figura 2.

d) Relatórios

Demonstra os relatórios de produção, mercado (mercado e formação de preços), financeiro (fluxo de caixa, contas a pagar e contas a receber) e gerais (indicadores, contábeis, balanço patrimonial, demonstrativo de resultados e especialista). A figura 6 mostra um exemplo de um relatório de indicadores econômicos.

Figura 6 – Relatório de indicadores econômicos

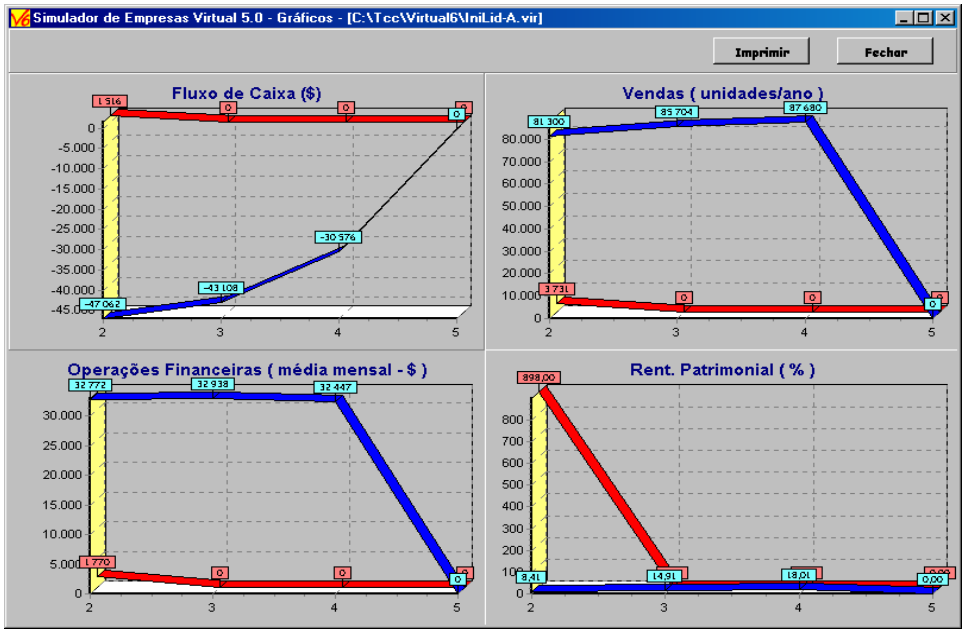


INDICADORES DE DESEMPENHO		Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul
Produção Estimada Ano (unidades)		141561	122876	128666	130081	124210	121400	120696	116898
Venda Estimada Ano (unidades)		141806	115128	125556	128124	122766	122448	118312	115232
Saldo Operações Financeiras ($)		21645	10900	13530	16701	15118	15233	14672	13961
Rentabilidade. Patrimonial (%)		22,19	0,73	9,65	21,22	20,14	13,37	11,33	6,29
Nível de Produtividade		1,15	1,15	1,16	1,13	1,10	1,05	1,03	1,00
Meta de Vendas/Produção (unidades)		120000	120000	120000	120000	120000	120000	120000	120000
Meta Financeira ($)		30000	30000	30000	30000	30000	30000	30000	30000
Meta de Rentabilidade (%)		20,00	20,00	20,00	20,00	20,00	20,00	20,00	20,00
EXTRATO MENSAL (dados do mês)									
FINANCEIRO									
Fluxo de Caixa ($)		194	-4325	-17479	-13982	-10793	-11897	-6613	-4148
Lucro Líquido ($)		18900	519	13159	31426	11961	-9709	818	-16999
Saldo de Fundo de Ações (cotas)		4500	4500	4500	4500	4500	4500	4500	4500
Disponibilidade para Desc. de Duplicatas ($)									
30 dias	*	124888	2397	**	53534	2904	4895	28286	13812
60 dias	*	188197	130766	145719	134368	61778	189081	139629	135628
90 dias	*	155766	63422	66940	88242	155964	80425	81445	74770
** Valor solicitado acima do disponível									
MERCADO (participação por produto)									
Fator de Qualidade (diferencial)		1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Vendas Acumuladas		141806	9594	20926	32031	40922	52381	60156	67913
Participação Vendas A (Merc. Interno) (%)		24,8	26,9	24,2	23,3	23,9	22,7	23,5	24,1
Participação Vendas A (Exportação) (%)		18,0	0,0	0,0	0,0	0,0	7,6	6,6	5,9
Participação Vendas B (%)		32,5	18,8	20,9	24,2	28,4	29,0	29,5	31,0
Participação Vendas C (%)		24,7	54,3	54,9	52,5	47,7	40,6	40,3	38,9
PRODUÇÃO									
Estoque MP 1 disponível p/ mês seguinte		4149,1	4948,1	4616,0	3405,1	5000,0	4092,5	4401,5	2647,3
Estoque MP 2 disponível p/ mês seguinte		2578,2	3400,8	4908,3	3488,0	2500,0	3000,0	4000,0	5000,0
Estoque MP 3 disponível p/ mês seguinte		5922,5	5761,2	4281,1	4336,5	4346,5	3073,9	3000,0	911,7
Índice de Paralisação de Equipamentos (%)		26,39	27,78	23,70	24,44	25,19	16,67	19,63	18,52
Produção Acumulada		141561	10240	21444	32520	41403	50583	60348	68191

e) Gráficos

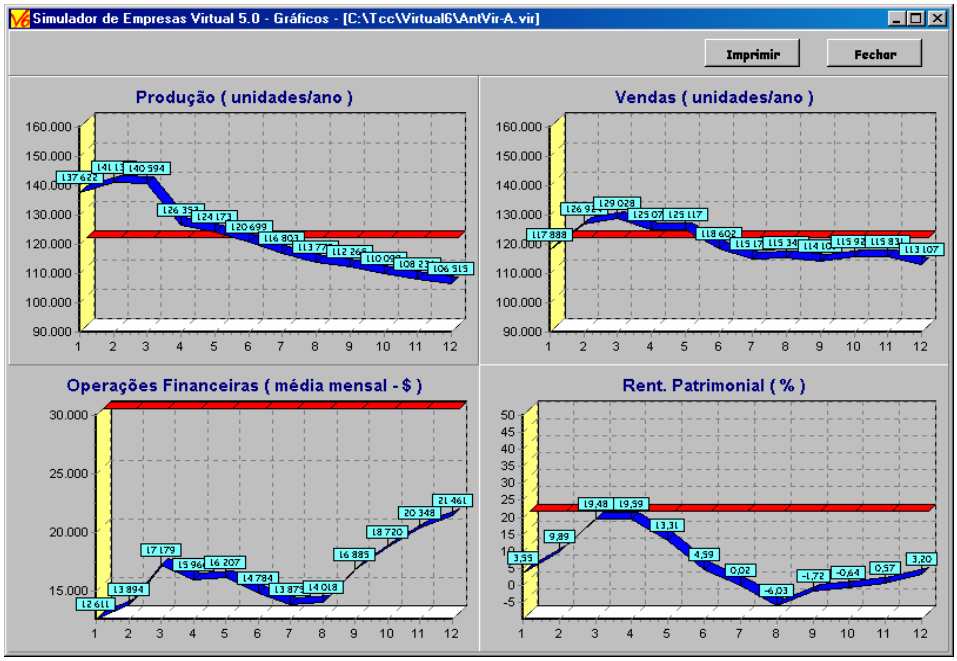
Orçado X Realizado: a figura 7 mostra os gráficos da posição atual das metas orçadas pelos participantes comparadas com as que foram realmente obtidas. O gráfico do canto superior esquerdo demonstra o orçamento do fluxo de caixa, o do superior direito demonstra as vendas efetuadas no período e os dois últimos gráficos (inferiores), mostram as médias de operações financeiras e a rentabilidade patrimonial decorrente dos processos realizados no período.

Figura 7 - Gráfico de Orçado X Realizado



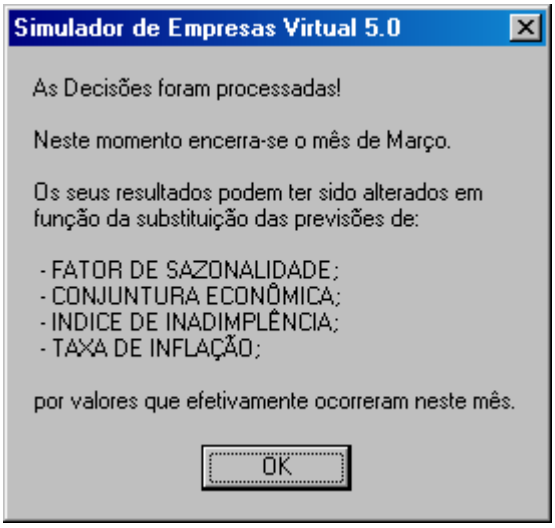
Metas X Realizado: a figura 8 mostra os gráficos da posição (desvio) atual das metas estabelecidas pelos participantes. O gráfico do canto superior esquerdo mostra as unidas que foram produzidas no período, o do superior direito a quantidade de produtos que foi vendida no mesmo período e os dois últimos gráficos (inferiores), mostram os rendimentos de aplicações e operações financeiras e a rentabilidade patrimonial decorrente dos processos realizados.

Figura 8 – Gráfico de Metas X Realizado



f) Processamento : processa as informações do mês corrente fechando este mês. Após o processamento, os valores dos meses anteriores ao atual não podem mais ser alterados, pois já foram processados. A figura 9 demonstra a mensagem que é informada após o processamento de um mês. Neste exemplo o mês atual (março) foi processado, portanto o mês de abril passa a ser o mês atual.

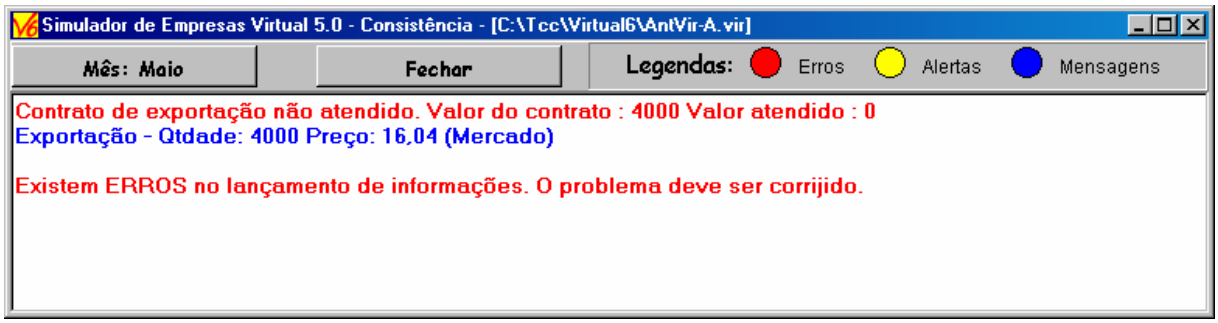
Figura 9 – Informações sobre o processamento



g) Consistência

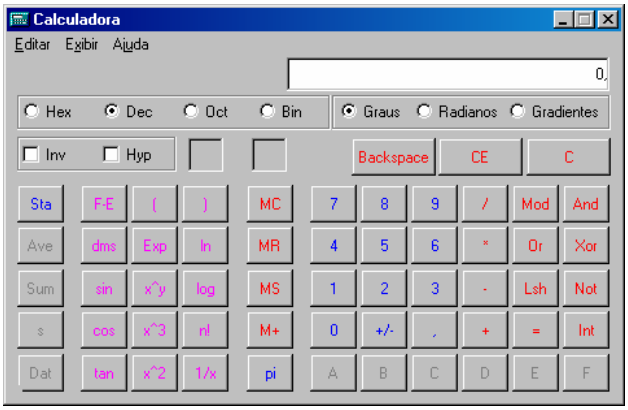
Conforme mostrado na figura 10, o VIRTUAL possui um modo de verificação dos dados informados pelos participantes do jogo. Ele consiste na verificação da autenticidade das informações inseridas, demostrando os erros (em vermelho), alertas (em amarelo) e mensagens (em azul) aos jogadores para uma melhor compreensão e verificação do que está sendo feito no decorrer do jogo.

Figura 10 - Mensagens de consistência geradas



h) Calculadora : exibe a calculadora padrão do sistema operacional conforme figura 11 abaixo.

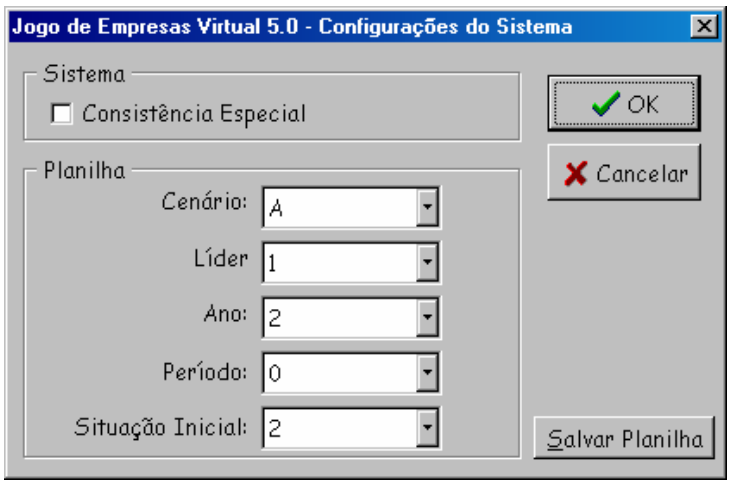
Figura 11 – Calculadora padrão do sistema operacional



i) Ferramentas

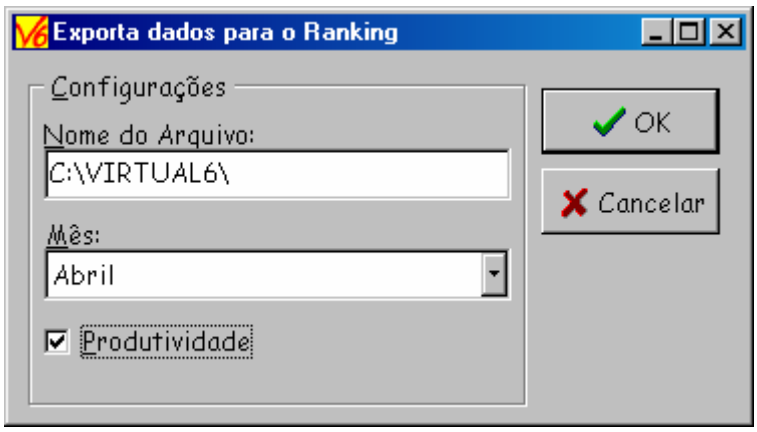
Facilitador: mostra as configurações do sistema (figura 12). Estas configurações passam a ser controladas pelo coordenador do jogo através do módulo servidor.

