UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

SIMULADOR DE EMPRESAS LÍDER: PROTÓTIPO DE UM SISTEMA BASEADO EM AGENTES

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

MARCOS VIRGILIO DA COSTA

BLUMENAU, JUNHO/2000.

2000/1-11

SIMULADOR DE EMPRESAS LÍDER: PROTÓTIPO DE UM SISTEMA BASEADO EM AGENTES

MARCOS VIRGILIO DA COSTA

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. Everaldo Arthur Grall

_____

Prof.

“As pessoas mais felizes não tem necessariamente o melhor de tudo na vida... Talvez elas sejam somente boas em fazer o melhor de tudo, com o que a vida lhes oferece pelo caminho”.

Dedico este trabalho a todas as formas de vida que de alguma forma conspiraram ao meu favor, sejam elas compreensíveis ou não pelo homem.

Agradecimentos

Agradeço aos meus pais Marcos Sandoval da Costa e Juçara Virgili Costa, por não terem se afastado de mim um sequer minuto desde que nasci, e por tudo, não tudo que fizeram, mais por tudo que foram, serão e significarão para mim.

A minha segunda mãe e avó Orlanda Burdgardt Virgili, que me envolveu em seu manto de carinho e doçura, nos momentos em que meus pais não estiveram fisicamente ao meu lado.

Meu irmão Marcos Eduardo Virgili, por cuidar de nossos pais enquanto eu estive afastado, mantendo nosso lar sempre transbordando de felicidade.

Meus tios Herbert Willecke Neto, Lúcia Virgili Willecke, Vilmar Quintino e Carmem Virgili Quintino que me deram todo apoio em todos os aspectos, nos momentos mais difíceis, e continuam me incentivando até hoje.

Ao professor Maurício Capobianco Lopes, por acreditar e confiar no meu potencial profissional desde o início de minha carreira acadêmica.

Sumário

Lista de Figuras

Lista de Quadros

Lista de Abreviaturas

FURB – Fundação Universidade Regional de Blumenau

IA – *Inteligência Artificial*

UFSC – Universidade Federal de Santa Catarina

UML – *Unified Modeling Language*

Resumo

Este trabalho tem como objetivo principal o desenvolvimento de uma versão do simulador de empresas Líder, baseado em uma sociedade de agentes, provendo uma expansão do potencial didático e funcional desta ferramenta de treinamento em gestão de recursos humanos. A utilização de agentes objetiva transformar o sistema mais próximo do modelo real. Para a implementação do protótipo, adotou-se a metodologia UML, utilizando a ferramenta Rational Rose 2000 como suporte da mesma, e o ambiente de desenvolvimento é o Delphi.

Abstract

The main goal of this work is the development of a new version about Líder business game based on agent society, providing an expansion about the didactic and functional potential of this management human resource engine training. The usage of agent technology is aimed to change the system to the closest possible of the real model. For the prototype implementation, was adopted the UML methodology, using Rational Rose 2000 tool for support it, and the build tool used was Delphi.

1. Introdução

O simulador de empresas Líder [LOP1994], é um sistema direcionado para treinamento em gestão de recursos humanos, com base no relacionamento do gerente de produção de uma empresa com seus colaboradores. Este relacionamento é baseado em decisões sobre a política de trabalho a ser adotada pela empresa, e, também sobre a forma de atuação sobre cada colaborador individualmente.

O objetivo dos participantes do treinamento é gerenciar uma equipe de colaboradores, fazendo com que eles sintam-se motivados para desempenhar suas funções, e atinjam as metas estabelecidas pela empresa.

O Líder é, atualmente, aplicado em disciplinas de graduação e pós-graduação de diversas universidades de Santa Catarina e seu valor como ferramenta para treinamento em gestão empresarial já foi amplamente comprovado por trabalhos como [LOP1994], [TEI1996] e [SOU1997]. Entretanto, isto não impede que o sistema computacional do simulador não continue sua evolução. O surgimento de novas tecnologias a cada dia, como, por exemplo, a Inteligência Artificial (IA), pode possibilitar uma representação do modelo comportamental humano mais próximo da realidade, contribuindo significativamente para o desenvolvimento do simulador de empresas Líder.

Atualmente, uma das áreas de estudo que vem se desenvolvendo mais rapidamente em relação aos simuladores de empresas, é sua interação com técnicas de IA, principalmente para modelagem de Sistemas de Apoio a Decisão.

Algumas áreas da IA consideram, como modelo, a coletividade e não um único indivíduo. As Sociedades de Agentes são um exemplo claro da aplicação deste modelo que tem como foco o comportamento inteligente em um contexto social de agentes, isto é, como coordenar seus conhecimentos, metas e planos para resolver problemas [HÜB1996] [JEN1997].

Um agente pode ser definido como uma entidade real ou virtual que emerge num ambiente onde pode agir intencionalmente, ser hábil a perceber e representar o ambiente, e a comunicar-se com outros agentes, além de ser autônomo [HÜB1996].

Por agir intencionalmente, entende-se que o agente planeja suas ações em conformidade com suas crenças e desejos para satisfazer uma determinada intenção. O fato de ser hábil a perceber e representar o ambiente torna-o apto a interagir com o mesmo e reagir a estímulos dele recebidos. A comunicação com outro agente, por meio de algum tipo de linguagem de comunicação, caracteriza uma capacidade de interação ou habilidade social. Por autonomia entende-se que o agente tem sua própria existência, que não é justificada pela existência de outros, e age conforme suas próprias crenças, conhecimentos e capacidades.

Sendo assim, o protótipo proposto visa representar, através de Agentes Inteligentes, o contexto social interativo do simulador de empresas Líder de uma forma mais próxima dos modelos humanos. Para isso, cada atual colaborador da empresa será um agente colaborador, agregando conceitos de Agentes, bem como conceitos especificados no próprio Líder através de seu modelo matemático, determinando, assim, um comportamento autônomo e reativo para cada agente colaborador imerso na empresa.

A metodologia adotada para a especificação do protótipo é a UML com a ferramenta Rational Rose 2000. A linguagem utilizada para a implementação é o Object Pascal e o ambiente Delphi.

1.1. Objetivos

O objetivo principal deste trabalho é a implementação de um protótipo do simulador de empresas Líder, baseado em agentes.

Como objetivos específicos pode-se citar:

a) rever a modelagem do simulador para uma eventual adequação do mesmo ao contexto de agentes, sem divergir do modelo matemático original do Líder, descrito em [MAL1990], [SAL1990], [LOP1994] e [NIV1998].
b) propor uma nova forma de comunicação entre participantes e agentes e entre os próprios agentes.

1.2. Estrutura

O trabalho foi segmentado em cinco capítulos.

O primeiro capítulo permite uma contextualização sobre o trabalho, apresentando a justificativa para seu desenvolvimento e seus objetivos.

O segundo capítulo descreve o simulador de empresas Líder, possibilitando um entendimento do mesmo como ferramenta de treinamento em gestão de recursos humanos, como também considerações sobre o sistema computacional atualmente utilizado.

O terceiro capítulo é dedicado às fundamentações e conceituações sobre as tecnologias que serão utilizadas no desenvolvimento deste trabalho, como Agentes e Sociedades de Agentes.

O quarto capítulo descreve a estruturação necessária para o desenvolvimento do protótipo até seu estágio final, destacando a metodologia utilizada, a modelagem do sistema e considerações sobre sua codificação.

O quinto capítulo complementa o trabalho, apresentando as conclusões, limitações e possíveis sugestões para serem implementadas e aprimoradas.

2. O Simulador de Empresas Líder

Este capítulo apresenta os simuladores de empresas como método de ensino no treinamento gerencial, dando ênfase ao simulador Líder que modela a ciência comportamental (sociologia e psicologia) através de uma ciência exata (matemática) modelada e implementada em um sistema computacional.

2.1. Os Simuladores de Empresas

Há mais de 40 anos que os simuladores de empresas em computador conquistaram com êxito o seu espaço incontestável entre os métodos de ensino. A grande vantagem dos simuladores de empresas como instrumento didático e de treinamento é a sua capacidade de acelerar o tempo real, sendo que através da simulação se comprimem, em poucos dias, vários anos de experiência, oferecendo assim, um preparo a funcionários e estudantes para suas atividades profissionais futuras [NIV1998].

Como método de ensino, tem-se observado que nenhum outro permite simular situações de decisões tão interessantes, com tamanha participação e interesse dos alunos. Segundo estudo mencionado em [NIV1998], 90% das equipes participantes do Líder considerou a experiência positiva e com elevado nível de comprometimento.

A aplicação de simuladores de empresas como instrumento de treinamento exige, dos participantes do jogo, a prática da arte do planejamento, bem como exerce e desenvolve a habilidade da tomada de decisão em nível de alta e média gerência/administração [LOP1994].

Um histórico sobre a evolução dos simuladores de empresas, bem como suas características e vantagens podem ser vistas em [MAL1990], [SAL1990], [LOP1994] e [NIV1998].

2.2. O simulador Líder

O simulador de empresas Líder apresenta, como principal finalidade, propiciar aos participantes, um ambiente empresarial hipotético para estimular o aprendizado ou o aprimoramento das habilidades gerenciais nos recursos humanos, fundamentalmente no que diz respeito à motivação e liderança.

Este simulador de empresas surgiu preenchendo a lacuna existente na área de desenvolvimento de recursos humanos utilizando-se dos recursos da informática. Os simuladores disponíveis no mercado nacional nesta área, antes do Líder, eram de caráter manual e implementados através de formulários e tabelas. Eram simuladores simples e concentravam-se num só aspecto e, possivelmente por serem manuais e não computacionais, mostravam a escassez de variáveis interagindo nos modelos [NIV1998].

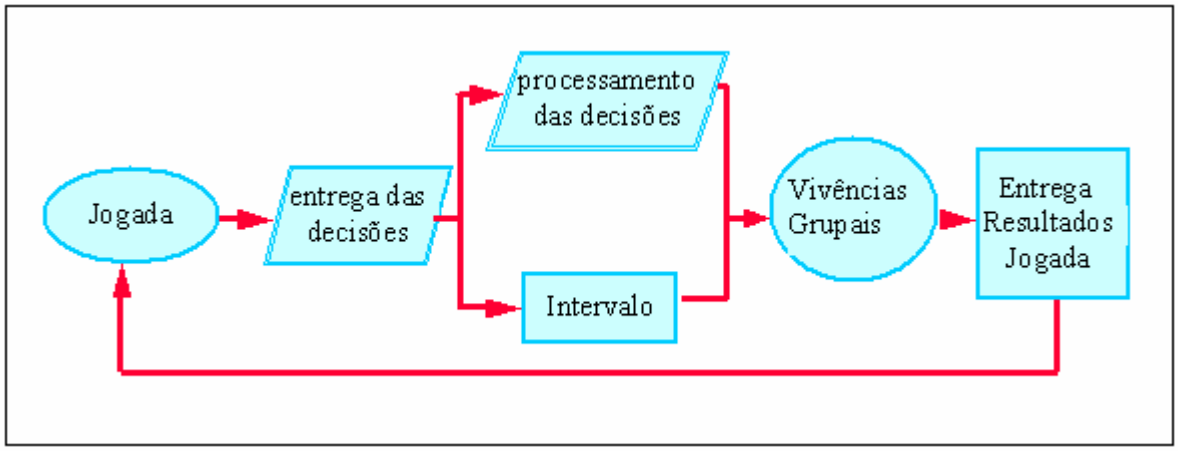
O objetivo do Líder consiste em oferecer a oportunidade de experimentar a aplicação prática da teoria comportamental. Sua intenção é aplicar, principalmente, os conceitos da teoria e técnicas da Liderança Situacional descritas por Hersey e Blanchard, a Hierarquia das Necessidades de Maslow e a teoria de Motivação - Higiene de Herzberg, transformando-os de teóricos e descritivos em práticos e prescritivos [NIV1998].

A aplicação do simulador de empresas Líder, não se resume apenas na utilização de um sistema computacional. Normalmente uma entidade externa ao sistema, um coordenador denominado Animador, geralmente o professor da disciplina, apresenta o sistema, sua finalidade e a forma correta de utilização.

Após uma apresentação e conceitualização do simulador aos participantes (processo não computacional), os animadores definem as empresas fictícias, escolhendo seu quadro de colaboradores, também fictícios, previamente definidos dentro do sistema. Os colaboradores dispõem-se em um organograma hierárquico empresarial.

O simulador, deste ponto em diante, consiste basicamente em tomadas de decisões sucessivas sobre os colaboradores que lhe são subordinados. Estas decisões são introduzidas no sistema computacional que, após a simulação, fornece as conseqüências das mesmas aos participantes. Os resultados servir-lhes-ão de base para a tomada de novas decisões. (figura 1).

Figura 1 - Dinâmica da Aplicação do Jogo Líder.



No simulador, assim como na prática, o clima organizacional depende de certas variáveis, consideradas entradas do sistema. Estes fatores influenciam o comportamento das pessoas, provocando motivação entre os colaboradores, diferentes níveis de satisfação e produtividade, e estimulando-os para produzirem o resultado final em termos de eficiência [NIV1998].

As variáveis dependentes constituem o clima organizacional da empresa. Na medida em que as variáveis de entrada produzirem influência positiva nas variáveis dependentes, maior será a eficiência dos colaboradores, quer dizer a capacidade dos mesmos de fazer a tarefa corretamente. Assim como, quanto mais influência negativa as variáveis dependentes apresentarem em conseqüência das variáveis de entrada, menor será a eficiência dos colaboradores [NIV1998].

Este processo define um comportamento para os colaboradores, e conseqüentemente um comportamento para a empresa. Se os colaboradores forem menos eficientes ou estiverem menos motivados, a produção da empresa tende a diminuir, abalando, conseqüentemente, a situação financeira da mesma.

Os participantes podem criar e desenvolver climas organizacionais, através de intervenções no seu estilo gerencial, no sistema de administrar pessoas, no projeto de trabalho, no sistema de motivação, no treinamento de sua equipe, no seu estilo de liderança, no sistema de remuneração, etc. Os participantes decidem, a partir de um amplo leque de possibilidades (incentivos), por aquelas que julgar mais capazes de atender às diversas

necessidades de seus colaboradores. Determinam, ainda, níveis de desempenho de produção desejáveis e os estilos de poder e de liderança [LOP1994], [NIV1998] (figura 2).

Figura 2 - Decisões a serem tomadas pelo participante

DECISÕES SOBRE A EMPRESA A NÍVEL GLOBAL	DECISÕES SOBRE OS FUNCIONÁRIOS A NÍVEL INDIVIDUAL
Alimentação	Metas de produção
Melhoria ambiental e ergonométrica	Estilo de liderança
Consultoria de job design	Base de poder
Lanches	Locação de pessoal
Intervalos de descanso	Treinamento antes de uma promoção
Plano de saúde	Treinamento específico para um trabalho
Redução no horário de trabalho	Treinamento de Liderança
Reuniões informais	Relatório do perfil das necessidades e maturidades
Promoções esportivas por funcionário	Aumento Salarial
	Prêmio

Fonte [NIV1998].

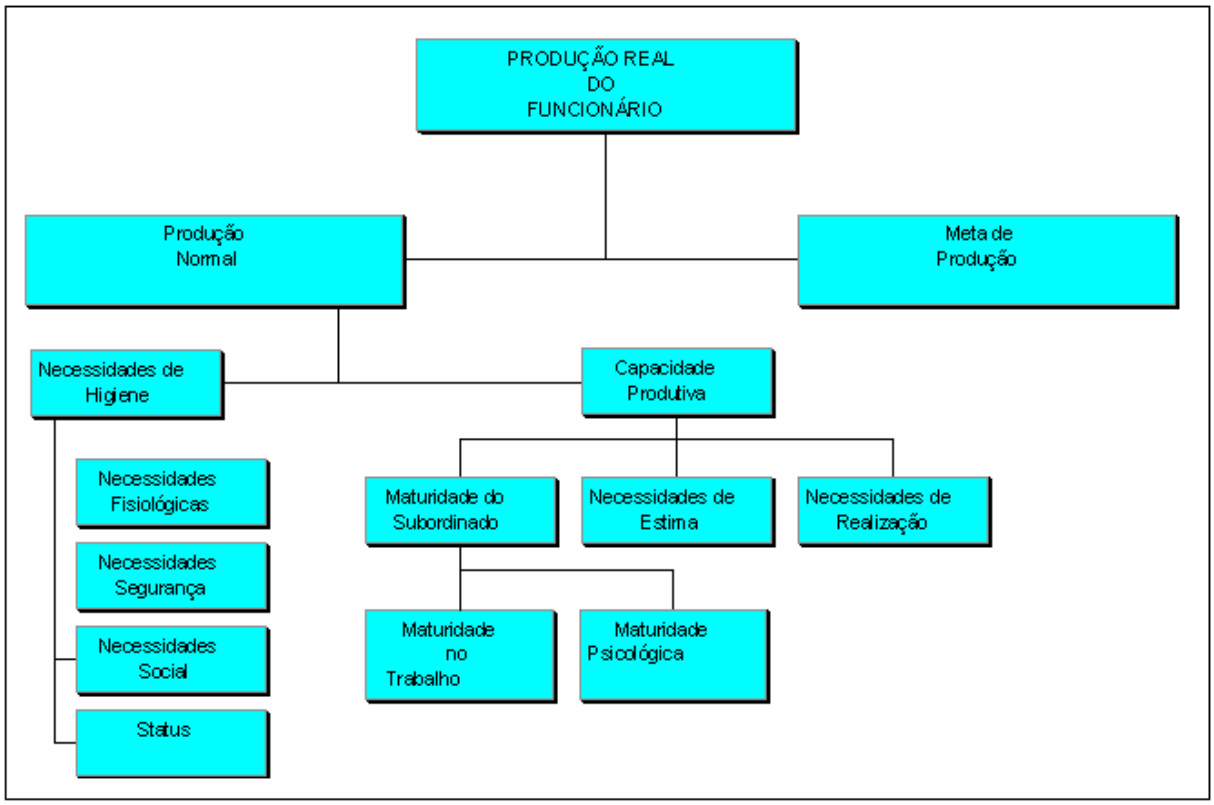
2.3. O modelo matemático

Este item apresenta a estrutura do modelo matemático do simulador de empresas Líder, ou seja, a forma na qual os diversos cálculos estão interligados e hierarquicamente posicionados para determinar a produção real individual de cada colaborador pertencente à empresa.

O modelo atualmente em vigor difere bastante do definido nos trabalhos de [SAL1990] e [MAL1990] mentores intelectuais da primeira versão do simulador de empresas Líder. As sucessivas melhorias e aperfeiçoamentos propostos por [LOP1994] e [NIV1998] aos modelos originais, resultaram na seguinte estrutura para o modelo, definido na figura 3.

É importante ressaltar, que cada fator representado na estrutura da figura 3, representa um determinado conjunto de cálculos e procedimentos descritos no modelo matemático do Líder. Em [NIV1998], existem diagramas que representam detalhadamente alguns fatores presentes na figura 3, provendo um entendimento mais detalhado sobre o cálculo de cada fator.

Figura 3 - Modelo Matemático do Simulador de Empresas Líder.



A estrutura da figura 3 representa graficamente a hierarquia dos fatores que influenciam no cálculo da produção final do colaborador, ou seja, cada fator calculado é utilizado diretamente no cálculo do fator posicionado hierarquicamente acima, e assim sucessivamente.

Pela figura 3 pode-se verificar o impacto de um fator no cálculo de outros fatores. A seguir é apresentado um exemplo de como é feito o cálculo da produção normal de um colaborador.

A produção normal é definida em função de dois fatores:

1. A capacidade produtiva, que está diretamente relacionada com os níveis de maturidade psicológica e no trabalho, e o grau de necessidade motivacional (estima e realização) do funcionário em questão;

2. O grau de satisfação das necessidades de higiene (fisiológicas, segurança, social e status).

Assim: $PN = NH \times CP$

Onde:

- PN é a produção normal do funcionário,
- NH é o grau de necessidades de higiene, e;
- CP é a capacidade produtiva do funcionário.

É importante destacar que nem todos os cálculos são de simples multiplicações. A descrição detalhada sobre o modelo matemático pode ser encontrada em [MAL1990], [SAL1990], [LOP1994] e [NIV1998].

2.4. *O sistema computacional atual*

Desde sua primeira especificação, através de fluxogramas, idealizada por [SAL1990] e [MAL1990], o Líder sofreu uma série de alterações, muitas destas necessárias para acompanharem as evoluções no modelo matemático.

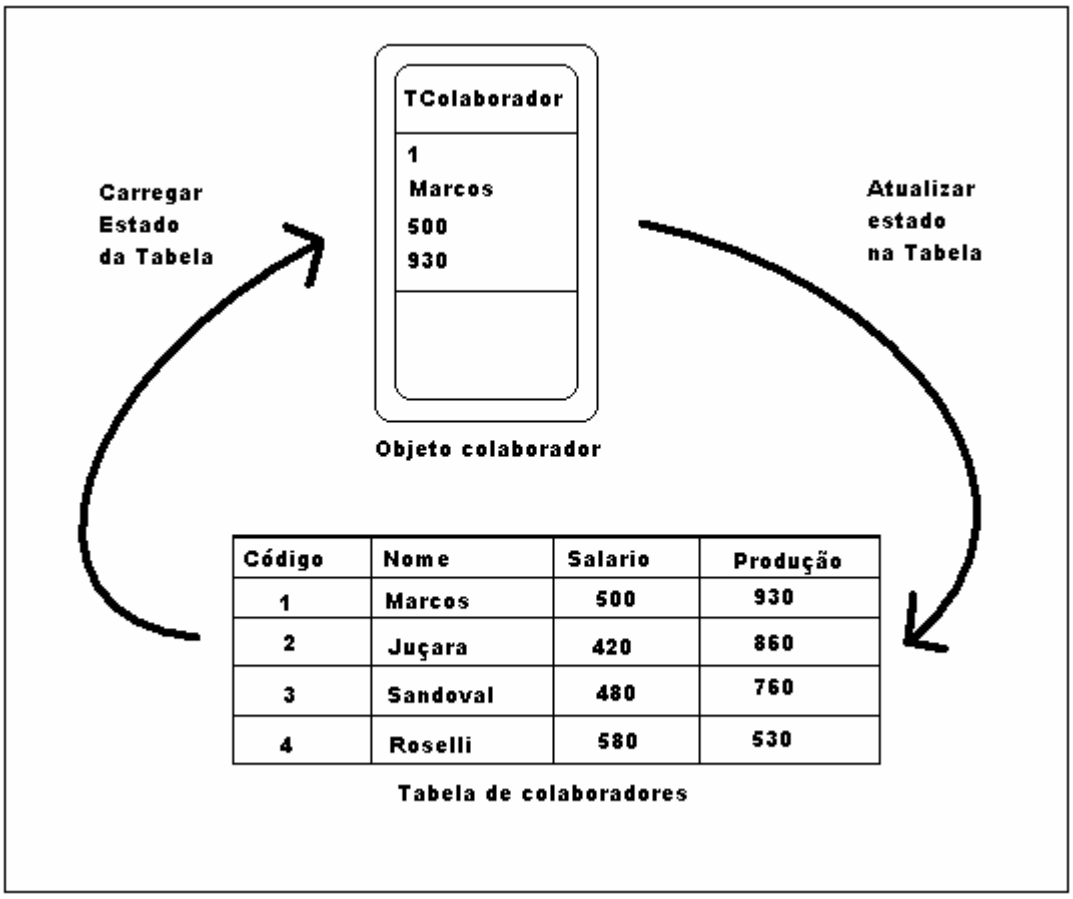
A primeira implementação em linguagem computacional foi proposta e implementada na Universidade Federal de Santa Catarina por [LOP1994]. Esta primeira versão do sistema, desenvolvida em Pascal, utilizava arquivos binários para armazenamento das informações relevantes à simulação, permitindo, assim, o salvamento do estado das empresas para uma eventual retomada de contexto, permitindo a continuidade da simulação. Esta versão não era destinada aos participantes do jogo, os quais tomavam decisões através de uma ficha preenchida manualmente em papel. O sistema era utilizado pelo animador, o qual introduzia as informações dos participantes no sistema para obter o processamento das decisões, e conseqüentemente avaliar os resultados.

Pesquisas realizadas na Universidade Regional de Blumenau, no sentido de melhorar o sistema atualmente em vigor, transportaram-no para a plataforma Windows, utilizando o ambiente de desenvolvimento Delphi, a linguagem Object Pascal e substituindo os arquivos binários por Bancos de Dados.

Esta nova estrutura resultou em uma amarração do código fonte do processamento das variáveis decisórias, anteriormente implementado em linguagem computacional, a scripts em SQL.

Os estados dos colaboradores passaram a ser armazenados em tabelas identificados que utilizam o código de colaborador como chave, apenas uma estrutura dinâmica em memória é utilizada durante a execução do sistema. Esta estrutura mantém o estado de um colaborador durante seu processamento, buscando seu estado armazenado no banco. Processadas as alterações, o mesmo atualiza seu novo estado no banco de dados, ficando livre para o processamento de outro colaborador. A figura 4 ilustra o processo de carga do estados dos colaboradores.