Figura 12 – Tela de configurações do sistema



Exportar Ranking: cria uma arquivo utilizado para o cálculo manual do ranking das equipes participantes. Mostrada na figura 13, esta função será automatizada pelo módulo servidor que irá demostrar automaticamente o ranking para o coordenador do jogo quando todas as equipes processarem o mês corrente.

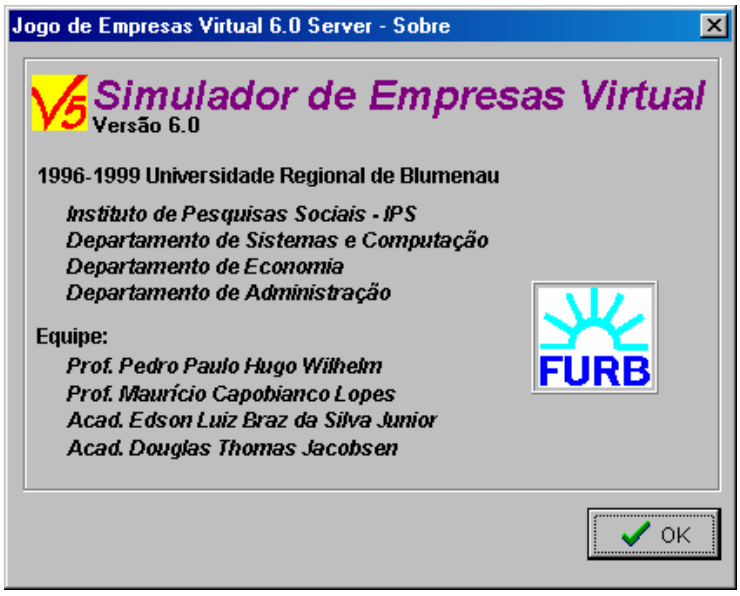
Figura 13 – Função de criação do arquivo para cálculo manual de ranking



j) Janelas : utilizado para demonstrar as janelas abertas de diferentes formas (lado a lado, cascata, organizar, maximizar e minimizar).

l) Sobre : tela com as informações sobre a versão e desenvolvedores do sistema VIRTUAL demonstrada na figura 14.

Figura 14 – Tela com informações sobre o VIRTUAL



Este capítulo especificou as características dos jogos de empresas e um demonstrativo das funções do simulador de empresas VIRTUAL. Para automatizar as funções deste sistema e torná-lo distribuído, facilitando assim sua comunicação com o módulo servidor, os objetos distribuídos (em especial o padrão CORBA que foi o escolhido) tem um papel fundamental, papel este e características que serão explanadas no capítulo seguinte.

3 OBJETOS DISTRIBUÍDOS

Objetos Distribuídos são estruturas que executam uma determinada tarefa. Sua principal característica refere-se à sua localização, eles podem ser abrigados todos em uma única máquina, ou distribuídos em máquinas distintas de uma rede de computadores LAN, WAN, ou Internet. Em conjunto, esses objetos são capazes de executar funções para um sistema (sistema distribuído) ([KRU1999]).

Objetos distribuídos são uma evolução dos objetos convencionais. Entretanto possuem uma interface com vários serviços de comunicação permitindo o seu acesso por outros objetos, de maneira que o objeto que o solicite desconheça o local onde o objeto chamado está abrigado, o sistema operacional que está sendo utilizado e a linguagem onde foi criado.

Um exemplo de aplicação de objetos distribuídos pode ser um programa em uma estação qualquer de uma rede. De repente é solicitado um serviço que determina alguma operação que este programa terá que executar. Para que o serviço seja atendido, o programa verificará quais são os objetos caracterizados por fazerem a operação solicitada, e envia um chamado à estes objetos, sendo que o programa não sabe onde estes objetos estão localizados na rede. Uma tecnologia, inerente ao objeto encontrará e o disponibilizará para o programa executar a tarefa solicitada.

Para que um programa formado por objetos distribuídos possa fazer uma solicitação a um objeto localizado no mesmo computador do sistema em execução, ou em qualquer estação de uma rede, desconhecendo o local onde o objeto a ser solicitado está presente, é necessário uma ponte de comunicação entre os componentes distribuídos, chamado de *middleware.* Este *middleware* é um padrão de protocolos e tecnologia desenvolvido cada um com um grau de abrangência.

Quando desenvolve-se um objeto distribuído, os serviços adicionais que o mesmo possui, dizem respeito justamente ao *middleware*.

Os padrões de objetos distribuídos mais conhecidos são:

a) DCOM - *Distibuted Component Object Mode:* é uma versão distribuída do modelo COM da Microsoft que promove comunicação entre componentes distribuídos em uma rede de computadores. Esses componentes podem se comunicar independente

da linguagem em que foram desenvolvidos e até linguagens de outros fabricantes, e podem estar abrigados em estações utilizando diferentes sistemas operacionais, dentro da abrangência da tecnologia.

b) CORBA – *Common Object Resquest Broker Arquiteture* – tecnologia desenvolvida pela OMG (*Object Management Group*). Tem o mesmo princípio que o DCOM, porém não tem limitação a diversidade de plataformas.

Para desenvolver objetos distribuídos, utilizam-se linguagens específicas, onde cada uma proporciona uma solução diferente. As linguagens mais conhecidas que implementam objetos DCOM são: Java, Microsoft Visual Basic, Microsoft Visual C++, PowerBuilder, Micro Focus Visual Object COBOL e Delphi. As linguagens que implementam objetos CORBA são: Delphi, Java, C, C++, Ada e Smaltalk.

3.1 VANTAGENS E DESVANTAGENS

Os objetos distribuídos vêm solucionar problemas de comunicação entre softwares de diferentes plataformas e desenvolvidos em linguagens diferentes, já que um sistema distribuído pode ser executado em qualquer plataforma.

Segundo [KRU1999], “a orientação a objetos permite a reutilização e recombinação de códigos e esta vantagem é ainda mais bem-vinda quando trata-se de objetos distribuídos, uma vez que, além de poder utilizar um objeto já existente, dispensando então a necessidade de criar novamente um mesmo objeto, não é preciso ficar dependente da linguagem à qual ele fora desenvolvido, sendo permitido comunicar e utilizar objetos criados em linguagens distintas. O objeto ainda pode ser personalizado de acordo com a aplicação e a necessidade do programador. Pode também ser utilizado por sistemas diferentes, tendo uma interface distinta para cada sistema, mas a mesma funcionalidade”.

Outras vantagens dos objetos distribuídos são a maior agilidade na comunicação entre o cliente e o servidor (já que os serviços já são conhecidos), redução de *overhead*, e não supercarregamento de um único computador. Com um sistema baseado em objetos distribuídos, e estes distribuídos pelas estações de uma rede, faz com que diminua a quantidade de usuários conectados a um único servidor, pois dificilmente todos estarão executando ao mesmo tempo as operações cabíveis a um único objeto.

O desenvolvedor não é obrigado a construir todos os objetos numa mesma estação da rede. Um problema que pode inviabilizar o uso de um sistema distribuído é a perda de comunicação na rede. Se a rede não estiver funcionando adequadamente, não haverá como utilizar o sistema, no caso dos objetos estarem espalhados pelas estações da mesma. Ou, se por algum motivo uma das estações da rede falhar, o objeto presente nesta estação estará indisponível, e com ele as funções que é responsável. Um meio para evitar este tipo de problema, é a duplicação de objetos em máquinas distintas da rede.

Ainda é possível contar com os recursos dos protocolos de comunicação que implementam meios de tolerância à falhas que ajudam a dissolver problemas que podem vir a ocorrer com a perda de conexão no momento em que dois objetos estão se comunicando.

Porém para maior segurança na hora de implementar um sistema distribuído numa rede, é preciso certificar-se de que a mesma é segura e bem projetada.

3.2 CORBA

Segundo [KRU1999], “a aplicação do modelo CORBA acontece para promover a intercomunicação de objetos distribuídos em uma rede de computadores, a fim de executar alguma tarefa”.

Objetos distribuídos CORBA são pequenas partes de códigos de um sistema maior, onde essas pequenas partes estão presentes em algum lugar da rede, e apresentam-se como componentes binários que podem ser acessados por objetos clientes remotos ou não, por meios de métodos de invocação.

Para criar um objeto, utiliza-se uma linguagem de alto nível para escrever o código do objeto, como C, C++, Java, Smaltalk e Ada, e o compila utilizando uma linguagem definida na especificação do CORBA, a IDL (*Interface Definition Language*). IDL é a linguagem adotada para especificar toda a sintaxe de todas as interfaces dos objetos da OMG como será visto posteriormente.

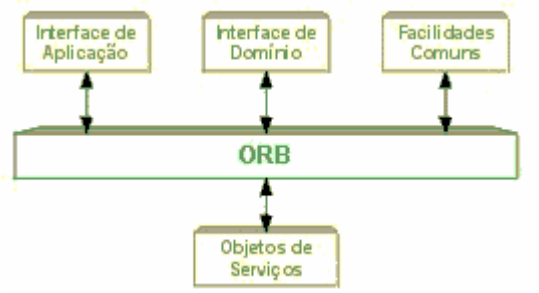
O CORBA dispõe de um meio que padroniza os objetos, permitindo sua comunicação: a utilização de interfaces. Cada objeto tem uma interface definida que deve conhecer e requisitar um serviço a outro objeto. Dessa maneira, não é necessário saber detalhes de sua

implementação (código) para que o objeto possa ser acessado. Para especificar essa interface dos objetos, a OMG utiliza a IDL para definir um padrão entre os objetos.

Com a definição das interfaces, o objeto cliente acessa um outro objeto pela execução de uma requisição (evento) ao mesmo, o qual tem suas características conhecidas por todos através de sua interface, que por sua vez é reconhecida por ser gerada a partir do padrão IDL, utilizado por todos os objetos da OMG, o que permite a interação entre diferentes objetos.

A figura 15 demonstra melhor a arquitetura do modelo de referência da OMG.

Figura 15 - Arquitetura do modelo de referência da OMG.



Fonte: [VAS1998, p.3]

3.2.1 OBJETOS DE SERVIÇOS

Os objetos de serviço, nada mais são do que coleções de serviços (interfaces) que são independentes de qualquer domínio de aplicação e que abrangem funções básicas as quais poderão ser usadas para a implementação de um objeto. Na verdade, os objetos de serviços podem ser vistos como um complemento das funções exercidas pelo ORB (*Object Request Broker*). O ORB é um conjunto de módulos de softwares que gerenciam a comunicação entre objetos e será visto com mais detalhes posteriormente.

Uma das grandes vantagens dos objetos de serviços, é que eles permitem que os desenvolvedores não precisem se preocupar com os serviços a nível de sistemas, e sim apenas com seus objetos.

3.2.2 INTERFACE DE APLICAÇÃO

A interface de aplicação é desenvolvida especificamente para uma determinada aplicação. O que juntamente ao fato da OMG não produzir aplicações, faz com que não haja uma padronização para essas interfaces. A interface de aplicação é utilizada nos objetos de aplicação para tornar as aplicações compatíveis com o padrão CORBA.

Conforme [KRU1999], “os objetos de aplicação são componentes desenvolvidos especificamente para aplicações do usuário final. Esses objetos correspondem as noções tradicionais das aplicações. Dentro deles, existem os objetos de negócio, que correspondem aos objetos que formam as aplicações compatíveis com o CORBA. A partir de um conjunto de objetos de negócio se é possível construir uma aplicação. Outro ponto importante, é que ele (objeto de negócio) manipula processos de negócios que estão presentes no mundo real. Devido a todas essas características, os objetos de aplicação não são padronizados pela OMG”.

Visando o gerenciamento da complexidade da aplicação, os objetos de negócio se concentram mais nas interfaces e suas implementações ([KRU1999]).

3.2.3 INTERFACES DE DOMÍNIO

As interfaces de domínio executam funções parecidas com as dos objetos de serviços e as facilidades comuns, entretanto, estas interfaces estão orientadas para o desenvolvimento de uma aplicação específica. Neste caso o desenvolvedor deve levar em consideração as características do sistema que esta sendo desenvolvido.

3.2.4 FACILIDADES COMUNS

Assim como os objetos de serviços, as facilidades comuns também apresentam-se como uma coleção de serviços (interfaces), os quais podem ser compartilhados com o usuário final, diferentemente dos objetos de serviços, que possuem seus serviços direcionados ao sistema ORB.

Outra diferença quanto aos objetos de serviços, é que os serviços oferecidos por essa interface (facilidades comuns) não são considerados fundamentais e sim de auxílio.

3.3 IDL E INTERFACES

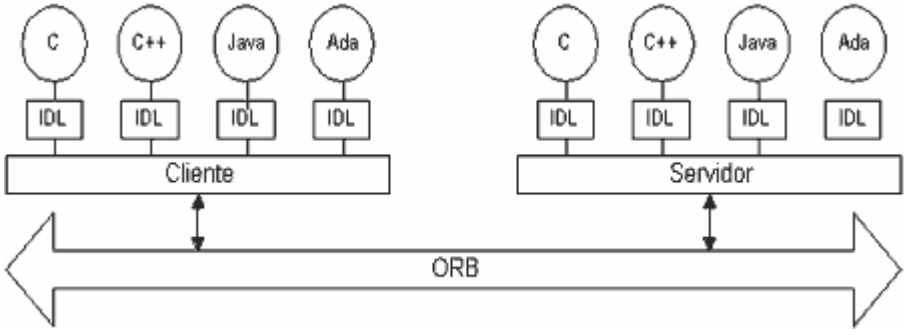
O CORBA é uma arquitetura que permite, através de um barramento comum (ORB), a comunicação entre um conjunto de objetos de diferentes sistemas, linguagens, plataformas e máquinas. Mesmo existindo essas diferenças nestes objetos, que podem estar relacionadas a linguagem utilizada para gerá-los, ou a tecnologia de rede utilizada, ou a plataforma onde ele é executado, estas diferenças não afetarão a interação desses objetos, graças às interfaces.

O Objeto não divulga detalhes de sua implementação para os outros objetos. O que ele faz na verdade é apenas fornecer aos demais os serviços que esse determinado objeto dispõe, como fazer para chamar esses serviços, e o que devem esperar como respostas. Todas essas informações constituem as interfaces, ou seja, interface é um conjunto de informações dos objetos que são disponibilizadas aos demais, utilizando para isso uma mesma linguagem, com a finalidade de permitir essa interação entre eles, sem que seja necessário se preocupar com detalhes individuais da implementação de cada objeto ([KRU1999]).

Entretanto, para que toda essa interação entre os objetos seja possível, essas interfaces devem utilizar uma linguagem para sua divulgação que seja puramente declarativa, gerando assim apenas declarações e não códigos, já que as interfaces estão desvinculadas a qualquer informação referente a implementação do objeto.

A figura 16 demonstra as diferentes linguagens utilizadas na construção dos objetos CORBA, e como as interfaces, sejam elas dos clientes ou do servidor, se comunicam utilizando um barramento comum, o ORB.

Figura 16 – Interfaces CORBA



Fonte: [ORF1996, p.50]

A IDL é a linguagem adotada pela OMG para especificar toda a sintaxe de todas as interfaces. Um exemplo de linguagem IDL (gerada pelo Delphi) pode ser vista no quadro 1.

Neste exemplo é mostrada a declaração da interface do objeto com os serviços acessíveis aos outros objetos. Os outros objetos não precisam conhecer como estes serviços foram implementados internamente, más somente sua estrutura de acesso.

Quadro 1 - Exemplo de linguagem IDL

```
module Prototipo
{
  interface IVirtualServer;
  interface IVirtualServer
  {
// Aqui são declarados todos os serviços disponíveis na interface do objeto e acessível
// aos outros objetos que interagem com ele.
    long InstanciaEquipe(in long NumeroEquipe);
    void EnviaNomes(in long NumeroEquipe, in string NomeGeral,
    in string NomeFinanceiro, in string NomeProd, in string NomeMercado);
    void EnviarQuantidades(in long NumeroEquipe, in long Periodo,
    in double ValorProd, in double ValorProdutividade, in double ValorMercado,
    in double ValorFinanceiro, in double ValorRentabilidade);
    void LiberaEquipe(in long NumeroEquipe);
    long RetornaConfig(in long NumeroEquipe, inout long Cenario,
    inout long TipoPlanilha, inout long Ano, inout long Periodo, inout long Situa);
  };
```

Todas as linguagens utilizadas no CORBA (Ada, Smaltalk, Java, C++, C e COBOL) devem ser capazes de expressar todas as construções utilizadas pela IDL.

3.4 REPOSITÓRIO DE INTERFACES

O Repositório de Interfaces, ou IR, é um componente utilizado pela IDL para se armazenar todas as definições dos seus objetos. Esse componente deve estar sempre disponível tanto ao cliente como também ao ORB e as implementações, para poder adicionar,

excluir ou mesmo executar a depuração dessas tais interfaces, além de também poder ter suas operações invocadas como qualquer outro objeto.

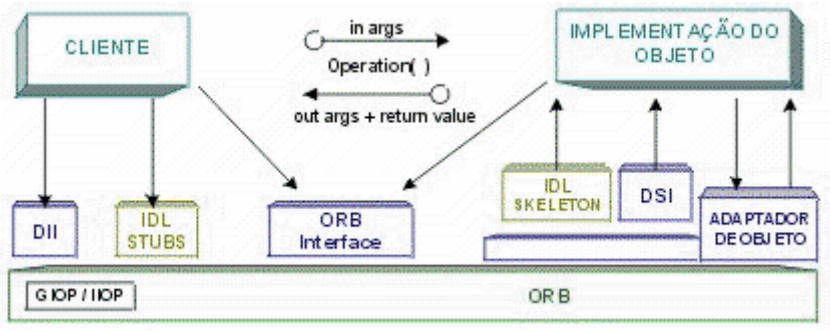
Um grande ponto referente ao IR, é que ele vem com o propósito de oferecer um auxílio na tentativa de solucionar o problema da inflexibilidade de mudança em tempo de execução, permitindo com isso, que o sistema de tipos seja acessado e escrito programavelmente em tempo de execução ([KRU1999]).

Pelo fato do IR permitir a descoberta das informações de tipo em tempo de execução, faz com que seja um excelente suporte para as invocações dinâmicas, além de ser uma grande fonte para a criação de código de suporte estático para as aplicações.

3.5 COMPONENTES DA ARQUITETURA CORBA

A arquitetura CORBA é um conjunto complexo de partes que possuem suas particularidades e especificações, todas elas reunidas para alcançarem um objetivo comum: promover um ambiente que atenda as especificações definidas pela OMG para o CORBA. Na figura 17 é mostrada a descrição das partes componentes do CORBA.

Figura 17 – Componentes CORBA



Fonte: [Vas1998, p.4]

3.5.1 ORB

O ORB (*Object Request Broker*) é fundamental para o CORBA pois exerce o papel de *middleware* entre os componentes. O ORB é responsável por toda comunicação e interação entre os objetos. Ele intercepta a chamada de um objeto para outro e fica responsável em encontrar um objeto que atenda as necessidades do pedido. Encontrando o objeto chamado, o

ORB passa os parâmetros para o mesmo, invoca os métodos necessários dele, e retorna para o objeto que solicitou os resultados de todo esse procedimento. Dessa maneira, o usuário não precisa se preocupar onde tal objeto está localizado, em que sistema operacional ele roda ou qual programa foi usado para desenvolvê-lo.

Como a comunicação é feita de forma transparente, entre o cliente e a implementação do objeto desejado, o ORB é a principal arma na simplificação da programação em ambientes distribuídos, já que o mesmo libera os programadores de realizarem o árduo trabalho de programar a baixo nível os detalhes das invocações dos métodos. O ORB tem como principal função, achar a implementação do objeto que atenda as necessidades da invocação de um determinado cliente. Após isso, fica responsável pela ativação (caso não esteja ativado) do mesmo, passar as informações necessárias para o objeto e devolver (quando for necessário) para o cliente o resultado desta operação. Todos esses passos são feitos de forma transparente para o cliente.

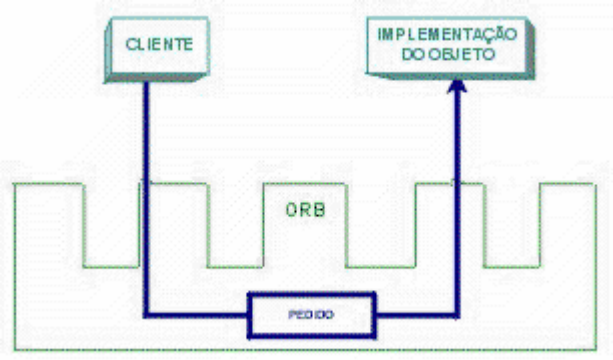
Assim, o ORB é visto como o *middleware* da especificação CORBA, pois possui um conjunto de módulos de softwares que gerenciam a comunicação entre objetos. Muitos autores, pelo fato do ORB gerenciar essa comunicação, denominam-no como sendo o barramento de objetos. O grande potencial desse barramento está em permitir que objetos façam, de forma transparente, requisições a objetos que podem estar localizados local ou remotamente. Essa transparência assegura que o cliente (requisitante) não tenha conhecimento de quais os mecanismos utilizados para se comunicar com o objeto desejado ([KRU1999]).

O ORB é composto desde o *Stub* do cliente, onde é invocado o serviço, até o *Skeleton* no servidor, ficando responsável por toda comunicação necessária entre esses dois componentes que serão detalhados a seguir.

O ORB permite a utilização de linguagens consideradas de alto nível na invocação de métodos no servidor, não importando em qual linguagem o servidor de objetos esteja implementado. Tudo isso é possível, porque o CORBA separa a interface da implementação, além de fornecer dados neutros para que seja possível a chamada de objetos através de diferentes linguagens.

A figura 18 mostra simplificadamente a comunicação do cliente o objeto através do ORB.

Figura 18 – Estrutura do ORB



Fonte: [MIR1998, p.10]

3.5.2 CLIENTE

Segundo [KRU1999], “cliente é toda entidade (programa) responsável em invocar uma determinada operação sobre uma implementação de objeto”. Como uma das características do CORBA, os detalhes utilizados (como o adaptador de objeto, ou o ORB utilizado) para que seja feito esse acesso aos serviços de um determinado objeto é transparente e dispensável do conhecimento do cliente, precisando o mesmo apenas saber interagir com a interface do tal objeto. Essa invocação feita pelo cliente, que para o mesmo aparenta ser localmente, poderá ser feita utilizando-se o *Stub* ou utilizando-se de um conjunto de rotinas de invocações feitas dinamicamente através da interface DII (Interface de Invocação Dinâmica) que será explicada a seguir.

O *Stub* tem como característica promover interfaces estáticas para criar e enviar requisições correspondentes dos serviços desejados do cliente para o servidor. O *Stub* é criado utilizando-se um compilador IDL e está localizado no cliente.

Conhecidos também como *proxies*, por possuírem a capacidade de localizar os objetos alvos, os *Stubs* são gerados a partir da compilação da interface do cliente definida anteriormente em IDL. Quando um cliente deseja chamar um método de um objeto, basta apenas indicar qual o objeto que deseja. Tal chamada é repassada para o *Stub* que terá como função receber essa chamada, localizar o objeto desejado, transformar a chamada para que

possa ser enviada pela rede e transmiti-la para o ORB para que se possa ser dado continuidade no resto do processo. A comunicação entre o *Stub* e o ORB se dá por meio de interfaces privadas.

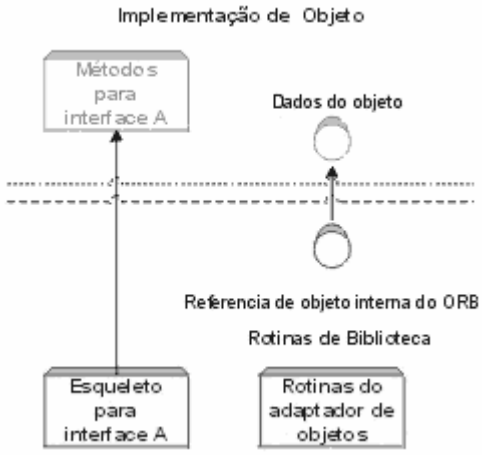
A Interface de Invocação Dinâmica (DII) permite ao cliente, por meio do descobrimento dos métodos em tempo de execução, um acesso direto aos mecanismos de requisição residentes em um ORB e também que realizem uma sincronização deferida não-bloqueada (operações separadas de envio e recebimento) e chamadas unidirecionais (só de envio). Neste tipo de invocação, esse processo é feito de forma dinâmica sem necessitar passar pela interface específica da IDL para que seja ativado (*Stub*).

Quando um cliente faz uma requisição a um determinado objeto, ele (cliente) não precisa saber se esse tal objeto está ativo ou não, pois o ORB se encarrega, antes de fazei-lo. Enfim, toda a função que o ORB se resume a fazer, está vinculada em requisições de serviços que são feitas pelo cliente ao servidor de objetos.

3.5.3 IMPLEMENTAÇÃO DE OBJETOS

As implementações de objetos são as operações que podem ser feitas em uma interface ORB IDL. Este componente abriga informações de objetos, como dados internos, os códigos de seus métodos e seus respectivos procedimentos de ativação e desativação determinados na criação do objeto. A figura 19 representa uma implementação de objeto.

Figura 19 – Implementação do Objeto



Fonte: [CAR1997, p.74]

O *Skeleton* está vinculado diretamente com o *Stub*, e tem como função fornecer a interface através do qual o método recebe as requisições.

Os *Skeleton*s executam função similar ao *Stub* do cliente, porém encontram-se situado no servidor. Os *Skeleton*s disponibilizam os métodos dos objetos servidores para o resto do sistema. Tem como função encontrar os serviços que são solicitados pelo *Stub*. Como o *Stub*, os *Skeleton* são criado a partir da compilação de interfaces previamente escritas em IDL. O *Skeleton* irá receber o pacote do ORB, após isso, irá descompactá-lo e enviá-lo para a implementação do objeto. Nesse momento a requisição será recebida pelo objeto que irá executá-la, e havendo alguma resposta, tal resposta irá realizar o caminho contrário ao que a requisição realizou: objeto para o *Skeleton*, *Skeleton* para ORB, do ORB para o *Stub*, e por fim, do *Stub* para o cliente.

Pelo fato de existir um *Skeleton*, não implicar em dizer que existe um *Stub* do outro lado, pois a invocação a ele pode ser feita de forma dinâmica, sem utilizar o *Stub*.

A Interface de Esqueleto Dinâmico (DSI) realiza uma função semelhante a realizada pela DII. A diferença básica, é que todas as atividades realizadas pela DSI são feitas no servidor. A principal característica da DSI é permitir que o ORB transmita requisições para uma implementação de objeto, apesar de não conhecer em tempo de execução o tipo de objeto utilizado.

3.5.4 ADAPTADOR DE OBJETO

Chamado de AO, o adaptador de objeto tem como principal função auxiliar o ORB no despacho de requisições para os objetos e na ativação de um determinado objeto.