Figura 4 - Trocas de estado do objeto colaborador



A conseqüência disto é a necessidade de uma contínua atualização do banco de dados após cada processamento, pois os resultados do processamento do período atual serão a base para o processamento do próximo período e assim por diante. Este constante acesso ao banco de dados, desviou recursos do equipamento, antes direcionados ao objetivo principal que é o processamento do modelo matemático, para procedimentos de acesso a banco de dados.

Se comparada à versão anterior, baseada em arquivos binários, a atualmente em vigor é visivelmente mais lenta. Além disto, como a ferramenta de implementação utilizada foi o Delphi com o banco de dados Paradox, a instalação do BDE (*Borland Database Engine*) que é a ferramenta da Borland para promover o acesso do aplicativo ao banco de dados, torna-se um requisito para que o sistema possa ser executado.

Uma vez que esta versão é destinada ao participante, fatores como interatividade e facilidade de utilização devem ser considerados. Neste aspecto, o sistema pode ser visto como simples de operar, principalmente por sua interface composta basicamente por três telas:

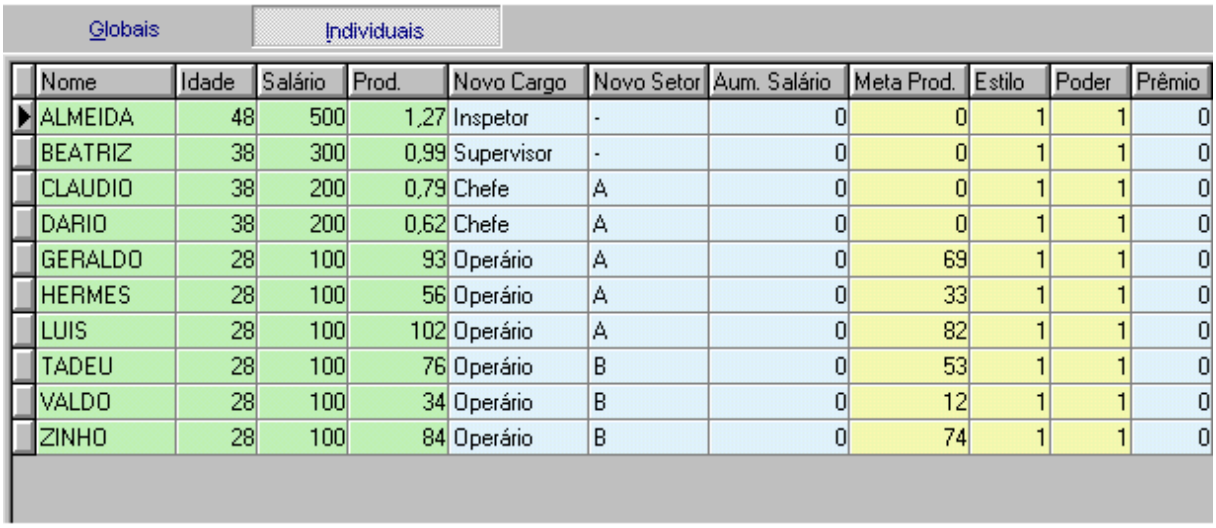
a) tomada de decisões globais (figura 5), onde o participante seleciona quais opções irá aplicar, dentre as disponíveis, a todos os colaboradores da empresa. Para aplicar o item desejado, basta selecioná-lo, pressionando o botão direito do mouse sobre o mesmo, deixando o quadrado branco com um sinal semelhante ao da opção “Job Design para Setor A”;

Figura 5 - Tela de decisões globais do sistema atual

b) tomada de decisões individuais, onde o participante pode aplicar as opções disponíveis (figura 6) a cada colaborador individualmente. Neste caso o nome de cada colaborador é apresentado, seguido de algumas informações atuais

(idade, salário, produção) que não podem ser alteradas pelo usuário, e outras que definem as decisões sobre ele (novo cargo, novo setor, aumento de salário, etc...);

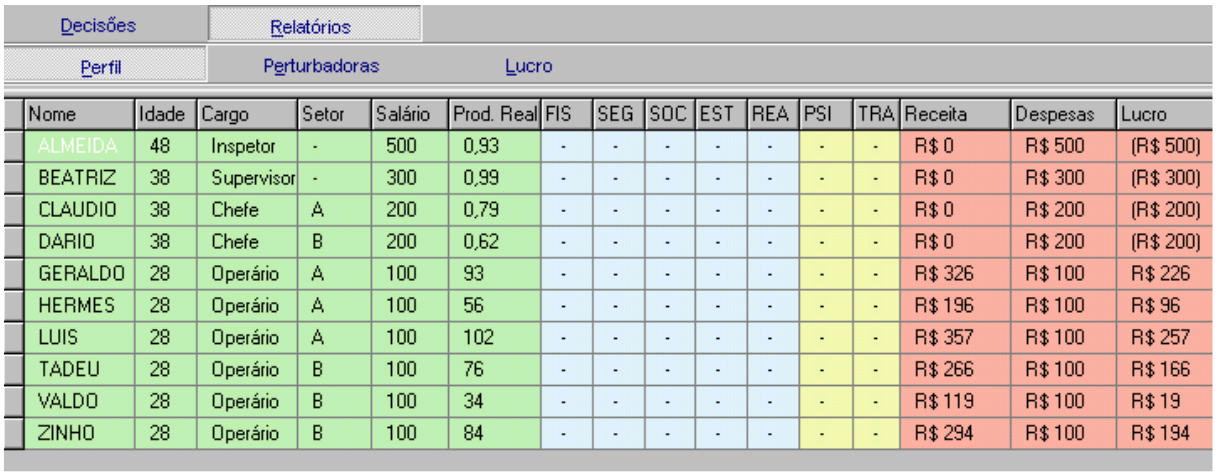
Figura 6 - Tela de decisões individuais do sistema atual



Nome	Idade	Salário	Prod.	Novo Cargo	Novo Setor	Aum. Salário	Meta Prod.	Estilo	Poder	Prêmio
ALMEIDA	48	500	1,27	Inspetor	-	0	0	1	1	0
BEATRIZ	38	300	0,99	Supervisor	-	0	0	1	1	0
CLAUDIO	38	200	0,79	Chefe	A	0	0	1	1	0
DARIO	38	200	0,62	Chefe	A	0	0	1	1	0
GERALDO	28	100	93	Operário	A	0	69	1	1	0
HERMES	28	100	56	Operário	A	0	33	1	1	0
LUIS	28	100	102	Operário	A	0	82	1	1	0
TADEU	28	100	76	Operário	B	0	53	1	1	0
VALDO	28	100	34	Operário	B	0	12	1	1	0
ZINHO	28	100	84	Operário	B	0	74	1	1	0

c) análise dos resultados, (figura 7) que permite uma verificação dos efeitos decorrentes das decisões tomadas no período anterior. Nesta área podem ser verificados o perfil dos funcionários, suas situações perturbadoras e o lucro da empresa. Os resultados são obtidos a partir do processamento das decisões.

Figura 7 - Tela de análise dos resultados do sistema atual



Nome	Idade	Cargo	Setor	Salário	Prod. Real	FIS	SEG	SOC	EST	REA	PSI	TRA	Receita	Despesas	Lucro
ALMEIDA	48	Inspetor	-	500	0,93	-	-	-	-	-	-	-	R$ 0	R$ 500	(R$ 500)
BEATRIZ	38	Supervisor	-	300	0,99	-	-	-	-	-	-	-	R$ 0	R$ 300	(R$ 300)
CLAUDIO	38	Chefe	A	200	0,79	-	-	-	-	-	-	-	R$ 0	R$ 200	(R$ 200)
DARIO	38	Chefe	B	200	0,62	-	-	-	-	-	-	-	R$ 0	R$ 200	(R$ 200)
GERALDO	28	Operário	A	100	93	-	-	-	-	-	-	-	R$ 326	R$ 100	R$ 226
HERMES	28	Operário	A	100	56	-	-	-	-	-	-	-	R$ 196	R$ 100	R$ 96
LUIS	28	Operário	A	100	102	-	-	-	-	-	-	-	R$ 357	R$ 100	R$ 257
TADEU	28	Operário	B	100	76	-	-	-	-	-	-	-	R$ 266	R$ 100	R$ 166
VALDO	28	Operário	B	100	34	-	-	-	-	-	-	-	R$ 119	R$ 100	R$ 19
ZINHO	28	Operário	B	100	84	-	-	-	-	-	-	-	R$ 294	R$ 100	R$ 194

Entretanto, esta simplicidade, acentua uma deficiência quanto à interatividade com os usuários do sistema. O período entre o término do processamento de um período e a próxima tomada de decisões, não disponibiliza nenhum retorno ao usuário, ou seja, o participante não

tem como saber o efeito de suas decisões no decorrer do período atual nem a nível individual nem global, até o processamento do período corrente terminar.

Desta forma, o sistema torna-se muito estático, havendo uma relação muito direta entre decisão e processamento. O ideal é que sistema passe a ser mais dinâmico, ou seja, para alterar o estado de um colaborador não deve ser necessário que haja um processamento das decisões. Ao contrário, o colaborador deveria poder processar seu modelo quando bem entendesse.

3. Agentes

Este capítulo apresenta a fundamentação das técnicas e tecnologias sobre agentes que são utilizadas na implementação do protótipo.

3.1. Definições

Dentre as diversas definições existentes sobre agentes, incluindo a definição de Hübner [HÜB1996] citada no início do texto, subseqüentemente serão citadas algumas que tenham uma maior afinidade com a finalidade deste trabalho, no que se refere à utilização e implementação desta tecnologia.

Segundo [SMI1994], "O agente é uma entidade de software persistente dedicada para um propósito específico. Persistente distingue agentes de sub-rotinas; agentes têm suas próprias idéias de como efetuar tarefas".

Maes diz que "agentes devem agir autonomamente para cumprir um conjunto de objetivos" [MAE1995].

"Um sistema de computador baseado em hardware ou software que desfruta as propriedades de: autonomia, capacidade social, reatividade e pro-atividade". Esta definição escrita por Wooldridge e Jennings [JEN1998], muito tem contribuído para a teoria de agentes, principalmente no que diz respeito à capacidade social dos mesmos.

Os agentes podem ser dispostos de maneira coletiva, formando sociedades. As sociedades de agentes, são um exemplo de idealização, na qual são considerados tanto os aspectos e comportamentos individuais, quanto à interação entre diversos agentes presentes em um contexto social.

3.2. Sociedades de agentes

Uma sociedade de agentes pode ser definida, numa perspectiva muito abrangente, como uma tripla ⟨ ***A, M, L***⟩ com um conjunto de agentes (***A***) que compartilham um conjunto de meios (***M***) e um conjunto de linguagens (***L***) que permitem comunicação entre eles. Não existe, nesta definição, nenhuma necessidade de se estabelecer alguma restrição como a homogeneidade de conhecimento, estrutura ou habilidades. Uma sociedade de agentes pode ser estabelecida com agentes que compartilhem apenas uma linguagem através da qual interagem.

Os agentes podem ser classificados em homogêneos e heterogêneos. Os agentes homogêneos possuem as mesmas habilidades, conhecimentos e a mesma estrutura. No mundo real, o que mais se observa, entretanto, são agentes heterogêneos. Portanto, qualquer processo de interação entre agentes que modele algum aspecto do mundo real, deve considerar agentes heterogêneos.

Moulin & Chaib-Draa [JEN1998] classificam a heterogeneidade em sociedades de agentes em três níveis:

- **baixa heterogeneidade** no caso que os agentes difiram apenas nos recursos disponíveis;
- **média heterogeneidade** no caso que os agentes difiram nos métodos de resolução de problemas;
- **alta heterogeneidade** no caso que os agentes compartilham apenas a linguagem de interação.

Esta noção de heterogeneidade pode ser entendida como uma medida do grau de *diferença estrutural* dos agentes em uma sociedade.

Em [JEN1998], são citados exemplos de sistemas baseados em sociedades de agentes, utilizados em sistemas de controle de tráfego aéreo, tráfego de veículos, controles de produção, gerenciamento de documentos e serviços financeiros. Um destes é o OASIS, sistema de controle de tráfego aéreo, que modela inclusive considerações sobre fenômenos meteorológicos e trajetórias de pouso onde haja maximização na economia de combustível das aeronaves. O SWARMM, sistema utilizado pelo Royal Australian Air Force no

treinamento em técnicas combate aéreo, inclui uma análise sobre a técnica utilizada pelo piloto durante o treinamento, e avalia o desempenho dos mesmos.

3.3. Classificação e Taxonomia

Em um contexto social de agentes, onde a heterogeneidade pode estar presente, definir algum tipo de classificação é uma forma de facilitar a compreensão de quais são as possibilidades e limites desta tecnologia. Uma possível classificação dos agentes permite, por exemplo, estabelecer que agentes podem ser ou não mais adequados para determinadas tarefas ou objetivos.