Segundo ([KRU1999]), “o AO é considerado o núcleo da comunicação do ORB, pois associa implementações de um objeto com o ORB”. Nos ORB’s tem se utilizado a denominação de BOA *(Basic Object Adapter)*.

O Adaptador de Objetos, é um componente da especificação CORBA que exerce o papel de intermediário na comunicação entre os objetos e o ORB, ajudando o ORB a entregar as requisições a implementação dos objetos bem como também ajuda na ativação dos objetos.

Esse componente é necessário devido a grande diversidade existente de implementações de objetos. O Adaptador de Objetos tem a oferecer os seguintes serviços:

a) registros de implementação de objetos: utilizado na localização da implementação do objeto;
b) geração de referência a objeto: tem a capacidade de gerar referência a objetos;
c) capacidade de invocações (em auxílio com o *Skeleton*);
d) ativar e desativar uma implementação de objeto: podem ativar ou desativar um determinado objeto quando for necessário;

segurança das interações: em conjunto com o ORB, deve garantir a entregar das requisições por meio de conexões múltiplas sem bloquear alguma dessas conexões de maneira indefinida.

3.5.5 GIOP

Quando a comunicação entre os ORB´s é necessária, precisa existir uma forma de permitir esta comunicação, uma ponte com um protocolo de comunicação que todos os ORB's possam entender.

O mecanismo de ponte foi criado, entretanto para que o mesmo se tornasse operável. Assim a OMG especificou o chamado GIOP (*General Inter-ORB Protocol*), que possui como característica ser um padrão que atende as premissas (especificando formato das mensagens e uma representação comum para os dados) que permite a interligação ORB-para-ORB além de ser capaz de trabalhar sobre qualquer protocolo de transporte, que lógico, tenha este o mínimo de pré-requisitos necessários que o caracterize como um protocolo de transporte ([KRU1999]).

Além de especificar um padrão para sintaxe de transferência, a OMG precisava elaborar alguma especificação que tivesse relacionado ao transporte dessa sintaxe. Com isso a organização adotou o TCP/IP como protocolo de transporte padrão, com isso surgiu o IIOP (*Internet Inter-ORB Protocol*), que especifica como as mensagens de padrão GIOP são transmitidas por meio de uma rede TCP/IP. O IIOP possibilita o uso da própria internet como *backbone* entre os ORB's. O IIOP provê interoperabilidade também entre ORB's compatíveis ao padrão CORBA. A especificação de interoperabilidade da OMG definiu também o ESIOP (*Environment-Specific Inter-ORB Protocol*), como um conjunto de protocolos que permitem a

interoperabilidade entre implementações baseados em GIOP, mas que utilizam protocolos proprietários de transportes.

3.6 SEGURANÇA

A Arquitetura CORBA implementa alguns serviços que visam a segurança na transmissão das operações realizadas. Como exemplo destacam-se:

a) autenticação: trata-se de um identificador único, que permite ao cliente acessar qualquer servidor em qualquer lugar. Essa autorização se procede a partir de um simples *log on*, que pega a autenticação e lhe oferece um conjunto de opções de segurança para a comunicação. Para uma maior segurança, nenhuma senha é armazenado no *login script* do cliente. Após feita a autenticação de um cliente em um determinado ORB da rede, o mesmo (ORB) se encarregada de propagar a autenticação por todo o resto da rede.

b) autorização: após feita a autenticação do cliente, os servidores de objetos ficam encarregados em verificar quais operações podem ser feitas por aquele determinado cliente. Para tal, os servidores usam as ACLs que contém uma lista de nomes e suas respectivas operações que podem ser executadas pelos mesmos. É possível implementar várias políticas de ALC, podendo serem implementadas tanto pelo ORB como pelos servidores de objetos.

c) a*udit*: serviço que monitora todos os eventos que ocorrem no ORB, incluindo *logons* de clientes, quais objetos estão sendo usados e com quais servidores. O Audit é considerado como parte fundamental na segurança, pois tem a capacidade de detectar possíveis intrusos.

d) criptografia: o CORBA também provê os serviços de criptografia de suas operação.

e) *non-repudiation*: dispõe de "guardas eletrônicos" que protegem todas as partes no sistemas de possíveis falsos pedidos de requisições.

3.7 CONTROLE DE VERSÃO

Para alterar um objeto, adicionando novos métodos, é possível criar este objeto com o mesmo nome do já antigo, passando então e existir dois objetos, de versões diferentes, como o mesmo nome, porém com identificadores únicos. No caso de uma chamada a um método (objeto), através de pesquisa por nome, se este objeto tiver mais de uma versão, a busca

resultará em quantas versões desse objeto existirem. O sistema se encarrega de converter todos os objetos encontrados, em referências que possam ser acessadas por todos, caso aconteça a impossibilidade de utilizar-se algumas.

No próximo capítulo será demonstrada a especificação e implementação do sistema. Serão apresentados mais detalhadamente aspectos do uso do CORBA utilizando o ambiente de desenvolvimento Delphi.

4 DESENVOLVIMENTO

Neste capítulo será apresentada a especificação e implementação do modelo proposto neste trabalho.

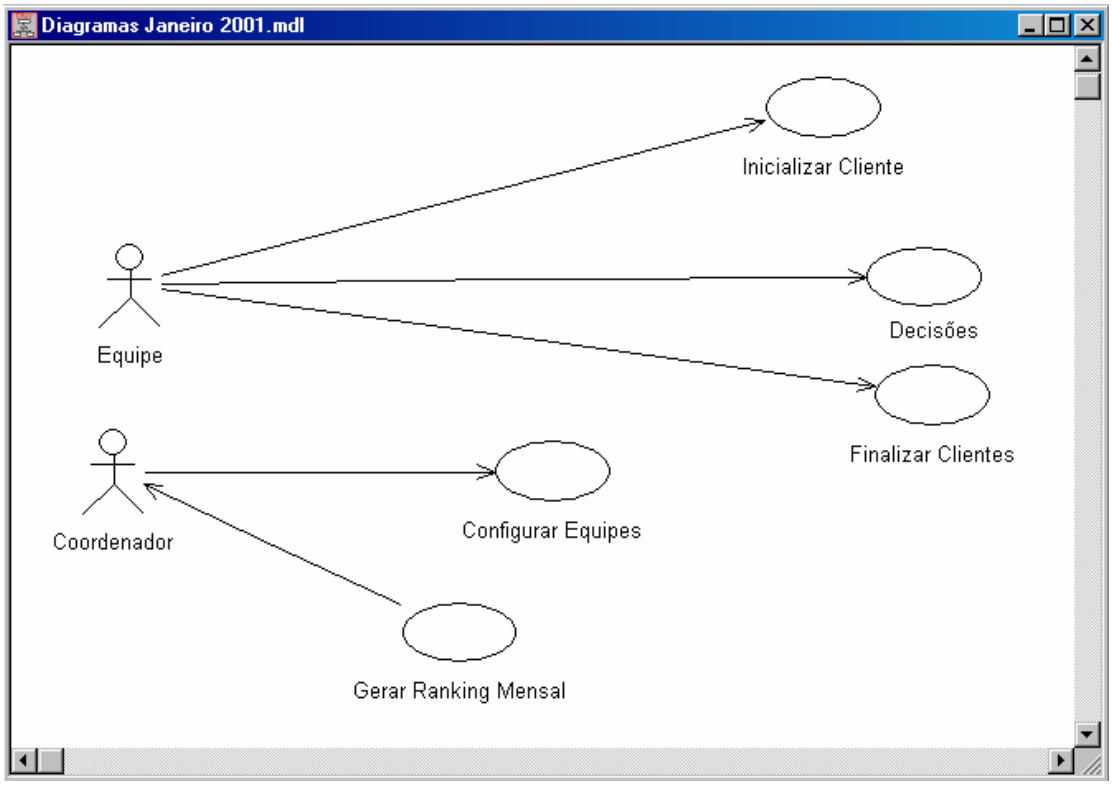
4.1 ESPECIFICAÇÃO DO PROTÓTIPO

A especificação do sistema foi feita utilizando-se algumas ferramentas da linguagem de modelagem UML - *Unified Modeling Language*. A ferramenta CASE utilizada para esta especificação foi o *Rational Rose* da *Rational®*, devido aos recursos disponíveis para aplicar as representações da UML, como o Diagrama de Classes, o Diagrama de Casos de Uso e o Diagrama de Seqüência que abaixo estão relacionados.

4.1.1 DIAGRAMA DE CASOS DE USO

A figura 20 apresenta os principais diagramas de casos de uso do sistema.

Figura 20 - Diagramas de caso de uso



Foram identificados os seguintes casos de uso do sistema:

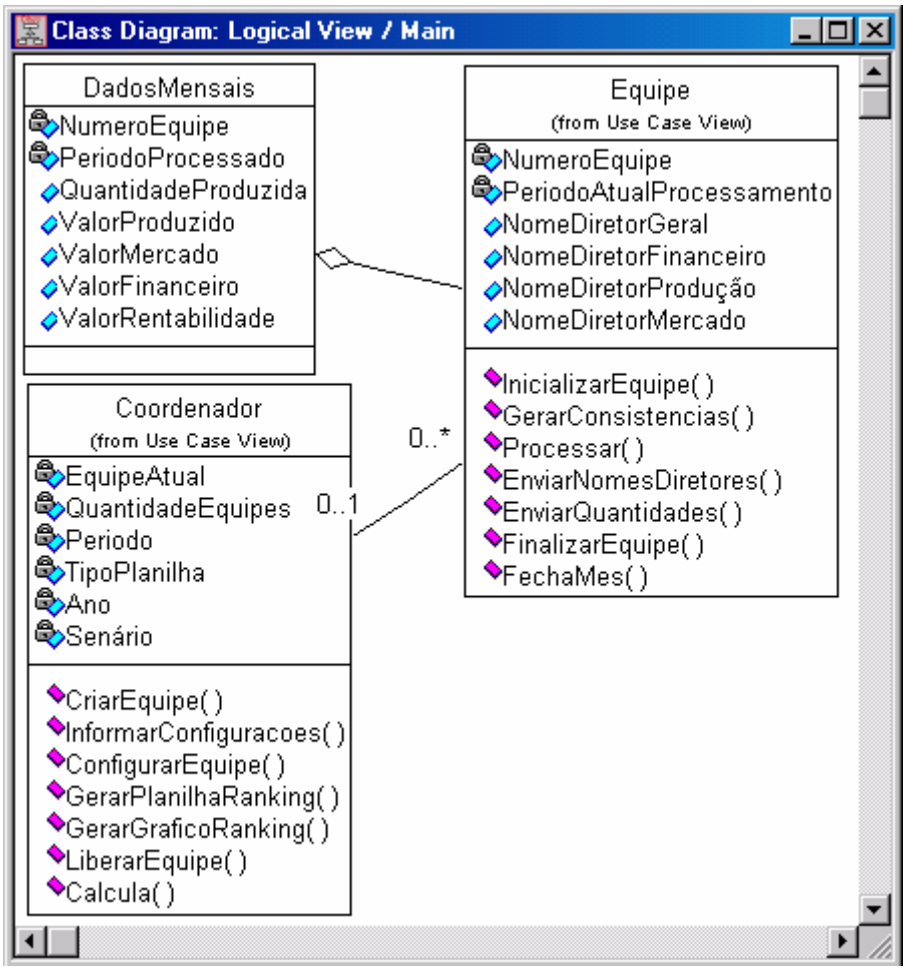
a) Inicializar Cliente: neste caso de uso a equipe inicializa o seu sistema, que é o mesmo apresentado no capítulo 2 deste trabalho. Esta inicialização deve ser comunicada ao servidor que irá registrá-lo;
b) Decisões: este caso de uso acontece quando a equipe decide fechar as decisões do mês atual; Os valores são processados e enviados para o servidor que os torna acessíveis ao coordenador;
c) Finalizar Cliente: neste caso de uso a equipe finaliza o sistema. Esta operação é, então, comunicada ao servidor;
d) Configurar Equipes: neste caso de uso o coordenador configura as características do jogo. Estas configurações são repassadas aos clientes;
e) Gerar Ranking Mensal: quando todas as equipes já processaram e enviaram seu valores, o servidor irá processar e calcular o ranking mensal das equipes. Será criada uma planilha para demonstração deste ranking e gráficos para melhor visualização dos dados.

4.1.2 DIAGRAMA DE CLASSES

As classes identificadas para este sistema, especificada no diagrama de classes abaixo, são equipe (cliente), dados mensais e a classe coordenador (servidor) demonstradas na figura 21.

A classe cliente possui uma estrutura de dados relacionados a ela que guarda todos os valores dos meses processados da equipe. Possui também o número da equipe, período processado e os nomes dos integrantes da equipe (diretores). A classe coordenador possui os dados para controle interno das equipes como: equipe atual, quantidade de equipes, período, tipo de planilha, ano atual e o cenário (A, B ou C) da simulação. Os métodos das classes serão explicados na demonstração do protótipo.

Figura 21 - Diagrama de Classes

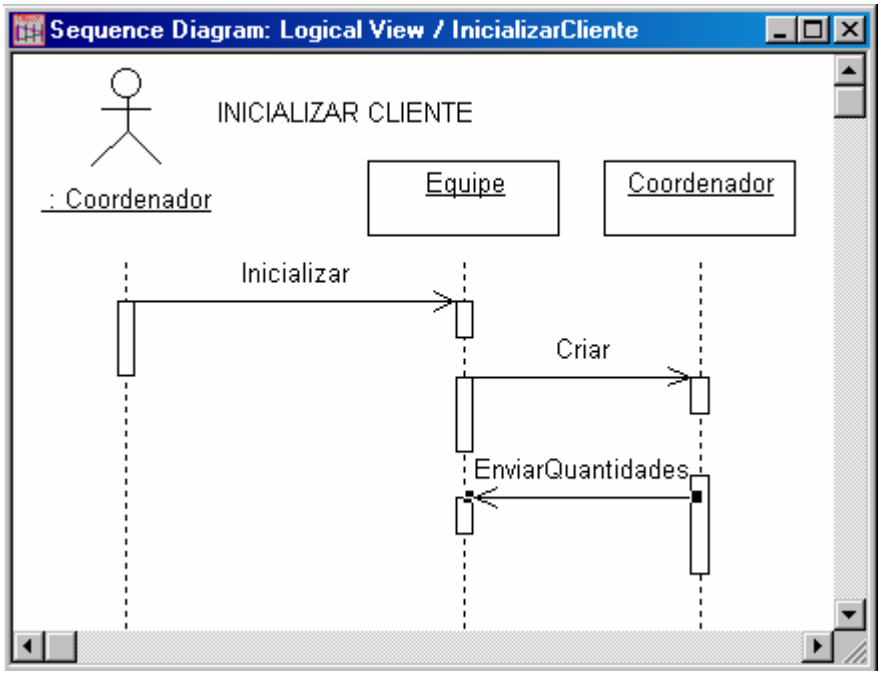


4.1.3 DIAGRAMA DE SEQUÊNCIA

Os diagramas de seqüências representam, como o próprio nome indica, a seqüência em que as ações ocorrem dentro do sistema. Eles demonstram como é feita a troca de mensagens entre as classes (objetos). Para cada caso de uso, há um diagrama de seqüência. Logo, haverá um diagrama de seqüência para todos os casos de uso demonstrados anteriormente.

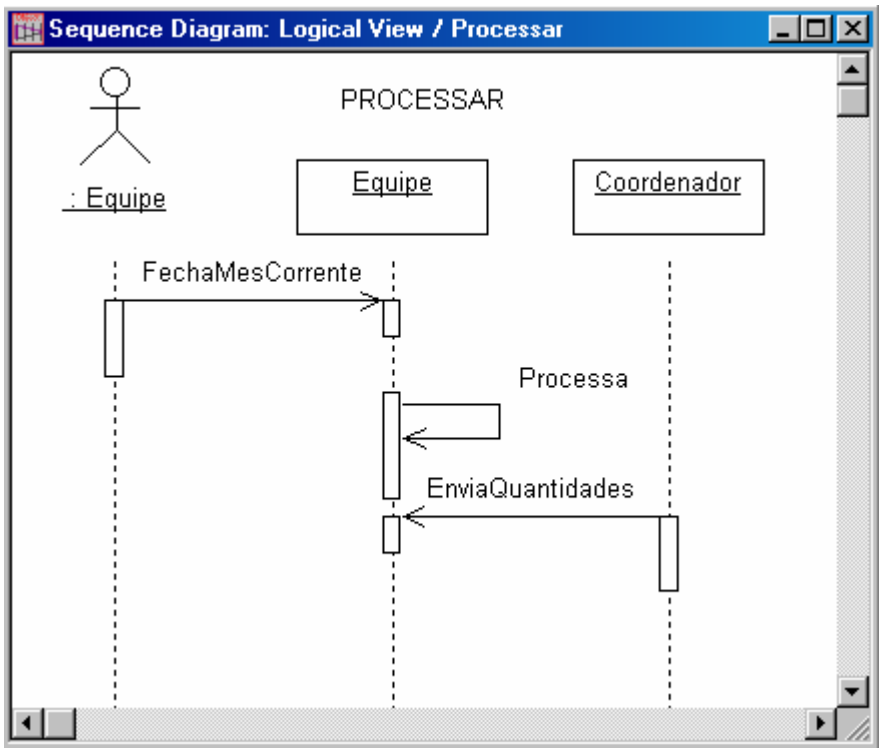
a) Inicializar Cliente: a equipe abre o sistema para ser executado. O cliente envia o número da equipe ao coordenador (Inicializar), que verifica se a equipe já foi criada, e, se não foi, a cria (CriaEquipe). O cliente envia as quantidades para cada mês já processado (EnviaQuantidades) e passa a estar pronto para ser utilizado pela equipe.

Figura 22 – Seqüência Inicializar Cliente



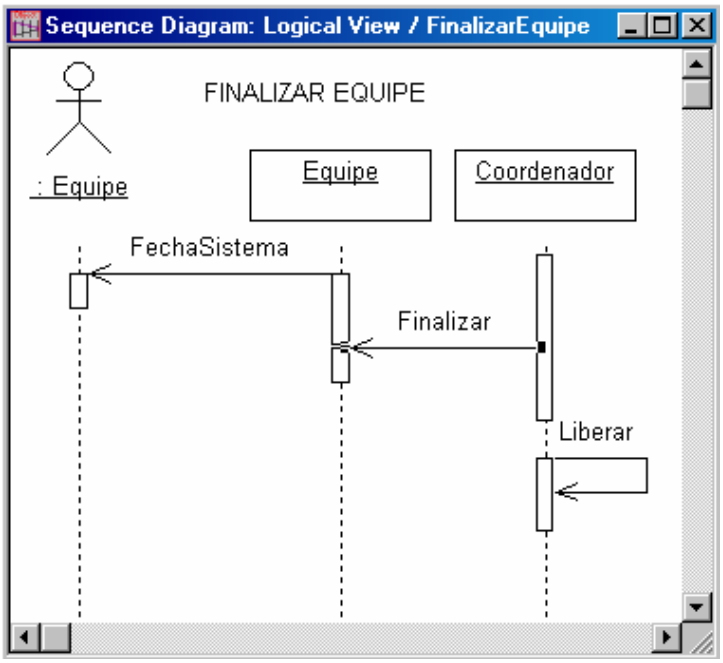
b) Processar: a equipe fecha as decisões do mês atual (FechaMesCorrente), consiste os valores (GerarConsitencias) e processa os mesmos (Processar). Os nomes dos diretores do mês atual são enviados para o coordenador (EnviaNomesDiretores) e as quantidades processadas também (EnviaQuantidades).

Figura 23 – Seqüência Processar



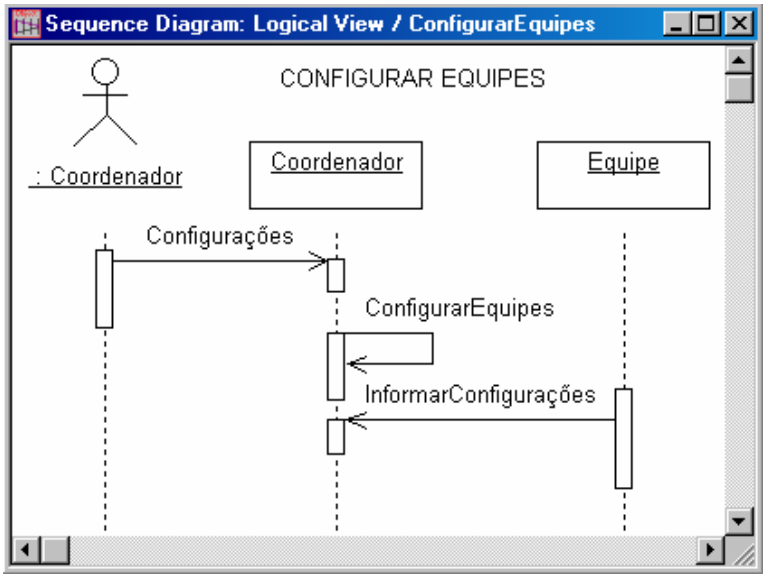
c) Finalizar Equipe: a equipe fecha o sistema (FechaSistema). Antes de finalizar totalmente o sistema uma mensagem é enviada ao servidor (FinalizarEquipe) que irá liberar o cliente (LiberarEquipe).

Figura 24 – Seqüência Finalizar Equipe



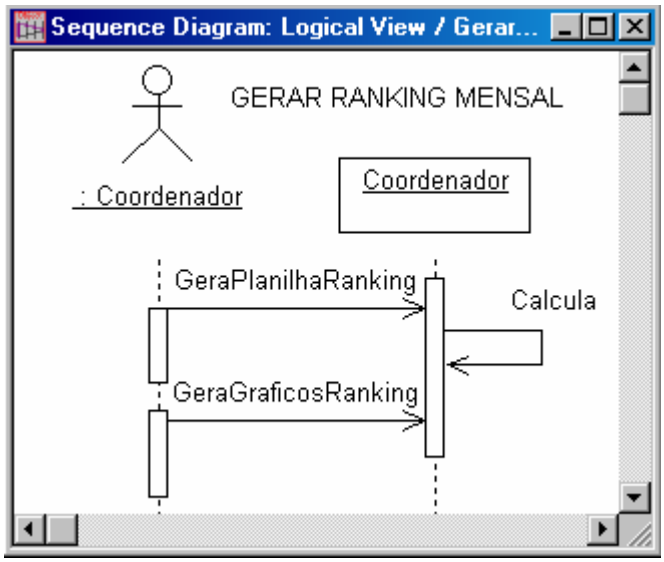
d) Configurar Equipes: o coordenador altera as configurações do jogo (Configuracoes) e o servidor configura suas equipes (ConfiguraEquipes). Após isto, o servidor envia as configurações para os clientes (InformarConfiguração).

Figura 25 – Seqüência Configurar Equipes



e) Gerar Ranking Mensal: o coordenador irá processar e calcular o ranking mensal das equipes (CalculaRanking). Será criada uma planilha para demonstração deste ranking (GerarPlanilhaRanking) e gráficos para melhor visualização dos dados (GerarGráficosRanking).

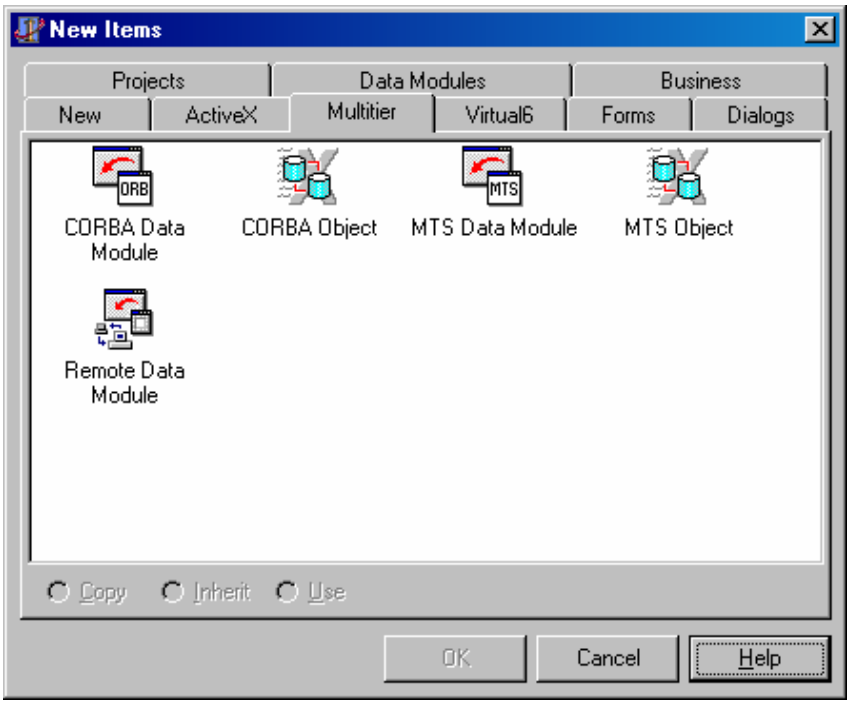
Figura 26 – Seqüência Gerar Ranking Mensal



4.1.4 IMPLEMENTAÇÃO

Segundo [CAN2000] “o Delphi 5 incorporou um bom suporte a tecnologia CORBA. Entretanto, este suporte a CORBA ainda não tornou-se nativo. O principal fator limitante do suporte a CORBA é que ele permite apenas chamadas de método através da execução de vínculo tardio do CORBA, ou seja, o descobrimento dos métodos do objeto em tempo de execução (DII), método este que não é o mais rápido”. Embora este fator não seja relevante para a construção do protótipo do VIRTUAL, é importante salientar que a Inprise (empresa fabricante do ambiente Delphi 5) já está desenvolvendo o conversor de IDL para Pascal (linguagem utilizada pelo ambiente Delphi), o que possibilitará utilizar todos os recursos do CORBA, tornando-o nativo do ambiente Delphi. Este foi um dos fatores que foi levado em consideração para a escolha de CORBA para a implementação deste protótipo (o suporte total a tecnologia CORBA). Outro fator importante para a escolha do padrão CORBA é que o ambiente Delphi possui também a estrutura auxiliar de construção de comunicação baseada somente em objetos (*Corba Object* – construção do objeto puro), que pode ser acessada através da opção FILE/NEW na pasta *Multitier* conforme mostra a figura 27.

Figura 27 – Acesso aos assistentes de construção CORBA do Delphi 5



Os itens da pasta são:

a) *Corba Data Module*: é a estrutura utilizada para trabalhar com banco de dados distribuídos propriamente dita;

b) *Corba Object*: é a estrutura utilizada para trabalhar com objetos distribuídos CORBA puros que é preciso para a implementação do protótipo do VIRTUAL, já que o mesmo só irá utilizar a comunicação para a passagem de pacotes de dados fixos e não dados variáveis como acontecem em bancos de dados, ou seja, a quantidade de informações passadas do cliente para o servidor é sempre a mesma, com o mesmo tipo de estrutura de dados e layout definido;

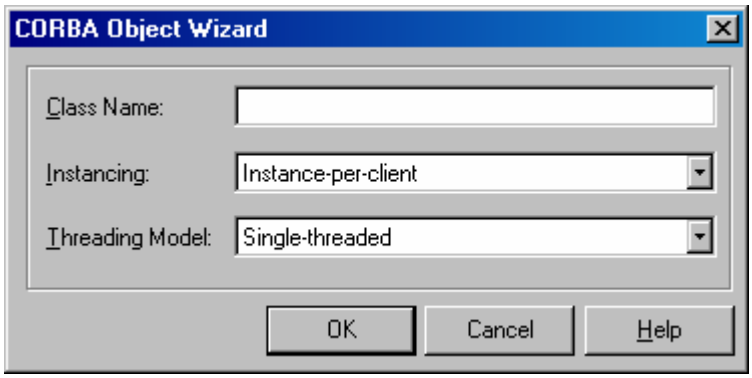
c) *MTS Object*: objetos para servidores de transação MTS (*Microsoft Transaction Server*) puros;

d) *MTS Data Module*: banco de dados para MTS;

e) *Remote Data Module*: banco de dados para acesso remoto.