A classificação de agentes é geralmente tomada como uma tipificação baseada em um conjunto de características observáveis. Vários autores as fazem, considerando características diferentes:

Wooldridge & Jennings [JEN1997] em sua definição de agentes listam as seguintes características:

- **reatividade** é perceber o ambiente e responder de maneira oportuna (*timely fashion*) a mudanças que acontecerem nele;
- **autonomia** é a capacidade de operar sem intervenção direta de humanos ou outros e ter alguma espécie de controle sobre suas ações e estado interno;
- **habilidade social** é a capacidade de interagir com outros agentes via algum tipo de linguagem de comunicação de agentes**;**
- **pró-atividade** é a capacidade de exibir comportamentos baseados em objetivos.

Brustoloni apud [JEN1998] propõe uma classificação dos agentes em três tipos:

- **agentes regulatórios** que reagem a estímulos e com um conjunto de conhecimentos e ações pré-determinados;
- **agentes planejadores** que podem estabelecer planos para atingir as suas metas;

- **agentes adaptativos** que, além da capacidade de planejar, têm condições de adquirir novos conhecimentos sobre o domínio onde atuam e aumentar a sua gama de habilidades.

Russel e Norvig [RUS1995] classificam os programas de agentes em quatro tipos:

- **agentes reflexos simples,** que reagem ao estado do mundo apenas a partir de um conjunto de regras (condição-ação) pré-determinadas;

- **agentes que acompanham o mundo** são agentes que além de um conjunto de regras, guardam algum estado interno que dado o estado do mundo, influencia a ação a tomar;

- **agentes baseados em metas** são aqueles capazes de armazenar algum tipo de meta em sua memória e estabelecer as suas ações a partir de algum tipo de planejamento que lhe permite buscar uma ação ou conjunto de ações para alcançar as suas metas;

- **agentes baseados em utilidade** são aqueles que agem no sentido de aumentar o seu grau de felicidade ou satisfação. Mais que metas, os agentes procuram maximizar alguma função utilidade explícita. As decisões dos agentes levarão sempre em conta como poderão estar a sua satisfação dada uma determinada trajetória de evolução do mundo e uma escolha de ação feita por eles.

Parte das classificações discutidas anteriormente apresentam um elemento em comum, a tipificação dos agentes a partir de determinadas características qualitativas. Estas classificações geralmente apresentam alguma forma de ordenação que vai dos agentes mais simples (os agentes **regulatórios, reativos** ou **reflexos**), até os mais complexos (agentes **adaptativos, baseados em utilidade e sociais)**.

3.4. O meio ambiente

Quanto ao meio ambiente em que o Agente estará inserido, é importante que se defina os tipos que podem existir, pois isto interfere diretamente na complexidade do agente em questão.

Norvig e Russel [RUS1995] definem as propriedades do meio ambiente como:

- **inacessível X acessível**: um ambiente é acessível ao agente se seus aparatos são capazes de perceber o estado completo deste ambiente;
- **determinístico X não-determinístico**: um ambiente determinístico é aquele cujo próximo estado é completamente determinado através do estado atual e das ações selecionadas pelo agente;
- **episódico X não-episódico**: num ambiente episódico, as ações dos agentes são divididas em episódios. Cada episódio consiste em o agente perceber e agir e a qualidade da ação dependem do próprio episódio, não dependendo do que aconteceu em episódios anteriores;
- **estático X dinâmico**: um ambiente é dinâmico para o agente se seu estado pode mudar enquanto o agente está deliberando;
- **discreto X contínuo**: um ambiente é discreto se houver um número limitado de percepções e ações claras e distintas.

4. Desenvolvimento do Protótipo

Neste capítulo, é apresentada a modelagem do sistema proposto, a estrutura dos agentes que compõem a sociedade de agentes do sistema, o detalhamento da implementação dos mesmos, a forma como se comunicam entre si e a nova dinâmica do simulador de empresas Líder.

4.1. Metodologia de Desenvolvimento

A utilização de uma metodologia orientada a objetos, é uma conseqüência da ampla relação existente entre agentes e objetos, defendida por quase todos os autores relacionados anteriormente como [NOR1995], [WOO1998].

Dentre as diversas metodologias orientadas a objetos existentes, a adotada neste trabalho é a UML (*Unified Modeling Language*) [FUR1998]. A UML oferece mecanismos de suporte a modelos estáticos, dinâmicos e funcionais, uma vez que todos os sistemas possuem uma estrutura estática e um comportamento dinâmico. É importante ressaltar que não serão utilizados todos os mecanismos disponíveis na UML para modelagem, análise e especificação deste sistema. Será utilizado um diagrama para cada modelo (funcional, dinâmico e estático), dentre os disponibilizados pela metodologia, contemplando de forma suficiente o entendimento e documentação do sistema proposto.

Dentre os recursos utilizados está o **diagrama de *use-case*** (casos de uso), que é uma forma de descrever e definir os **requisitos funcionais** de um sistema, num contexto onde estruturas denominadas atores, representam as entidades externas que interagem com o sistema modelado.

Para a **representação estática** é utilizado o **diagrama de classes**, que consiste nas classes presentes no domínio do sistema, que representam tudo o que é gerenciado pela aplicação modelada e seus respectivos relacionamentos. Não será especificado o **diagrama de objetos**, que retrata o perfil do sistema em um determinado momento de sua execução, com os respectivos objetos instanciados a partir das classes juntamente com seus respectivos estados.

A **representação dinâmica** é feita através do **diagrama de seqüência**, que aborda a colaboração dinâmica entre vários objetos de um sistema. O aspecto mais importante deste diagrama é que a partir dele percebe-se a seqüência de mensagens enviadas entre os objetos, ou seja, a interação entre os objetos em um ponto específico da execução do sistema. Não serão especificados os diagramas de:

- **estado,** que é um complemento para as descrições das classes, mostrando todos os estados possíveis que objetos de uma certa classe podem se encontrar, e os respectivos eventos que resultam nestas alterações;

- **colaboração**, que mostra de forma semelhante ao diagrama de seqüência, a colaboração dinâmica entre os objetos, porém exibindo além da troca de mensagens, os relacionamentos entre os objetos;

- **atividade**, cuja finalidade é capturar ações (trabalho e atividades que serão executados) na implementação de uma operação (método), e seus resultados em relação as mudanças de estado numa instância de um objeto.

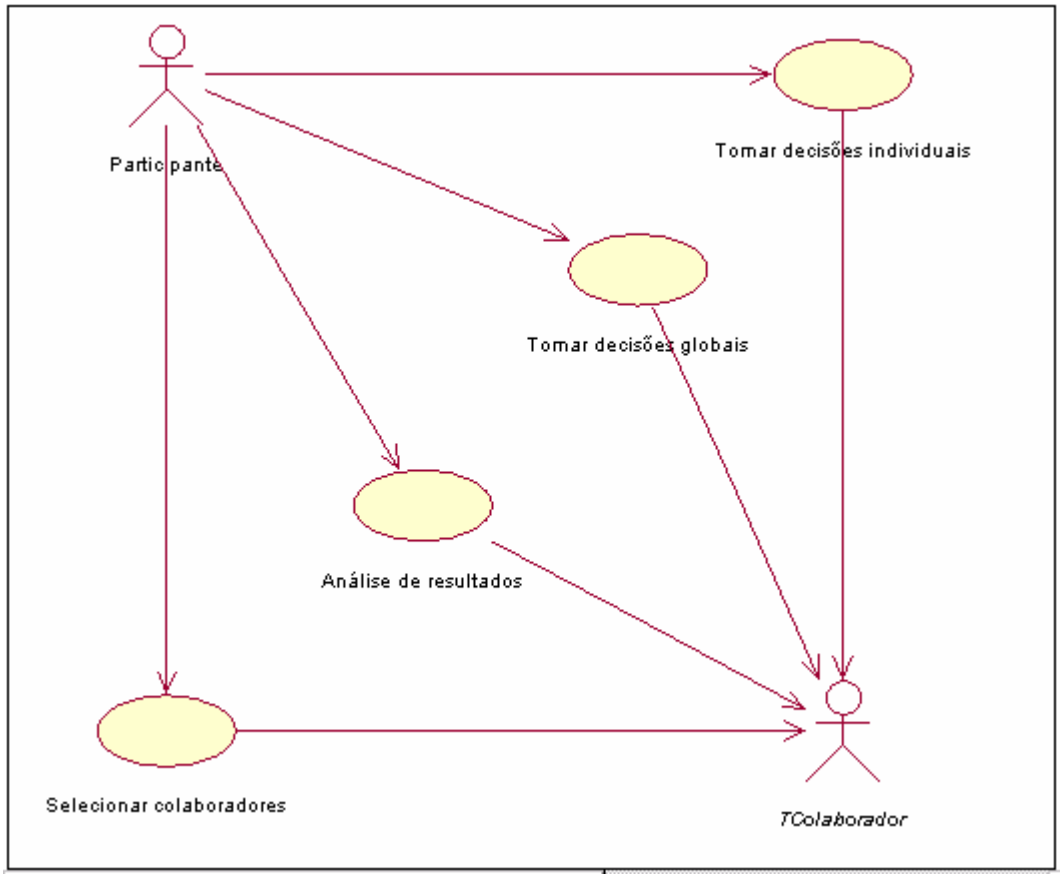
4.2. Modelagem do Sistema

Este tópico expõe uma visão conceitual do sistema proposto, através dos diagramas disponibilizados pela metodologia de desenvolvimento adotada, e suas devidas complementações textuais.

4.2.1. Diagramas de Use-Case

Na figura 8 será mostrado o diagrama de casos de uso do sistema, seguido de uma breve explanação sobre cada caso de uso, provendo um entendimento conceitual do processo representado no diagrama. Um entendimento completo será disponibilizado através dos diagramas de seqüência no decorrer do capítulo.

Figura 8 - Diagrama de Use-case



Os **casos de uso** do sistema são:

a) **Selecionar Colaboradores**: momento onde ocorre a seleção da equipe de colaboradores que irão compor a empresa. Na versão anterior do Líder, o animador selecionava o quadro de colaboradores e suas respectivas locações hierárquicas na empresa. No sistema atual, o participante fica responsável pela seleção do seu quadro de colaboradores, e inclusive de determinar seu turno e célula de trabalho (que são inovações deste novo modelo). Após isso, os colaboradores selecionados são ativados e alocados dentro do contexto da empresa, ou seja, em suas respectivas células de trabalho.

b) **Tomar Decisões Individuais**: momento onde o participante toma decisões individuais sobre um determinado colaborador da empresa. Após isso, o colaborador que sofreu a ação inicia um autoprocessamento para atualizar seu estado, e conseqüentemente o estado da empresa;

c) **Tomar Decisões Globais**: momento onde o participante toma decisões que afetam todos os colaboradores da empresa, caracterizando também o fim de um

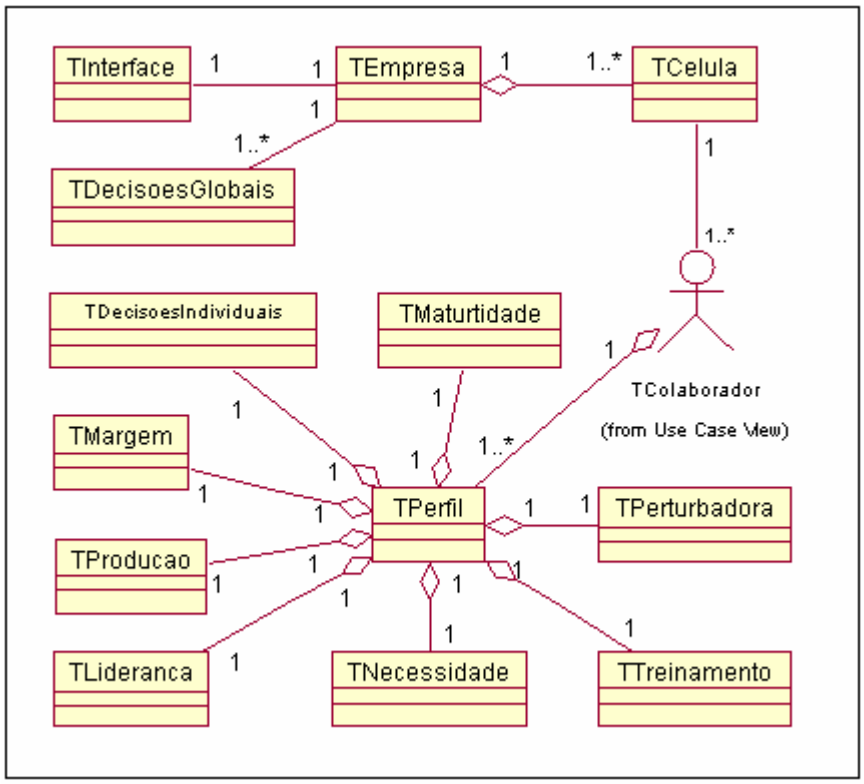
período da aplicação. Após isso, todos os colaboradores da empresa iniciam um autoprocessamento para atualizarem seus respectivos estados, e conseqüentemente o estado da empresa;

d) **Analisar Resultados**: disponível aos participantes, este caso de uso coleta informações de cada colaborador da empresa, formata estas informações de uma maneira a facilitar uma análise dos perfis de cada colaborador e conseqüentemente da situação da empresa como um todo, servindo de base para as próximas decisões.