Após ser escolhida a opção *Corba Object* foi aberta a escolha das configurações da classe CORBA a ser criada como mostra a figura 28.

Figura 28 – Criação da classe CORBA principal



Na figura 28 basta apenas definir o Nome da Classe (que para o protótipo será *VIRTUALServer*), o tipo de instanciação (será mantida uma instância por cliente – *Instance per Client*) e o módulo de execução das *threads* (*Single-threaded* que é o padrão).

Após configurado o nome e propriedades iniciais, o Delphi abre o editor de objetos com as definições da classe já criada como mostra a figura 29 e sua declaração inicial como mostra a figura 30.

Figura 29 – Classe CORBA e Interface inicial criadas automaticamente.

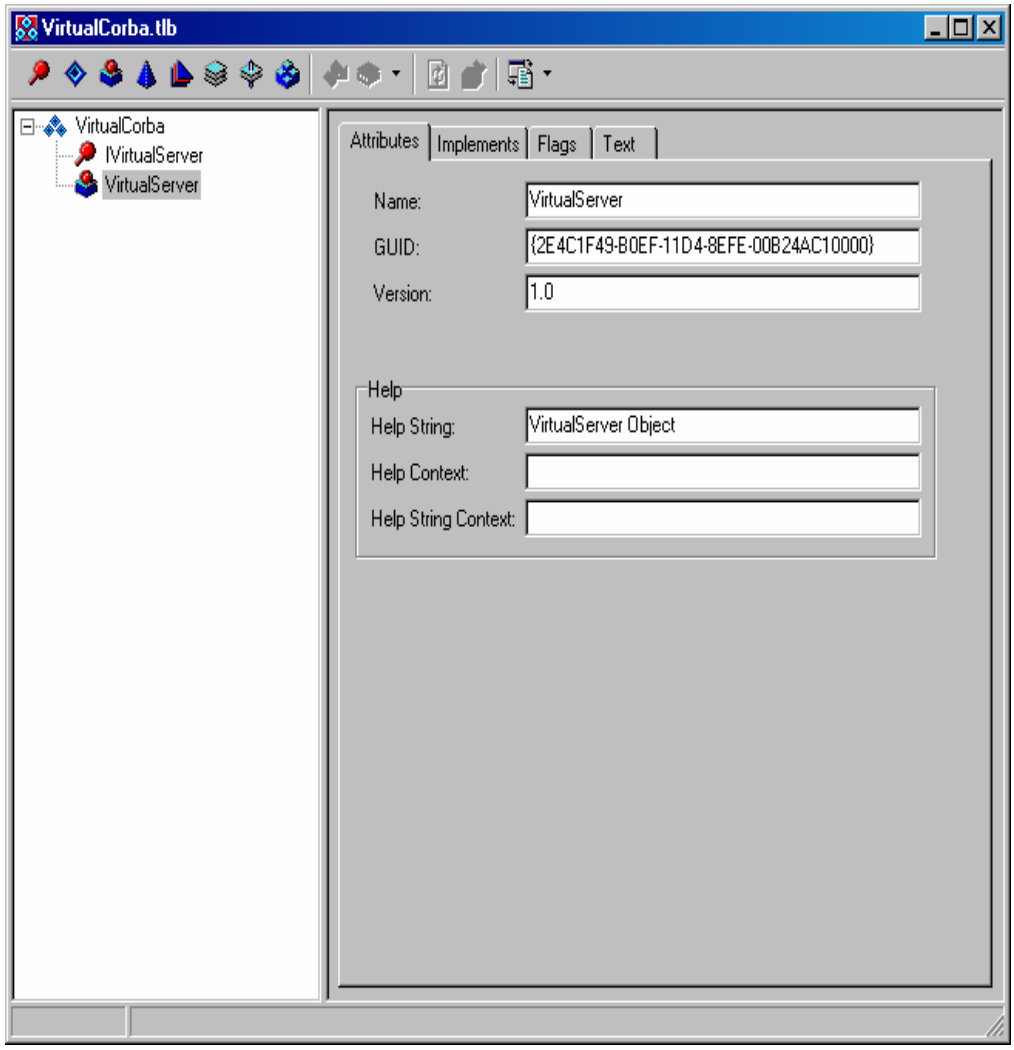
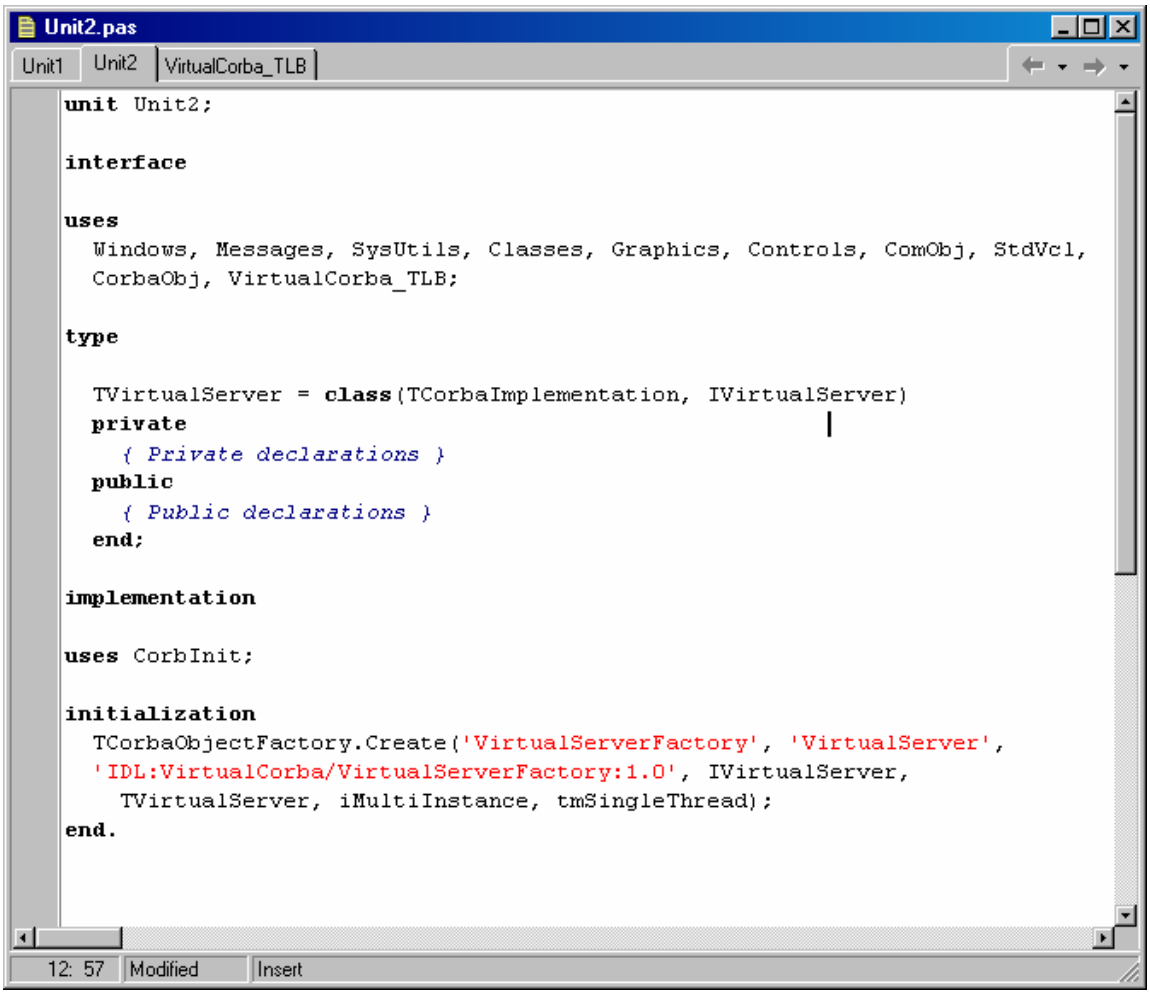


Figura 30 – Classe declarada automaticamente.



Os quadros 02 e 03 mostram toda a definição dos componentes do objeto CORBA que foram sendo criados, como a Interface, o *Stub* para a comunicação com o cliente e o *Skeleton* do servidor.

A partir disto serão implementadas as estruturas de dados que serão enviadas para a comunicação entre o cliente e o servidor e as propriedades e métodos necessários para o desenvolvimento do protótipo.

O método TCorbaObjectFactory.Create da figura 30 é o método responsável pela criação e registro da classe no sistema operacional, para que todos os outros sitemas possam localizar este objeto e se comunicar com o mesmo.

Quadro 2 – Implementação dos Componentes do Objeto

```
unit VIRTUALCorba_TLB;
const
  VIRTUALCorbaMajorVersion = 1; // VERSÃO DO OBJETO.
  VIRTUALCorbaMinorVersion = 0;
  // IDENTIFICAÇÕES DAS INTERFACES E OBJETOS.
  LIBID_VIRTUALCorba: TGUID = '{2E4C1F46-B0EF-11D4-8EFE-00B24AC10000}';
  IID_IVIRTUALServer: TGUID = '{2E4C1F47-B0EF-11D4-8EFE-00B24AC10000}';
  CLASS_VIRTUALServer: TGUID = '{2E4C1F49-B0EF-11D4-8EFE-00B24AC10000}';
type
  IVIRTUALServer = interface;
  IVIRTUALServerDisp = dispinterface;
  VIRTUALServer = IVIRTUALServer;
  // DECLARAÇÃO DA INTERFACE.
  IVIRTUALServer = interface(IDispatch)
    ['{2E4C1F47-B0EF-11D4-8EFE-00B24AC10000}']
  end;
IVIRTUALServerDisp = dispinterface
    ['{2E4C1F47-B0EF-11D4-8EFE-00B24AC10000}']
  end;
  // STUB.
  TVIRTUALServerStub = class(TCorbaDispatchStub, IVIRTUALServer)
  public
  end;
  // SKELETON.
  TVIRTUALServerSkeleton = class(TCorbaSkeleton)
  private
    FIntf: IVIRTUALServer;
  public
    constructor Create(const InstanceName: string; const Impl: IUnknown); override;
    procedure GetImplementation(out Impl: IUnknown); override; stdcall;
end;
```

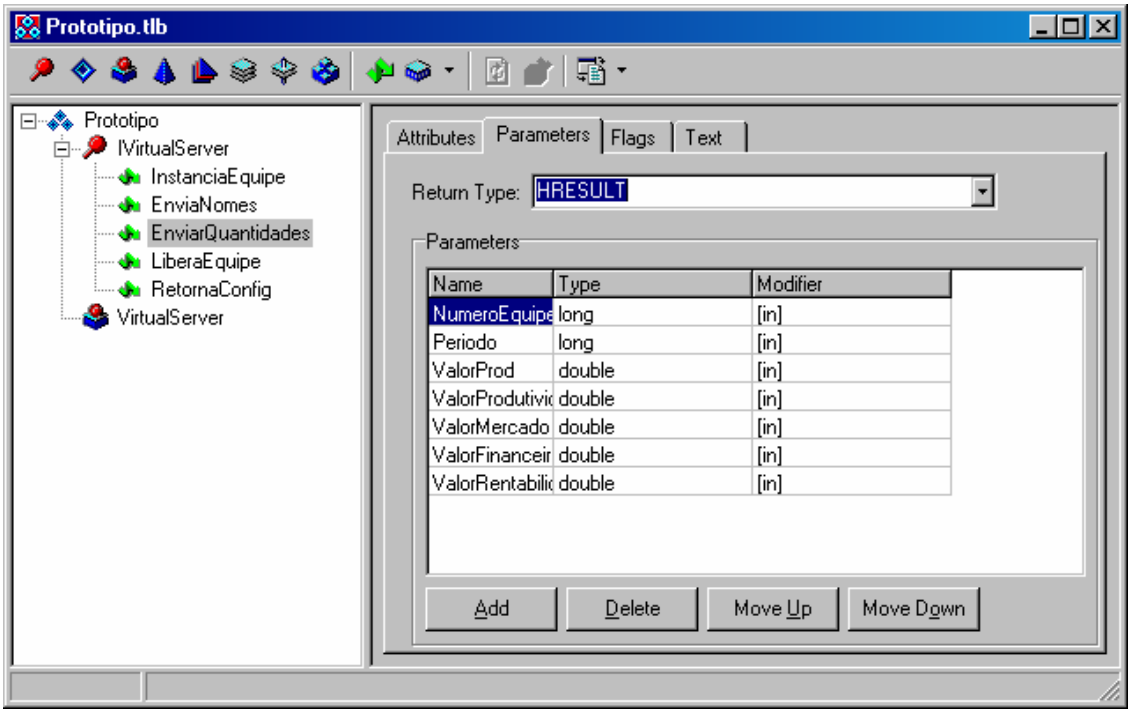
Quadro 3 - Declaração da Classe do objeto

```
// CLASSE DA INTERFACE.
  CoVIRTUALServer = class
    class function Create: IVIRTUALServer;
    class function CreateRemote(const MachineName: string): IVIRTUALServer;
  end;
{$IFDEF LIVE_SERVER_AT_DESIGN_TIME}
  TVIRTUALServerProperties= class;
{$ENDIF}
  TVIRTUALServer = class(TOleServer)
  private
    FIntf:        IVIRTUALServer;
{$IFDEF LIVE_SERVER_AT_DESIGN_TIME}
    FProps:       TVIRTUALServerProperties;
    function      GetServerProperties: TVIRTUALServerProperties;
{$ENDIF}
    function      GetDefaultInterface: IVIRTUALServer;
  protected
    procedure InitServerData; override;
  public
    constructor Create(AOwner: TComponent); override;
    destructor  Destroy; override;
    procedure Connect; override;
    procedure ConnectTo(svrIntf: IVIRTUALServer);
    procedure Disconnect; override;
    property  DefaultInterface: IVIRTUALServer read GetDefaultInterface;
  published

TVIRTUALServerCorbaFactory = class
    class function CreateInstance(const InstanceName: string): IVIRTUALServer;
  end;
```