4.2.2. Diagrama de Classes

No diagrama apresentado na figura 9, tem-se a visualização das classes que compõem o domínio do sistema, seus respectivos inter-relacionamentos cardinalmente ilustrados.

Figura 9 - Diagrama de Classes



Na seqüência tem-se uma breve descrição do papel de cada classe do diagrama no contexto global do sistema:

a) **Classe Interface:** responsável pela interação entre o sistema com os usuários do mesmo;

b) **Classe Empresa:** representação da empresa, onde estão as células de trabalho que contêm os respectivos colaboradores;

c) **Classe Decisões Globais:** componente da classe Empresa, contendo todas as decisões destinadas a empresa, em um determinado período;

d) **Classe Célula:** ilustra a célula de trabalho, que detém o controle da lista de colaboradores pertencentes à mesma;

e) **Classe Colaborador:** representa cada colaborador pertencente à empresa, mantendo um histórico do estado interno do colaborador, que se altera a cada período processado;

f) **Classe Perfil:** parte da classe Colaborador, que concentra cálculos e atributos que compõem o processamento das decisões do simulador;

g) **Classe Decisões Individuais:** componente da classe Perfil, contendo todas as decisões individuais destinadas a este colaborador, em um determinado período;

h) **Classe Liderança:** componente da classe Perfil, responsável pelos cálculos relativos à eficácia na liderança do colaborador;

i) **Classe Necessidade:** componente da classe Perfil, responsável pelos cálculos relativos às necessidades dos colaboradores;

j) **Classe Perturbadora:** componente da classe Perfil, responsável pelos cálculos relativos à identificação de situações perturbadoras no comportamento do colaborador;

k) **Classe Treinamento:** componente da classe Perfil, responsável pelos cálculos relativos à definição dos efeitos de determinados treinamentos a determinados colaboradores;

l) **Classe Produção:** componente da classe Perfil, responsável pelos cálculos relativos à definição da produtividade do colaborador;

m) **Classe Maturidade:** componente da classe Perfil, responsável pelos cálculos relativos à maturidade dos colaboradores;

n) **Classe Margem:** componente da classe Perfil, responsável pelos cálculos relativos à definição das receitas e despesas geradas pelo colaborador.

O detalhamento de cada classe, ou seja, seus métodos e atributos correspondentes, são disponibilizados no Anexo I.

4.2.3. Diagramas de Seqüência

Cada caso de uso descrito no tópico supra-enumerado, será graficamente demonstrado com seu respectivo diagrama de seqüência. Os diagramas de seqüência são disponibilizados como anexo. Abaixo uma lista dos diagramas e seu respectivo anexo:

a) Selecionar colaboradores – ANEXO II;

b) Tomada de decisões individuais – ANEXO III;

c) Tomada de decisões globais – ANEXO IV;

d) Análise dos resultados – ANEXO V.

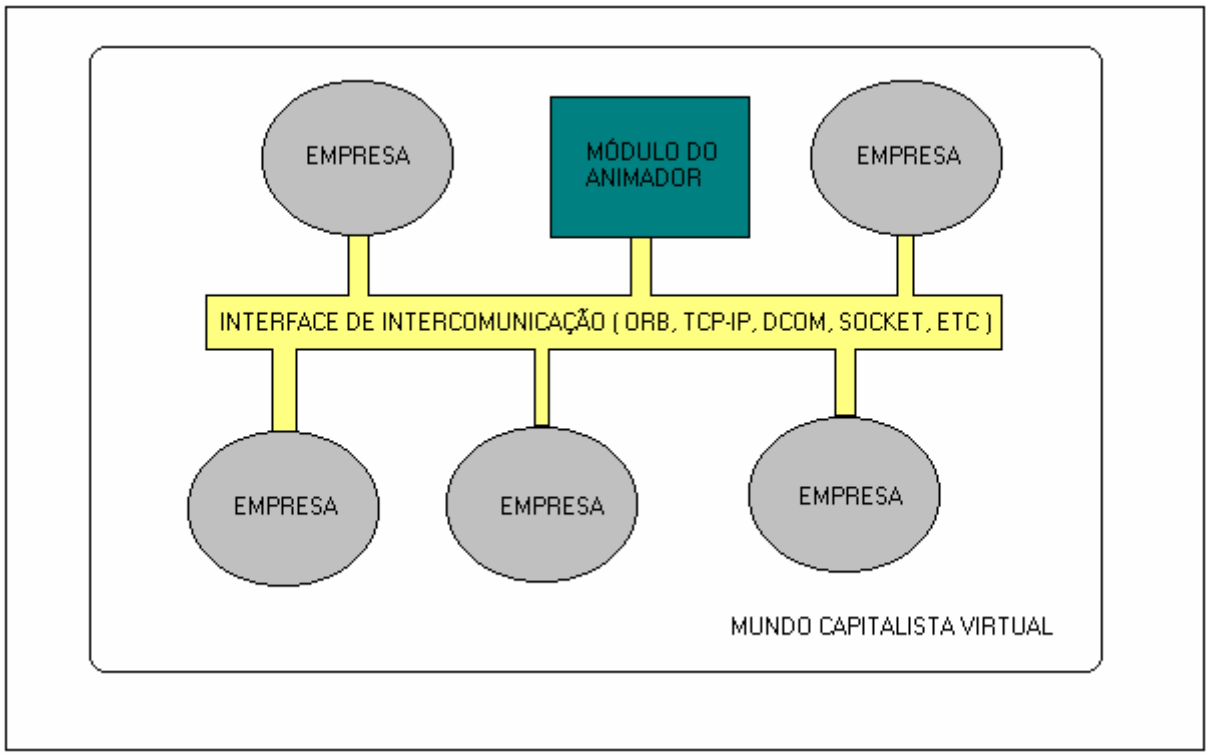
4.3. O Líder baseado em agentes

Neste tópico será exposto o novo contexto do simulador de empresas Líder após sua reestruturação como sistema baseado em agentes.

4.3.1. Contextualização

O sistema desenvolvido é o principal componente para um projeto que envolve o Líder em uma simulação mais próxima possível do modelo real. Este modelo este que envolve contextos multi-empresariais concorrentes, que são fundamentais para aprimorar o simulador de empresas Líder, como ferramenta de treinamento em gestão de recursos humanos. (figura 10).

Figura 10 - Contextualização do sistema proposto



O modelo proposto neste trabalho visa o gerenciamento de uma única empresa, ou seja, no computador onde o sistema está rodando, existe uma empresa dentro de um mundo virtual formado por diversos outros computadores que também executam o sistema (figura 10).

No entanto, se imaginar uma sala de aula ou laboratório onde existam equipes de participantes dispostos em máquinas separadas, porém participando de uma mesma aplicação do simulador, pode-se ter, então, empresas em um mesmo contexto geográfico, concorrentes, e com a possibilidade de intercomunicação.

Indo um pouco mais além, a possibilidade de empresas terem acesso a determinadas informações umas das outras, pode-se fazer com que um colaborador da empresa Y descobrisse, por exemplo, que na empresa X, o salário de um outro colaborador que executa a mesma função que ele, ganha mais, isto tendo um efeito direto sobre sua motivação.

Um outro fator importante é a possibilidade de criação de um módulo separado para o animador, onde este teria acesso a informações estratégicas sobre cada empresa, viabilizando uma avaliação permanente das equipes no andamento da simulação.

Das diversas tecnologias existentes para a comunicação entre computadores ligados em redes, e para comunicação entre objetos distribuídos como CORBA (*Common Object Request Broker Architecture*) e DCOM (*Distributed Component Object Model*) são disponibilizadas no ambiente no qual o protótipo foi implementado, Delphi versão 5.0, o que torna ainda mais simples a implementação deste projeto.

Assim, cumpre ressaltar que neste trabalho será implementado o modelo para o gerenciamento de uma empresa.

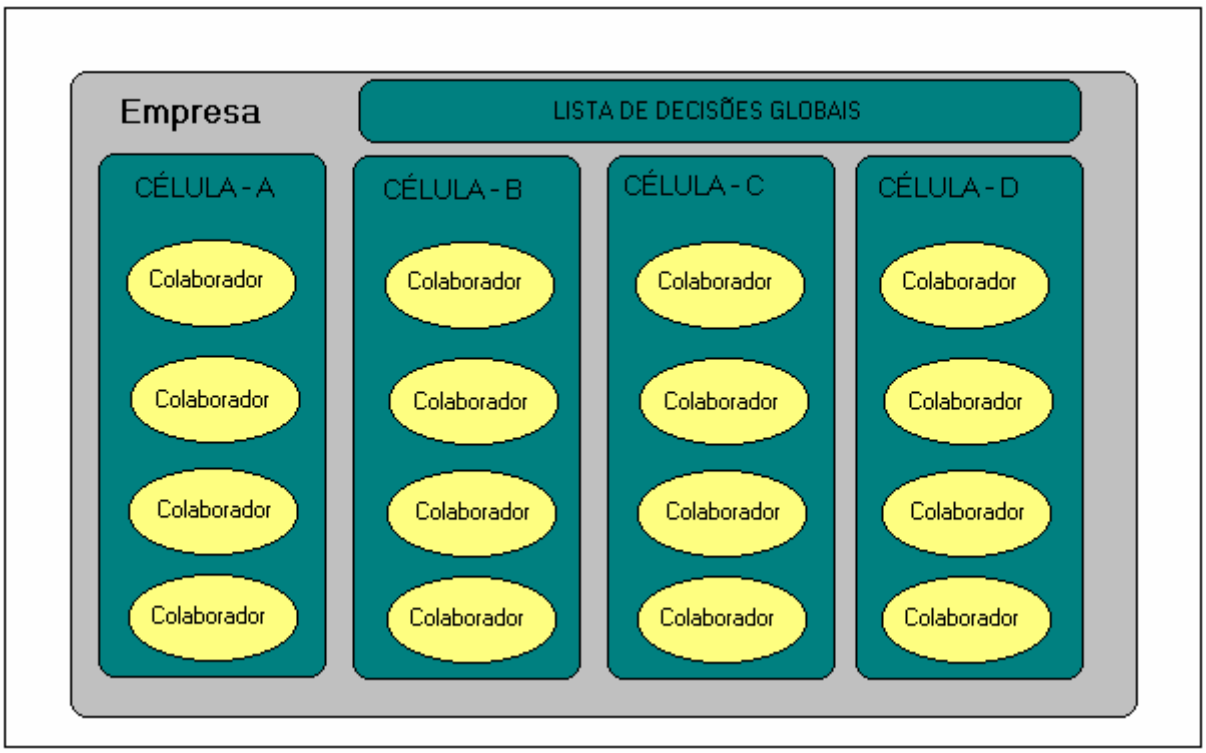
4.3.2. O sistema proposto

O sistema é implementado em duas partes: interface e aplicação. Toda interação entre o usuário e o sistema é feita através de uma interface de operação, que impede a amarração do código implementado relativo a aplicação, com detalhamentos de código particulares do ambiente de desenvolvimento utilizado. Como o sistema é composto de objetos, a comunicação entre as partes (interface e aplicação) é feita através de serviços disponíveis em ambas as partes conforme diagramas dispostos anteriormente no capítulo de especificação do sistema.

Na figura 11, está a estrutura da classe empresa, onde ficam encapsuladas todas as outras estruturas que compõem o sistema, como a lista de células de trabalho, contendo cada uma sua lista dinâmica de colaboradores.

As células de trabalho foram implementadas como listas dinâmicas apesar de terem um número fixo de quatro células (A, B, C e D), prevendo possíveis alterações na estrutura atual do simulador. Cada célula pode conter oito colaboradores por turno de trabalho, ou seja, são três turnos (matutino, vespertino e noturno) com oito colaboradores por turno, totalizando 24 colaboradores por célula de trabalho, multiplicados por quatro células, resultam em 96 colaboradores no máximo.