Como a interface já foi criada, agora deve-se implementar os métodos que farão a comunicação entre os objetos. Conforme figura 31 foram implementados os seguintes métodos:

Figura 31 – Implementação dos métodos de comunicação



a) InstanciaEquipe: este método é chamado a cada vez que o cliente é aberto, a equipe manda a sua identificação para o servidor, que com base nesta identificação (número da equipe) verifica se ela já foi criada, caso contrário cria a equipe, e inicializa a mesma. A implementação é demonstrada no quadro 4;

Quadro 4 – Implementação do Método InstanciaEquipe

```
function TVirtualServerStub.InstanciaEquipe(NumeroEquipe: Integer): Integer;
var
  OutBuf: IMarshalOutBuffer;
  InBuf: IMarshalInBuffer;
begin
  FStub.CreateRequest('InstanciaEquipe', True, OutBuf);
  OutBuf.PutLong(NumeroEquipe);
  FStub.Invoke(OutBuf, InBuf);
  Result := InBuf.GetLong;
end;
```

b) EnviaNomes: envia os nome dos diretores para o cadastramento dos mesmos no mês de processamento atual da equipe. Este método é chamado sempre que o mês atual da equipe é processado. A implementação é demonstrada no quadro 5;

Quadro 5 – Implementação do Método EnviaNomes

```
procedure TVirtualServerStub.EnviaNomes(NumeroEquipe: Integer; NomeGeral:
PChar;
      NomeFinanceiro: PChar; NomeProd: PChar; NomeMercado: PChar);
var
  OutBuf: IMarshalOutBuffer;
  InBuf: IMarshalInBuffer;
begin
  FStub.CreateRequest('EnviaNomes', True, OutBuf);
  OutBuf.PutLong(NumeroEquipe);
  OutBuf.PutText(NomeGeral);
  OutBuf.PutText(NomeFinanceiro);
  OutBuf.PutText(NomeProd);
  OutBuf.PutText(NomeMercado);
  FStub.Invoke(OutBuf, InBuf);
end;
```

c) EnviaQuantidades: envia os valores processados para o cadastramento dos mesmos no mês de processamento atual da equipe. Este método é chamado sempre que o mês atual da equipe é processado. A implementação é demonstrada no quadro 6;

Quadro 6 – Implementação do Método EnviaQuantidades

```
procedure TVirtualServerStub.EnviarQuantidades(NumeroEquipe: Integer;
Periodo: Integer; ValorProd: Double; ValorProdutividade: Double;
ValorMercado: Double; ValorFinanceiro: Double; ValorRentabilidade: Double);
```

d) LiberaEquipe: este método libera as informações do cliente. É chamado após a requisição do cliente para sua liberação. A implementação é demonstrada no quadro 7;

Quadro 7 – Implementação do Método LiberaEquipe

```
procedure TVirtualServerStub.LiberaEquipe(NumeroEquipe: Integer);
var
  OutBuf: IMarshalOutBuffer;
  InBuf: IMarshalInBuffer;
begin
  FStub.CreateRequest('LiberaEquipe', True, OutBuf);
  OutBuf.PutLong(NumeroEquipe);
  FStub.Invoke(OutBuf, InBuf);
end;
```

e) RetornaConfig: método que é executado no servidor pelo coordenador e retorna as configurações informadas para o cliente, que passa a processar de acordo com estas configurções;

f) GerarRanking: após todos os clientes processarem sua informações e enviarem as mesmas para o servidor, estas informações são reunidas e processadas para gerar o ranking final do mês de processamento. Após este método, pode ser chamado o método que irá gerar o gráfico e a planilha de ranking;

g) GerarGraficoRanking: gera o gráfico de ranking comparativo dos clientes até o mês processado;

h) InicializarEquipe: é executado após a criação da equipe e inicializa os valores iniciais da equipe, bem como os valores já processados anteriormente;

i) FinalizarEquipe: Envia uma requisição para o server que irá liberar as informações da equipe.

Como os clientes necessitam de um ORB para se comunicar com o servidor, será utilizado o VisiBroker ORB da Inprise que vem junto com a versão Enterprise do Delphi 5, demonstrado na figura 32.

Figura 32 - VisiBroker ORB da Inprise

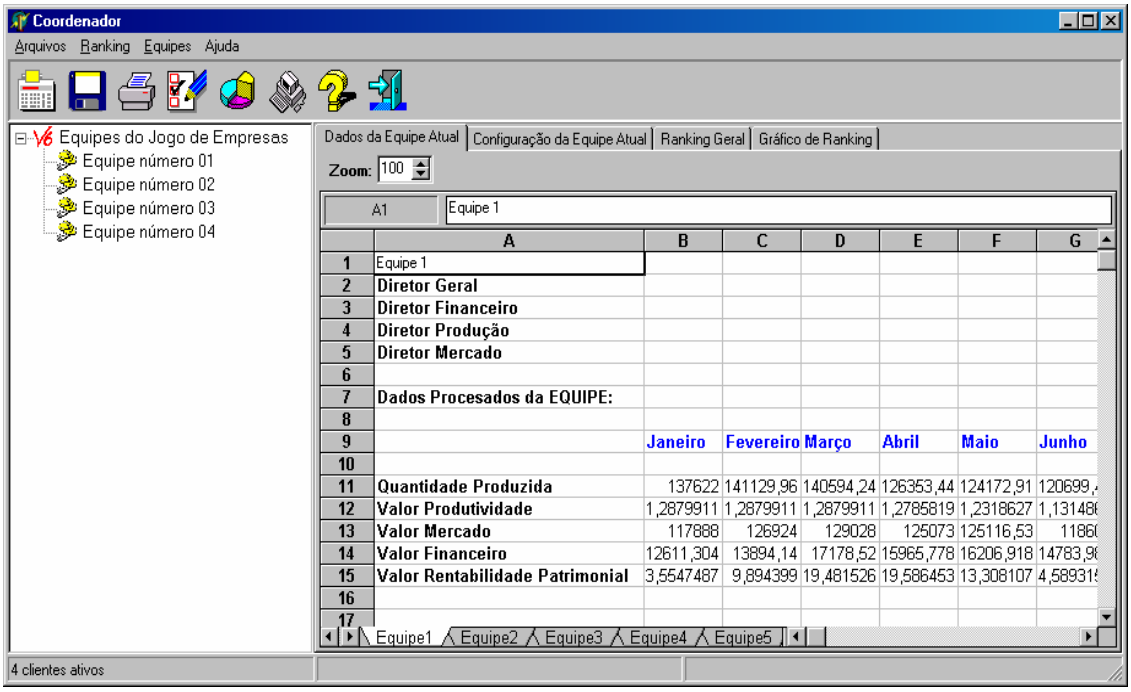
Este ORB deverá estar sempre em execução para que o servidor e os clientes consigam se comunicar entre si, pois é através do ORB que os objetos CORBA localizam e requisitam seus serviços através da rede. O VisiBroker ORB poderá estar executando em qualquer máquina da rede, na máquina do servidor ou do cliente ou em outra máquina que não contém nem o servidor, nem o cliente.

Após todos estes passos desenvolvidos, os serviços CORBA do servidor já estão criados e mesmo já pode ser implementado para atender os objetivos do simulador.

4.1.4.1 MÓDULO DO COORDENADOR

A figura 33 mostra a tela principal do módulo servidor que foi criado e suas funções serão descritas a seguir:

Figura 33 - Tela Principal do Protótipo Servidor



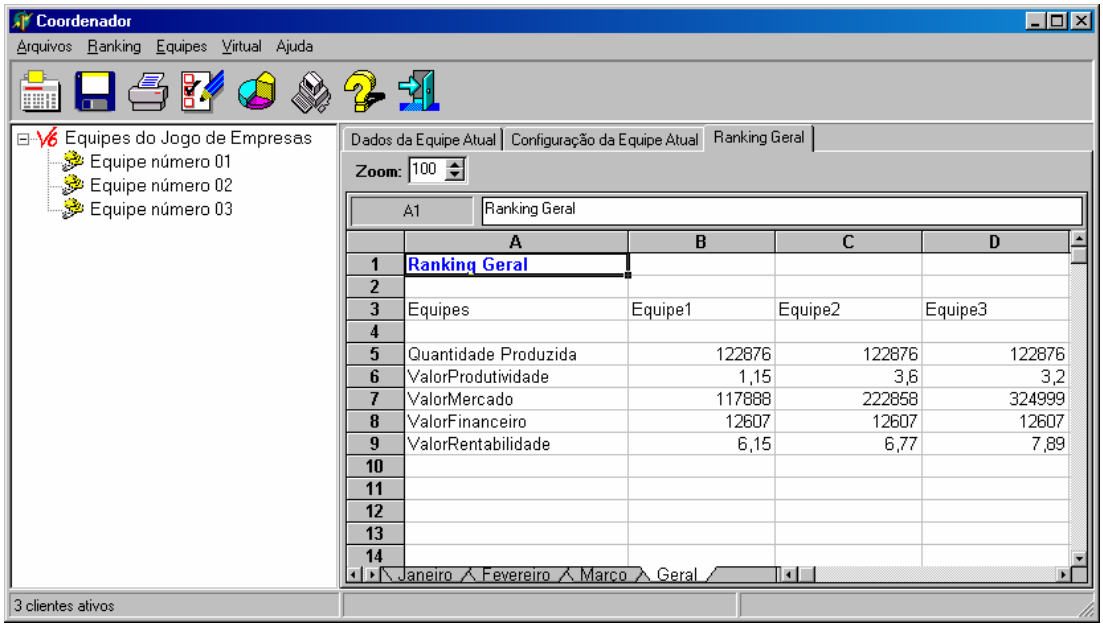
a) Arquivo

- Salvar: salva a planilha atual;
- Salvar Como: salva a planilha atual com outro nome;
- Imprimir: imprime a planilha atual;
- Sair: finaliza o sistema;

b) Ranking

- Processar Ranking: calcula o ranking atual das equipes;
- Gerar Planilha Ranking: gera planilha com o demonstrativo do ranking atual das equipes, conforme figura 34;

Figura 34 – Ranking Geral

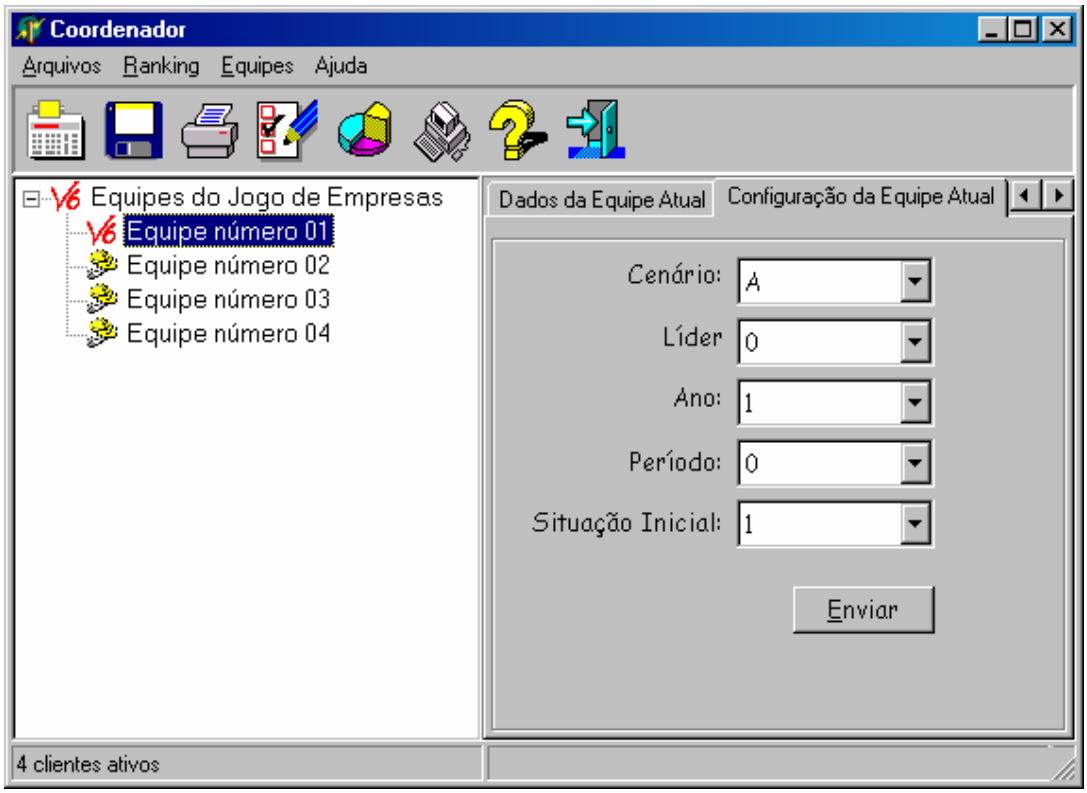


- Gerar Gráfico Ranking: gera gráfico com o demonstrativo do ranking atual das equipes;

c) Equipes

- Configuração: configura a equipe atual conforme figura 35.

Figura 35 – Configuração da equipe selecionada



d) Virtual: informações sobre o software;

e) Ajuda: ajuda e auxílio ao usuário do sistema.

É importante destacar que além da implementação deste módulo, foram feitas alterações no sistema já existente, sem, entretanto, modificar sua interface.