Figura 11 - Diagrama da estrutura do protótipo



4.3.2.1 Os agentes colaboradores

O protótipo pode ser visto como um conjunto de colaboradores (agentes) dispostos de uma forma organizada (sociedade), para um gerenciamento inteligente de seus respectivos estados encapsulados, modelando uma estrutura comportamental humana baseada no modelo matemático do Líder. Ou seja, cada colaborador possui uma autonomia de processamento de seu estado interno, o qual é disponibilizado aos seus companheiros através de mensagens (métodos), que constituem basicamente em uma linguagem de comunicação.

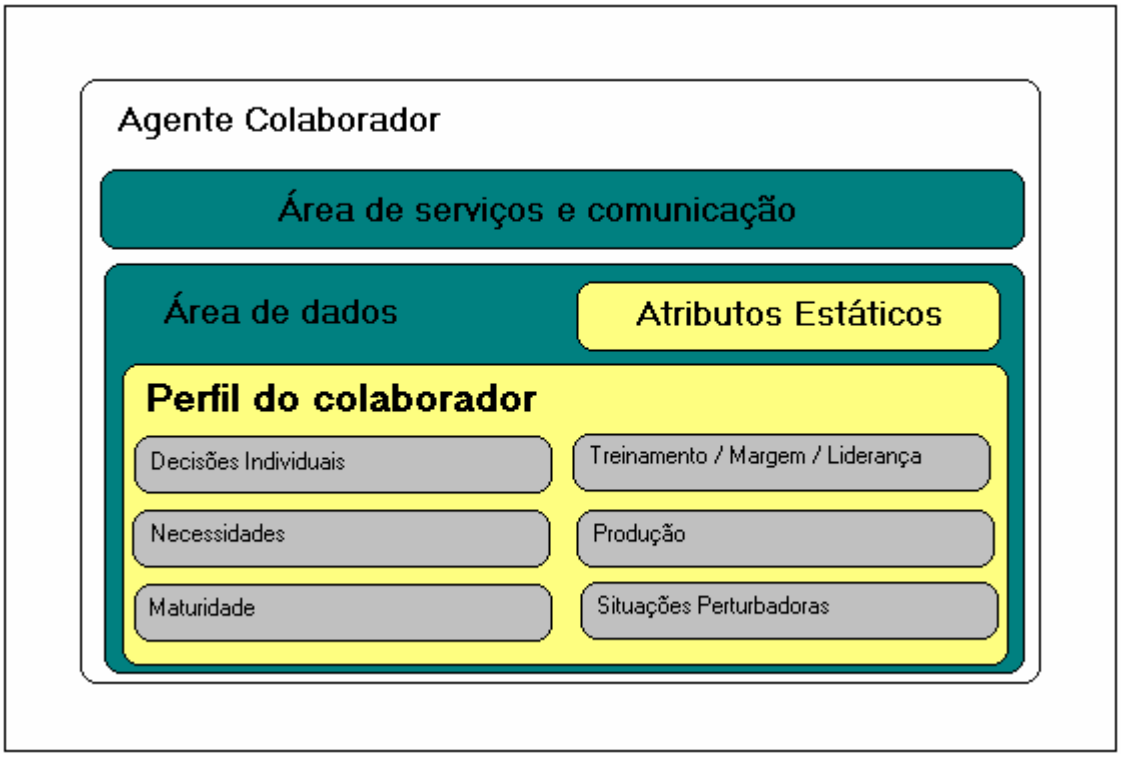
Estes agentes não possuem nenhuma estrutura de gerenciamento de acesso exclusivo a recursos, nem de sincronismo de processos, pois o sistema foi modelado para prover uma autonomia aos agentes, e evitar que um agente altere o estado interno de outro diretamente, limitando o acesso apenas a consulta de informações. Apenas o próprio agente altera seu estado interno, baseado em sua inteligência quando do processamento das decisões dos participantes.

A estrutura dos agentes pode ser vista em duas camadas:

a) área de serviços e comunicação, onde ficam os métodos privados e públicos utilizados na comunicação entre o agente e a interface e entre os próprios agentes.

b) área de dados, que se subdivide em outras duas: estática e dinâmica: na parte estática, ficam os atributos que não se alteram durante a existência do agente. São atributos definidos durante sua criação, e geralmente identificadores exclusivos (nome, idade, etc). Na parte dinâmica, estão os atributos que se alteram, decorrentes dos efeitos das decisões dos participantes (maturidade, salário, produção, etc), a figura 12 ilustra a estrutura descrita acima.

Figura 12 - Estrutura dos agentes



Os agentes mantêm seu estado interno em memória até o encerramento do sistema, ou sua eliminação do sistema decorrente de uma demissão.

4.3.3. O Protótipo

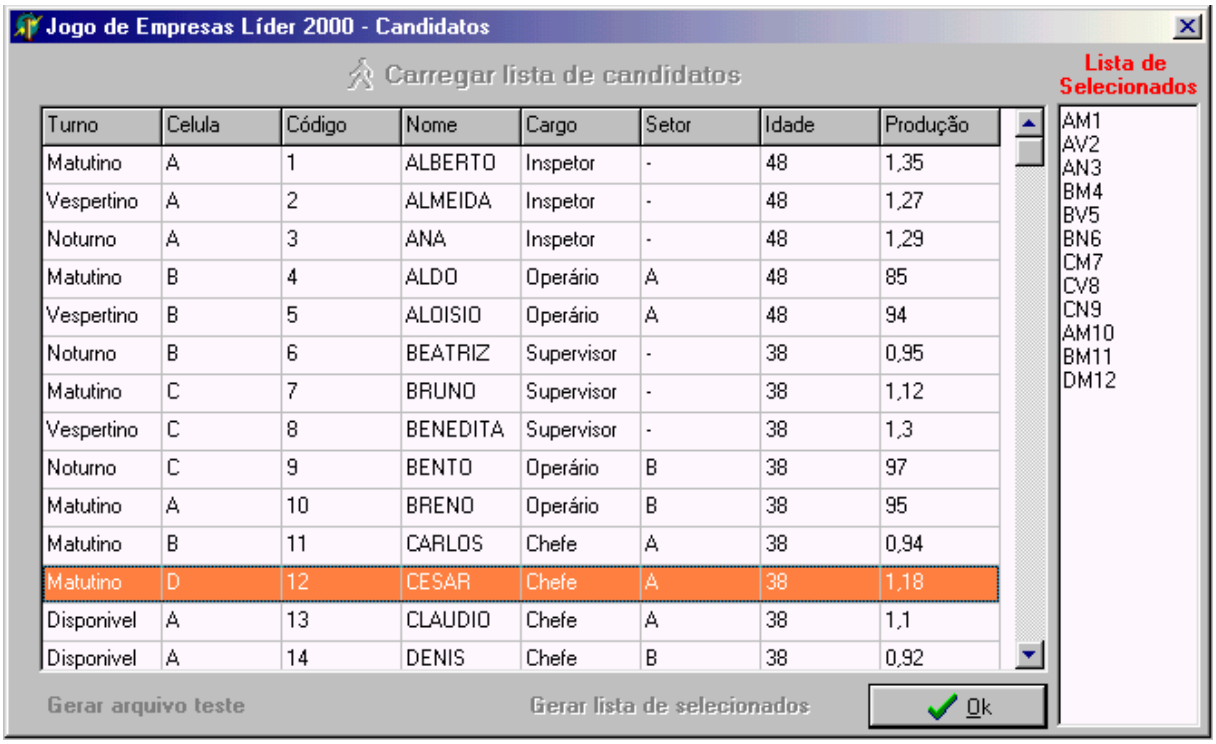
Este item apresenta a visualização do protótipo desenvolvido, com uma seqüência coerente de utilização do mesmo passo a passo, apresentando, ainda, detalhes da implementação. A apresentação será feita por casos de uso, seguindo a modelagem descrita anteriormente.

4.3.3.1 Selecionar colaboradores

A figura 13 contempla a tela seleção de candidatos, onde são definidos quais, entre os colaboradores disponibilizados previamente pelo sistema, irão compor o quadro de funcionários da empresa.

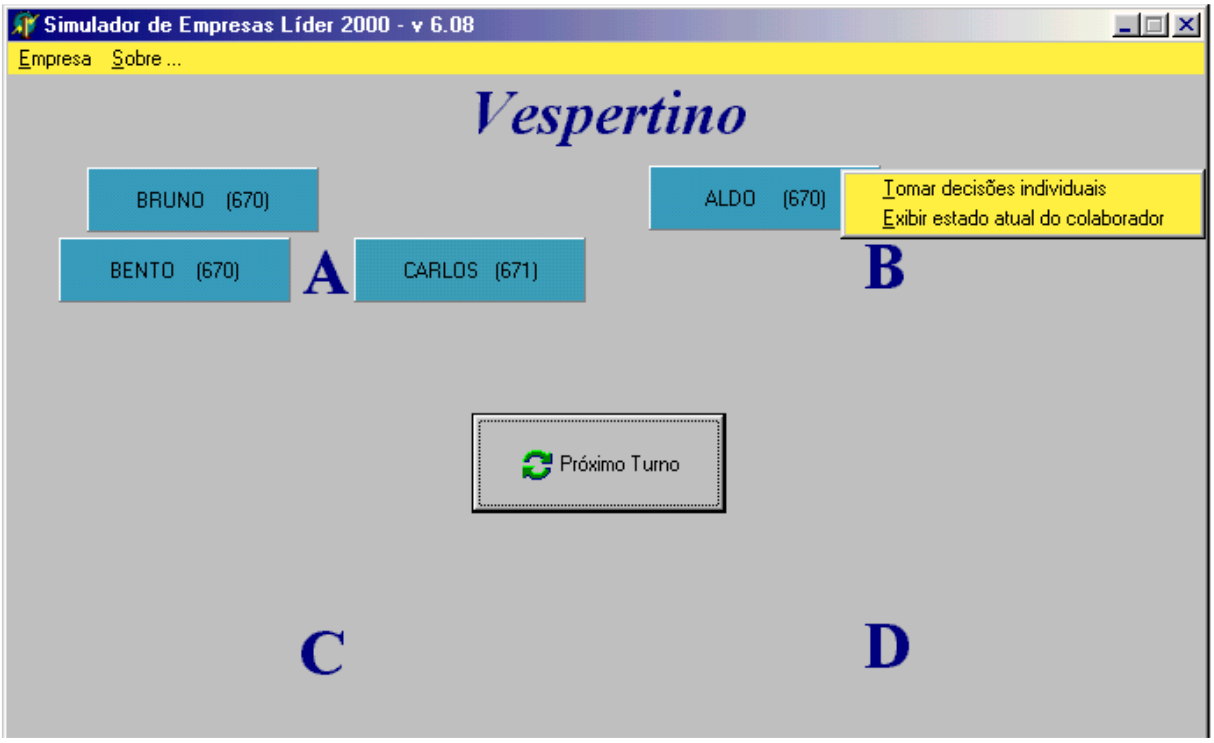
Esta etapa é que define em qual célula e turno o colaborador será alocado. Caso o turno permaneça como “disponível”, automaticamente o candidato permanecerá fora do quadro.

Figura 13 - Tela de seleção de candidatos do protótipo



Após devidamente selecionados, os colaboradores escolhidos são instanciados em suas respectivas células de trabalho, e os designados para o turno matutino, automaticamente são exibidos na tela de interface, devidamente dispostos em suas células de trabalho. (figura 14).

Figura 14 - Tela principal do protótipo



No nível de código, o objeto empresa é instanciado após a seleção do quadro, onde a lista de selecionados é enviada como parâmetro (quadro 1). O retorno do método *GetListaSelecionados*, é uma estrutura pré-definida do Delphi do tipo *TStringList*, e o objeto empresa, uma instância da Classe *TEmpresa*, especificada anteriormente. *F_Candidatos*, é uma tela componente da camada de interface.

Quadro 1 – Criação da empresa

```
procedure TF_Interface.Criar1Click(Sender: TObject);
begin
  try
    Criar1.Enabled := false;  //tratamento de interface
    F_Candidatos := TF_Candidatos.create(self); //Criação do Form
    F_Candidatos.ShowModal; //Ativação do Form
    //verificando confirmação do usuário
    if F_Candidatos.ModalResult = mryes then
      //Criação da empresa
      Empresa := TEmpresa.Create(F_Candidatos.GetListaSelecionados)
    else
      Criar1.Enabled := true; //tratamento de interface
   finally
     F_Candidatos.free; // liberação do form alocado
   end;
end;
```

Depois de enviada como parâmetro para o procedimento *TEmpresa.Create*, a lista de colaboradores selecionados é analisada, e são instanciadas as células de trabalho necessárias com seus respectivos colaboradores. (quadro 2).