As figuras 36, 37 e 38 mostram os gráficos gerados pelo sistema:

Figura 36 – Gráfico de resultados financeiros

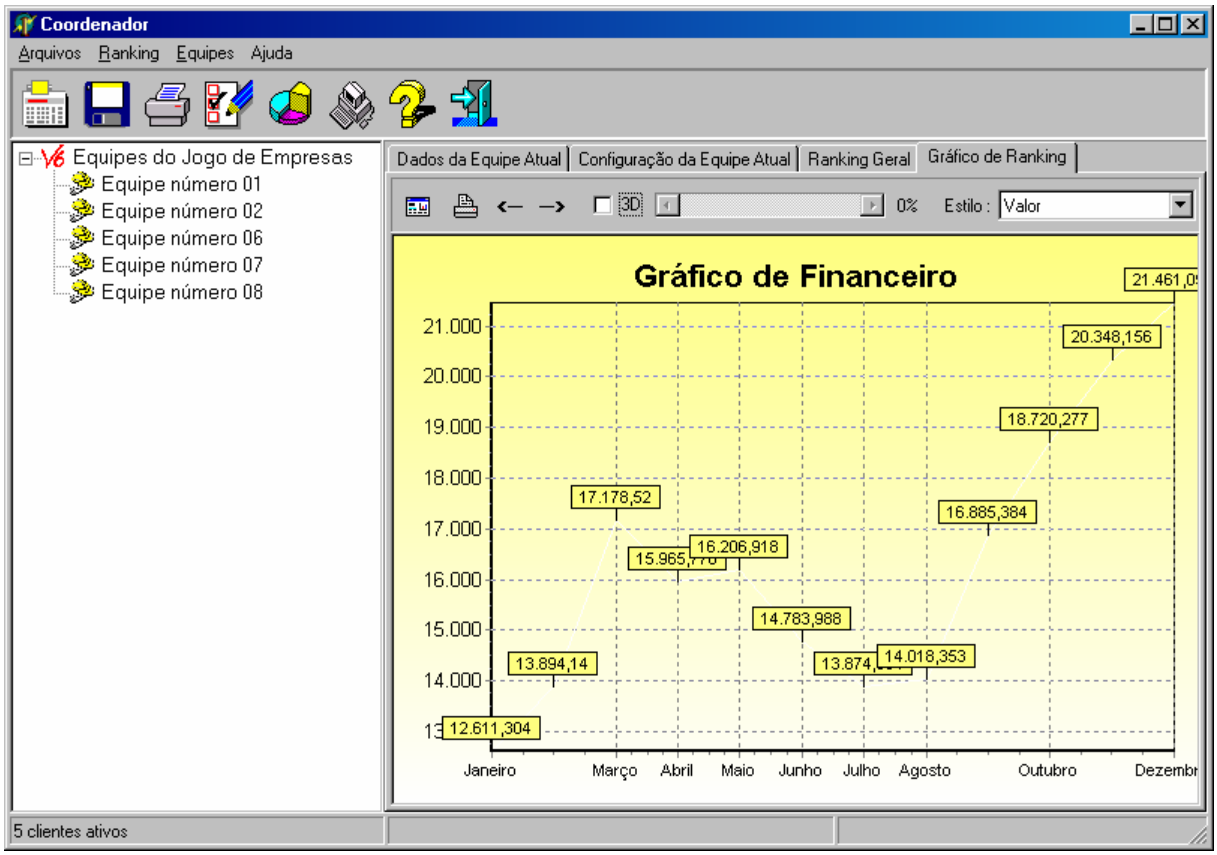


Figura 37 – Gráfico de Mercado

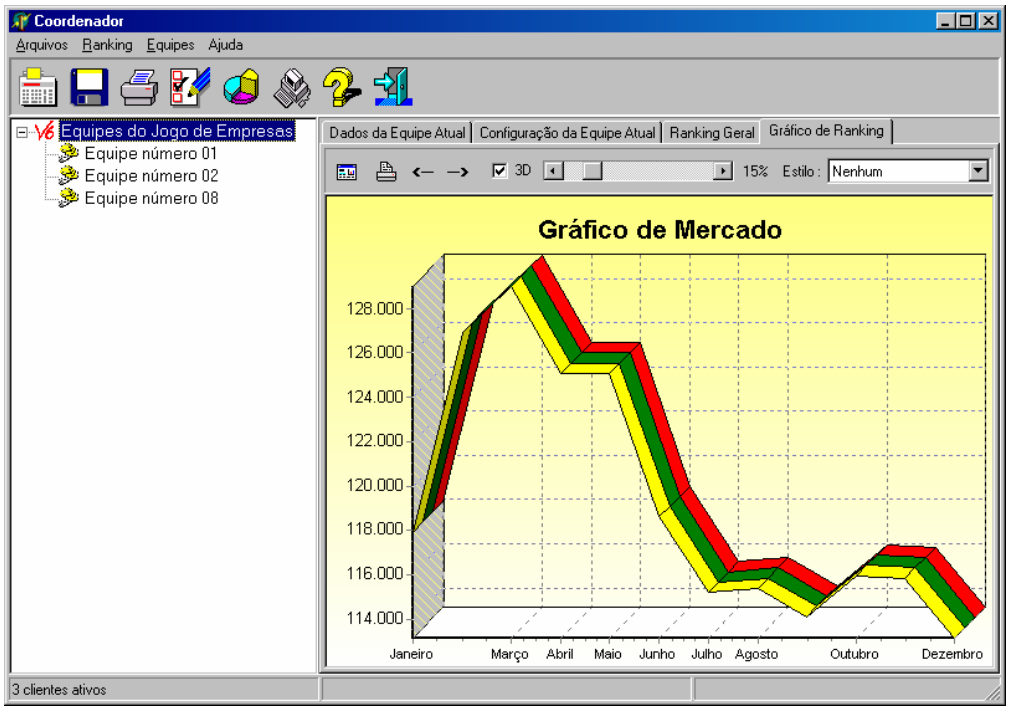
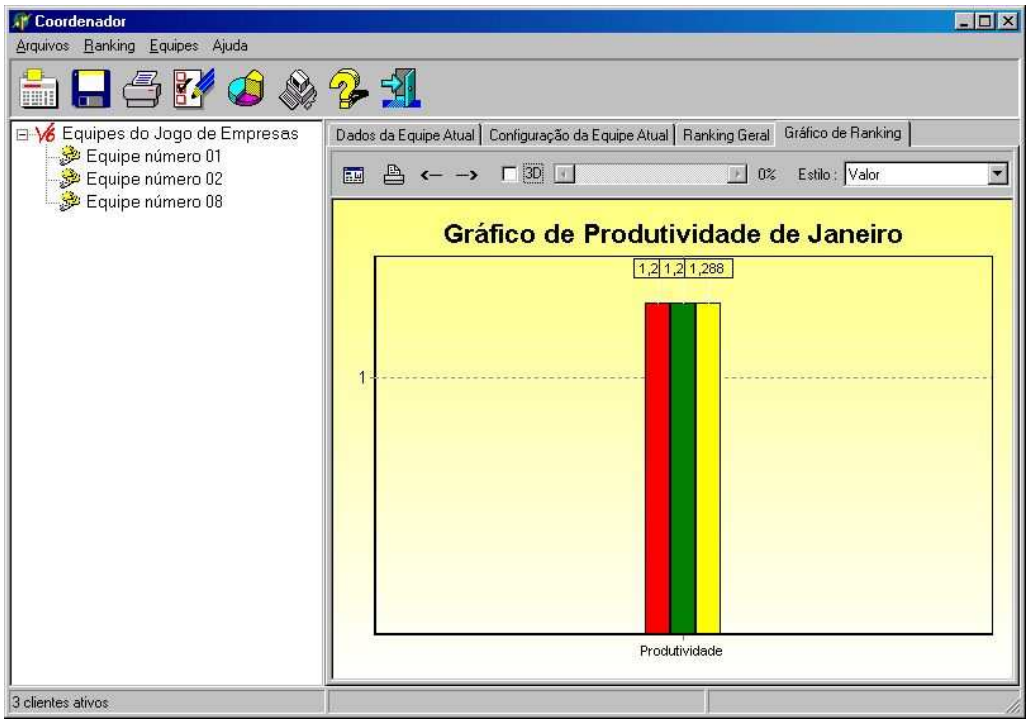


Figura 38 – Gráfico de Produtividade



5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

A necessidade de utilizar aplicativos baseados em objetos distribuídos cresce cada vez mais, e acredita-se que os mesmos virão a substituir os softwares tradicionais.

Para justificar essa afirmação, tem-se vários fatos, um deles diz respeito a atual diversidade de Sistemas Operacionais. A Microsoft por muito tempo dominou esse mercado e hoje isso não é mais verdade, pois muitos outros sistemas vêm surgindo. Por isso a necessidade de desenvolver aplicativos baseados em objetos distribuídos, uma vez que serão compatíveis com qualquer plataforma. Outra justificativa refere-se aos ambientes distribuídos, os quais estão cada vez mais freqüentes nas grandes empresas, e requerem softwares distribuídos.

O principal objetivo deste trabalho, que era implementar um módulo de auxilio para o coordenador e demonstração de resultados foi atingido. Os benefícios que o mesmo obteve foram a agilização do processo de cálculo de ranking do sistema virtual (que passou de manual para informatizado) e um melhoramento na demonstração das decisões tomadas pelas equipes, pois agora os resultados são demonstrados imediatamente e de diversas formas (vários gráficos e planilhas).

Referente aos objetivos secundários, que eram facilitar e agilizar o uso do software VIRTUAL, por parte do coordenador e verificar a efetividade do uso do padrão CORBA com *softwares*, é que o padrão CORBA tornou possíveis ambos os objetivos, pois o sistema ficou mais fácil, ágil de operar, coordenar e a tecnologia utilizada (CORBA), atendeu bem as expectativas em relação a automação e comunicação dos módulos.

A implementação de objetos CORBA no ambiente Delphi muito interessante, pois os assistentes de programação desta ferramenta automatizam alguns processos, tornando o desenvolvimento ágil e correto e a manutenção da aplicação fácil de ser feita. Um exemplo disto é o assistente de construção das classes CORBA no qual somente os parâmetros são informados e as classes são criadas automaticamente pelo ambiente.

5.2 DIFICULDADES ENCONTRADAS

As maiores dificuldades encontradas na fase do levantamento bibliográfico foram a determinação da escolha de padrão para implementação do protótipo (DCOM ou CORBA), pois ambos poderiam solucionar o problema proposto, entretanto, o CORBA demonstrou ser a solução ideal.

Quanto à implementação, foram encontradas pequenas dificuldades também oriundas do aprendizado do padrão CORBA e seus vários serviços como: comunicação entre o módulo cliente e o módulo servidor, passagem de parâmetros no formato CORBA e controle de instâncias ativas.

5.3 SUGESTÕES

Uma sugestão é disponibilizar o uso dos cliente e servidor na Internet, já que o padrão CORBA suporta muito bem este tipo de ambiente.

É sugerido também que seja feito um controle maior em relação a tolerância a falhas, como um servidor de auxílio em outra estação que contenha os dados do servidor principal, e no caso de um problema de rede, este servidor auxiliar, continue a executar o VIRTUAL normalmente.

Outra preocupação, é com relação a independência de plataforma. Não foram realizados testes em outras plataformas. Principalmente porque o ambiente Delphi ainda é suportado somente pela plataforma Windows, entretanto, futuramente ele deverá suportar outras plataformas já existentes, o que possibilitará o uso do VIRTUAL em outras plataformas.

Outra sugestão é que sejam gerados controles de performace para maior acompanhamento por parte do coordenador como: controle de tempo das decisões e acompanhamento das decisões.

REFERÊNCIAS BIBLIOGRÁFICAS

[BOR1999] BORLAND INTERNATIONAL INC. **Delphi user's guide**. Scotts Valley : Borland, 1999.

[CAN2000] CANTU, Marco. **Dominando o Delphi 5.0 - a bíblia.** São Paulo : Makron Books, 2000.

[CAP1999] CAPELETTO, Johni Jeferson. **Comunicação entre objetos distribuídos utilizando a tecnologia CORBA (common object request broker architecture).** Blumenau, 1999. TCC (Bacharelado em Ciências da Computação) Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.

[CAR1997] CARDOSO; COELHO; HONORATO. **Objetos Distribuídos: uma ponte entre aplicativos.** Dissertação de Mestrado. CESUPA: Belém- Pa, 1997.

[FUR1998] FURLAN, José Davi. **Modelagem de objetos através da UML: análise e desenho orientados a objeto.** São Paulo : Makron Books, 1998.

[GRA1993] GRAMIGNA. M.R. Miranda. **Jogos de empresas.** São Paulo: Makron Books, 1993.

[HON1999] HONORATO, Paulo Francioli Junior. **Objetos distribuídos**. 1999. Endereço eletrônico : http://www.nautilus.com.br. Data da consulta: 10/03/2000.

[KOP1989] KOPITTKE, Bruno H.. **Jogos de empresas: combinação de dois jogos.** 9º ENEGEP. Porto Alegre, 1989.

[KRU1999] KRUPS, Roberto T. **Objetos distribuídos** 1999. Endereço Eletrônico: http://planeta.terra.com.br/informatica. Data da consulta: 12/09/2000.

[MEN1997] MENDES, Maria de Lourdes de Melo Salmito. **O modelo GS-RH: uma integração de jogos de empresas para treinamento e desenvolvimento gerencial.** Dissertação de Mestrado. Universidade Federal de Santa

Catarina. Florianópolis, 1997.

[MIR1998] MIRANDA, José Carlos. **The Corba / The Common Object Reques Broquest Architecture** 1998. Dissertação de Mestrado. UFPA: Belém – Pará.

[ORR1996] ORFALI, R.; HARLEY, O.; EDUARDS, J. **The Essential Client/Server Survival Guide**. Wiley Computer Publishing, 2 ed., 1996.

[RIC1998] RICARDO D. Mello. **Aplicativos distribuídos** 1998. Endereço Eletrônico: http://members.nbci.com/hp_ricardo/aplidist.htm. Data da consulta: 01/09/2000.

[TAN1977] TANABE, Mario. **Jogos de empresas.** Dissertação de Mestrado. Faculdade de Economia e Administração. Universidade de São Paulo.São Paulo, 1977.

[VAS1998] VASCONCELOS, V. **Corba O quê?** Dissertação de Mestrado. UFPA: Belém-Pa, 1998.

[VIR1996] VIRTUAL: **Manual do jogador.** Blumenau: IPS - Universidade Regional de Blumenau, 1996.