Quadro 2 - Definição das células de trabalho

```
constructor TEmpresa.Create(Selecionados : TStringList);
var
  Ind, cont           : Byte;
  Celula              : TCelula;
  String_Atual, temp  : String;
  TURNO, Celula_Atual : CHAR;
  Colaborador         : Integer;
  achou               : boolean;
begin
  Self.TurnoAtual        := 'N';
  Self.NUMERO_EMPREGADOS := 0;
  Celulas := TList.Create; //instanciar lista de celulas !!!
  if selecionados <> nil then
  begin
    For cont := 0 to Selecionados.Count - 1  do
    begin
      //analizar o string CELULA+TURNO+COLABORADOR ex-AN12
      String_Atual := Selecionados.Strings[cont];
      Celula_Atual  := String_Atual[1]; //pegar o ID da célula
      //VERIFICAR SE ESTA CELULA JÁ EXISTE NA LISTA
      ind := 0;
      achou := false;
      while (ind < Celulas.count) and (not achou) do
      begin
        if TCelula(Self.Celulas[ind]).GetID = Celula_Atual then
          achou := true
        else
          ind := ind + 1;
      end;
      Temp := copy(String_Atual,2,1);
      TURNO :=  Temp[1]; //extrair o TURNO do colaborador (M,V,N)
      //separar o codigo do colaborador no string
      String_Atual := copy(String_Atual,3,length(String_Atual)-1);
      Colaborador := StrToInt(String_Atual); //separar o codigo
      if Not achou then
      begin
        Celula := TCelula.Create(Celula_Atual); //instanciar célula
        Celula.Inserir_Colaborador(Colaborador,TURNO);//inserir colaborador
        Self.Celulas.Add(Celula); //adicionar celula na empresa
      end
      else
      begin
        TCelula(Self.Celulas[ind]).Inserir_Colaborador(Colaborador, TURNO);
      end;
      Self.NUMERO_EMPREGADOS := Self.NUMERO_EMPREGADOS + 1;
    end;
  end;
end;
```

Cada colaborador é uma *thread* do Delphi (quadro 3). Sendo assim, neste momento existem tantas *threads* ativas no sistema, simultaneamente, quanto o número de colaboradores designados para o turno corrente. Para evitar uma possível degradação do sistema, em nível de desempenho, os colaboradores alocados nos outros turnos, ficam suspensos até a hora do seu turno de trabalho. A ativação por turnos limita o número de threads ativas para 32 no máximo, que é o máximo de colaboradores permitido por turno.

Quadro 3 - Classe TColaborador

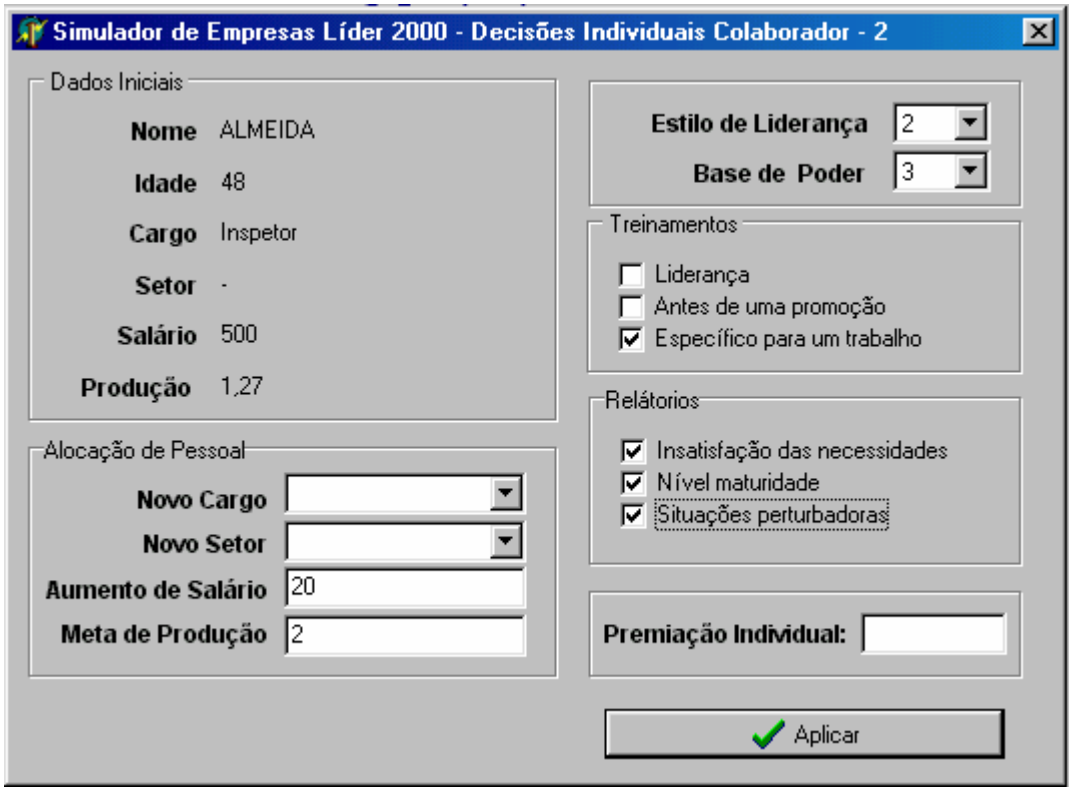
```
TColaborador = Class(TThread) // herança da classe TThread
private
  PeriodoAtual             : Integer; // período atual do colaborador
  ProcessarAgora           : Boolean; // autoriza o processamento
  AutoCheck                : Boolean; // auto-processamento
  CODIGO_COLABORADOR       : Integer; // CODIGO_FUNCIONARIO
  NOME                     : String;
  IDADE                    : Integer;
  TURNO                    : CHAR;
  CELULA                   : CHAR;
  Media                    : Real;   // de producao
  RGB_R, RGB_G, RGB_B      : Integer; // Índice de cor
  Trabalhando              : Boolean; //controle interno
  State                    : Integer; //controle interno
  Cor                      : TColor;
protected
  procedure Execute;override; //LOOP ETERNO DA THREAD
public
  Perfis                   : TList; //Lista de pefis processados
  Constructor create(suspenso : boolean;
                     Codigo_Colaborador: integer;
                     TURNO, CELULA: Char);
  Destructor Destroy;
  procedure Processar;
  procedure Show; //ativa o processo e exibe na tela
  procedure Hide; // suspende o processo;
  procedure AtualizarCapacidadeProdutiva;
  procedure AtualizarExperiencia;
  procedure Media_Func;
  function GetCodigo_Colaborador: Integer;
  function GetNOME: String;
  function GetIDADE: Integer;
  function GetTURNO : Char;
  function GetCELULA : Char;
  function getPERIODO_ATUAL : Integer;
  function GetState:Integer;
  procedure SetState(State:Integer);
  function GetCor:TCOlor;
  procedure SetCor(Cor : TCOlor);
end;
```

Inicialmente são atribuídas cores aos colaboradores, de acordo com seu perfil inicial. Se a cor está mais próxima do vermelho, indica um perfil não muito produtivo, se próxima do azul, indica uma produtividade satisfatória. As cores se alteram a cada tomada de decisões.

4.3.3.2 Tomar decisões individuais

Para tomar decisões individuais sobre os colaboradores dispostos na tela, basta pressionar o botão direito do mouse e selecionar a opção “decisões individuais” presente no menu. Imediatamente é aberta a tela de decisões individuais (figura 15), onde se encontram as opções de decisão para o colaborador selecionado.

Figura 15 - Tela de decisões individuais do protótipo



Após a tomada de decisão, a tela de decisões individuais é fechada, retomando o foco para a tela inicial, e conseqüentemente, possibilitando um acompanhamento do resultado das decisões no colaborador através de sua alteração de cor. (figura 14).

Este procedimento pode ser realizado para cada colaborador quantas vezes o participante desejar. No quadro 4, está apresentado o método *GetColaborador*, que é utilizado para localizar o colaborador dentro da empresa através de seu código, para que sejam aplicadas as decisões sobre o mesmo.

Quadro 4 - Método GetColaborador da classe empresa

```
function TEmpresa.Get_Colaborador (Codigo : Integer):TColaborador;
var
  cels, colabs : Integer;
  celula       : TCelula;
  colaborador  : TCOlaborador;
begin
  Colaborador := nil;
  //percorrer lista de células da empresa
  For Cels := 0 to celulas.Count - 1 do
  Begin
    //guardar célula atual
    celula := TCelula(self.Celulas.items[cels]);
    //percorrer lista de colaboradores da célula atual
    for colabs := 0 to  celula.GetListaColaboradores.count - 1 do
    begin
      //verificar se o colaborador é o requisitado pelo código
      if TColaborador(Celula.GetListaColaboradores.items[colabs]).
         GetCodigo_colaborador = codigo then
      Begin
        // se for o colaborador guardar na variável colaborador
        Colaborador :=
          TColaborador(Celula.GetListaColaboradores.items[colabs]);
      end;
    end;
  end;
  // mandar o colaborador como retorno da função.
  Result := Colaborador;
end;
```

A autonomia do agente colaborador permite que ele se autoprocesse a cada tomada de decisões, ou após um determinado tempo sem que o participante interaja sobre ele.

Na implementação, o agente sabe a hora de processar pois, seu serviço “processar” foi requisitado por ele mesmo ou por algum outro componente do sistema, e alterou seu atributo “ProcessarAgora” para “true”, ativando a rotina demonstrada no quadro abaixo. É importante ressaltar que a rotina do quadro 5 é apenas uma parte pertencente do método “execute” definido na classe *TColaborador*, o qual é herdado da classe *TThread* do Delphi, que tem como característica principal o fato de ser executado intermitentemente enquanto a thread está ativa.

Quadro 5 – Código fonte do processamento

```
    If Self.ProcessarAgora then  //PROCESSAR
    begin
      Self.Cor := RGB(25,254,25); //cor para indicar processamento (verde)
      //Atualizar agente na interface
      F_Interface.AtualizaAgente(Self);
      sleep(300); // Pausa de 300 milisegundos
      TPerfil(Self.Perfis[Self.PeriodoAtual]).Lideranca.Calcular;
      TPerfil(Self.Perfis[Self.PeriodoAtual]).Necessidades.Calcular;
      TPerfil(Self.Perfis[Self.PeriodoAtual]).SitPert.Calcular;
      TPerfil(Self.Perfis[Self.PeriodoAtual]).Maturidade.Calcular ;
      TPerfil(Self.Perfis[Self.PeriodoAtual]).Producao_Real :=
         TPerfil(Self.Perfis[Self.PeriodoAtual]).Producao.Calcular;
      TPerfil(Self.Perfis[Self.PeriodoAtual]).Margem.Calcular(
         TPerfil(Self.Perfis[Self.PeriodoAtual]).Producao_Real);
      //PERFIL INICIAL PARA PROXIMO PERIODO = ATUAL
      Self.Perfis.Add(TPerfil(Self.Perfis[Self.PeriodoAtual]));
      //ATUALIZAR PERIODO
      Self.PeriodoAtual := Self.PeriodoAtual + 1;
      // IMPORTANTE PROCESSAMENTO  - ENCERRAR
      Self.ProcessarAgora := False;
    end
```

O código demonstrado no quadro 6, também pertencente ao procedimento “execute” da classe *TColaborador*, demonstra como foi implementada a forma em que o agente analisa a necessidade ou não de iniciar um processamento de si mesmo, tendo como base para sua decisão, o número de interações que os outros colaboradores sofreram. Para isto, consulta um a um seus colegas de trabalho, perguntando quantas interações cada um já sofreu até o momento.

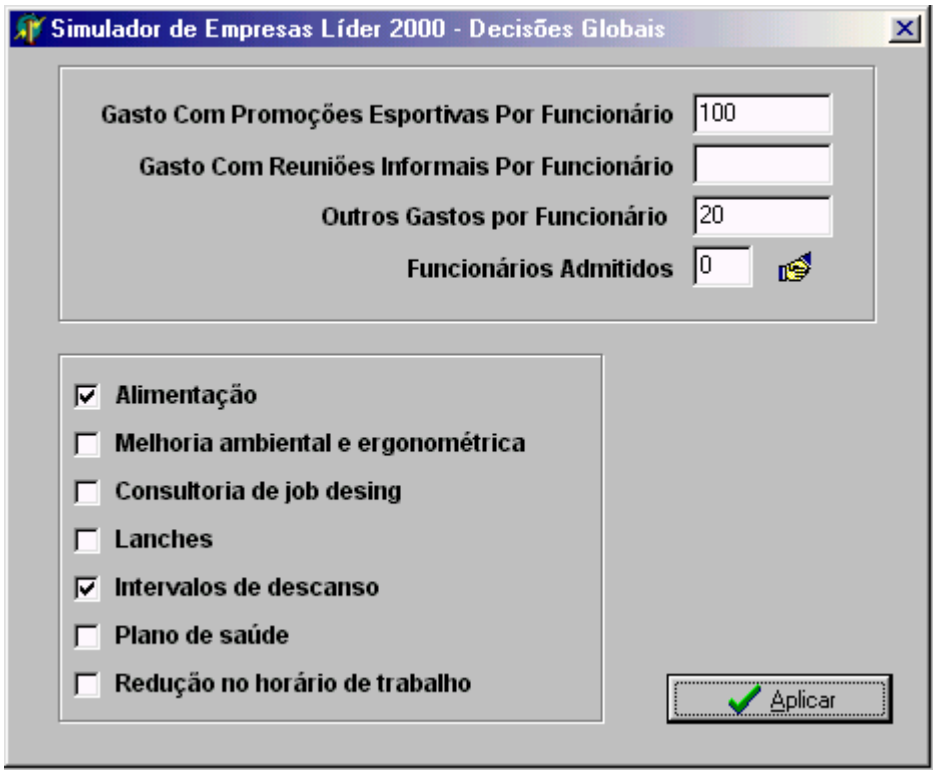
Quadro 6 - Verificação do processamento automático

```
if Self.AutoCheck then
begin
  Self.AutoCheck := false;
  Menor_Periodo_empresa := 500; //VALOR DE INICIALIZAÇÃO
  List_Cel := TCelula(F_Interface.Empresa.GetListaCelulas);
  For Celulas := 0 to List_Cel.Count - 1 do
  begin //PERCORRER TODAS AS CELULAS DA EMPRESA
    List_Col := TCelula(List_Cel.Items[Celulas]).GetListaColaboradores;
    For Agentes := 0 to List_Col.Count - 1 do
    begin //PERCORRER TODOS OS AGENTES DA CELULA
      Col_Atual := TColaborador(List_Col.Items[Agentes]);
      if Col_Atual.GetPeriodo_Atual < Menor_periodo_empresa then
        if Col_Atual.GetCodigo_colaborador <> Self.Codigo_colaborador then
          Menor_periodo_empresa  := Col_Atual.GetPeriodo_Atual;
    end;
  end; //COMPARAÇÃO COM OS OUTROS AGENTES EM Nro DE INTERAÇÕES
  if Menor_periodo_empresa > Self.PeriodoAtual + 3 then
    self.ProcessarAgora := true; //SE NECESSÁRIO AUTO-PROCESSAR
  end;
```

4.3.3.3 Tomar decisões globais

Para a tomada de decisões globais, clica-se com o botão direito do mouse sobre qualquer parte da tela, onde não esteja posicionado um colaborador. A partir daí será exibida a tela de decisões globais (figura 16).

Figura 16 - Tela de decisões globais do protótipo



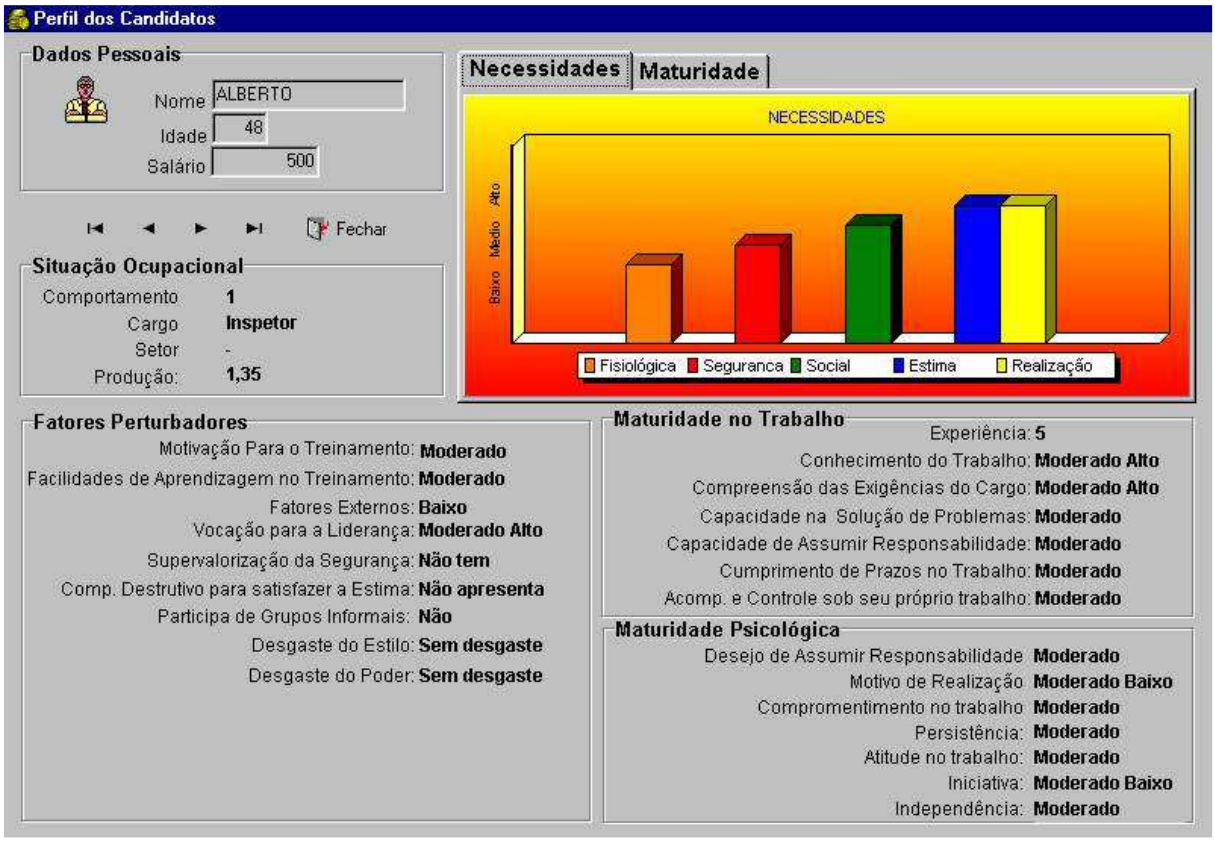
Nesta tela, são apresentadas as opções de decisões que podem ser tomadas, com efeito global, ou seja, tem impacto sobre todos os colaboradores da empresa, inclusive os que não estão aparecendo na tela e que façam parte de outro turno de trabalho.

Após este procedimento, todos os colaboradores (agentes) se autoprocessarão simultaneamente, podendo alterar seu estado e conseqüentemente sua cor.

4.3.3.4 Análise dos resultados

Nesta tela (figura 17), há a possibilidade de se visualizar o perfil completo do colaborador desejado. Para ativá-la, basta pressionar o botão direito do mouse e selecionar a opção “Perfil atual” presente no menu.

Figura 17 - Tela de análise de resultados individual



5. Conclusões e Sugestões

Este capítulo apresenta as conclusões, limitações e sugestões referentes ao trabalho desenvolvido.

5.1. Conclusões

Os objetivos do trabalho foram alcançados, destacando-se que a nova modelagem adaptada ao contexto de agentes não alterou nenhum resultado dos cálculos presentes no modelo original do Líder, mantendo assim a integridade do processamento como nas versões anteriores. A nova forma de comunicação entre os colaboradores, através de troca direta de mensagens, substituiu o processo anterior de acesso a um meio comum (banco de dados), tornando o sistema mais ágil.

A utilização de agentes permitiu um modelo para o sistema mais próximo do real. A nova interface baseada neste modelo, permitiu uma maior interatividade entre os participantes e a evolução dos colaboradores, os quais puderam acompanhar os efeitos de suas decisões sobre os mesmos.

Quanto às ferramentas utilizadas, a Rational Rose 2000, na modelagem do sistema, se mostrou eficiente e bastante completa no que diz respeito ao suporte a metodologia adotada (UML). Com relação ao ambiente Delphi, o mesmo apresentou estabilidade na implementação de threads, tornando sua utilização para este tipo de aplicação, satisfatória.

5.2. Limitações

Como limitações:

a) a persistência dos objetos foi implementada com arquivos binários, o que não é coerente à metodologia orientada a objetos;
b) não estão presentes todos os relatórios disponíveis para a análise dos resultados do participante.

5.3. Sugestões

Como sugestões destacam-se:

a) estudo de uma forma para a implementação da persistência dos agentes, no qual a utilização de bancos de dados orientados a objetos, ou tecnologia compatível com armazenamento de objetos em disco, ou ainda um mapeamento de como armazenar objetos em estruturas relacionais;

b) intercomunicação entre várias instâncias do sistema através de redes de computadores, simulando uma empresa e filiais, ou concorrentes;

c) relatórios dirigidos aos participantes que incluam históricos decisórios, e evolutivos dos colaboradores, aproveitando que neste novo modelo implementado, tanto os perfis, quanto as decisões individuais ficam armazenados dentro do agente como seu estado interno, funcionando como um histórico da simulação.

.

Anexo I

Anexo II

Anexo III

Anexo IV

Anexo V

Referências Bibliográficas

[CAN1998] CANTU, Marco. **Dominando o Delphi 3: a bíblia**. São Paulo: Makron Books, 1998.

[DAV1995] DAVIS, Harold. **Delphi ferramentas poderosas**. São Paulo: Berkeley Brasil Editora, 1995.

[FRA1997] FRANKLIN, Stan, GRAESSER, Art (1996) "Is it an agent, or just a Program? A Taxonomy for Autonomous Agents", *Proceedings of the Third International Workshop on Agent Theories, Architetures, and Languages*, Springer-Verlag, http://www.msci.memphis.edu/~franklin/AgentProg.html, em 29/10/97.

[FUR1998] FURLAN, José Davi. **Modelagem de objetos através da UML – The unified modeling language**. São Paulo: Makron Books, 1998.

[HÜB1995] HÜBNER, Jomi Fred. **Migração em sistemas multiagentes abertos**. Porto Alegre: Universidade Federal do Rio Grande do Sul, 1995. Dissertação de Mestrado.

[HÜB1996] HÜBNER, Jomi Fred. **Desenvolvimento de um ambiente de bases de conhecimento distribuídas**. Blumenau, 1996. Relatório final projeto PIBIC/CNPq. Universidade Regional de Blumenau.

[JEN1994] JENNINGS, Nicholas R. (1994) **"THE ARCHON SYSTEM AND ITS APPLICATIONS"**, *Second International Working Conference on Cooperating Knowledge Based Systems (CKBS-94) (Invited Paper),* Keele, UK, pp.241-257, ftp://ftp.elec.qmw.ac.uk/pub/isag/distributed-ai/publications, em 29/10/97.

[JEN1997] JENNINGS, Nicholas R., WOOLDRIDGE, Michael (1994) "Intelligent Agents: Theory and Practice", http://www.doc.mmu.ac.uk/STAFF/mike/papers.html, em 30/09/97.

[JEN1998] JENNINGS, Nicholas R., WOOLDRIDGE, Michael. **Agent technology foundations, applications and markets**. New York: Springer, 1998.

[LOP1994] LOPES, Maurício Capobianco. **Jogo de empresas LÍDER: aperfeiçoamento do modelo e do sistema**. Florianópolis, 1994. Dissertação de Mestrado em Engenharia de Produção. Universidade Federal de Santa Catarina.

[MAE1995] MAES, Pattie. "**Artificial Life Meets Entertainment: Life Like Autonomous Agents**", Communications of the ACM, 1995.

[MAL1990] MALDONADO, Luis Alberto Taja. **Implantação em micro computador de um modelo comportamental para treinamento de liderança empresarial: um enfoque da liderança situacional**. Florianópolis, 1990. Dissertação de Mestrado em Engenharia de Produção. Universidade Federal de Santa Catarina.

[MAT1996] MATCHO, Jonathan. Usando Delphi 2. Rio de Janeiro : Campus, 1996.

[NIV1998] NIVEIROS, Sofia Inês. **Estudo e aperfeiçoamento do modelo das maturidades dos funcionários no jogo de empresas líder**. Florianópolis, 1998. Dissertação. Programa de Pós-Graduação em Engenharia de Produção. Universidade Federal de Santa Catarina.

[RUS1995] RUSSELL, Stuart e NORVIG, Peter. **Artificial intelligence: a modern approach.** New Jersey : Prentice Hall Series in Artificial Intelligence, 1995.

[SAL1990] SALVATIERRA, Edwin Gery Maldonado. **Implantação em micro computador de um modelo comportamental para treinamento de liderança empresarial: um enfoque nas necessidades humanas**. Florianópolis, 1990. Dissertação de Mestrado em Engenharia de Produção. Universidade Federal de Santa Catarina.

[SMI1994] SMITH, D. C. CIFRA e J. SPOHER. "**KidSim: Programming Agents Without a Programming Language**", Communications of the ACM, 1994.