O *Design* Centrado no Aprendiz no Sistema Jonas: uma Experiência de Desenvolvimento de um Sistema para Formação na Empresa

Marcos Augusto Francisco Borges

Dissertação de Mestrado

O *Design* Centrado no Aprendiz no Sistema Jonas: uma Experiência de Desenvolvimento de um Sistema para Formação na Empresa

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Marcos Augusto Francisco Borges e aprovada pela Banca Examinadora.

Campinas, 3 de outubro de 1997

Profª. Drª. Maria Cecília Calani Baranauskas
(Orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Instituto de Computação

Universidade Estadual de Campinas

O *Design* Centrado no Aprendiz no Sistema Jonas: uma Experiência de Desenvolvimento de um Sistema para Formação na Empresa

Marcos Augusto Francisco Borges

Setembro de 1997

Banca Examinadora:

- Profª. Drª. Maria Cecília Calani Baranauskas (Orientadora)
  Instituto de Computação - UNICAMP
- Prof. Dr. José Armando Valente
  Núcleo de Informática Aplicada à Educação - UNICAMP
- Profª. Drª. Ariadne Maria Brito Rizzoni Carvalho
  Instituto de Computação - UNICAMP
- Prof. Dr. Jacques Wainer (Suplente)
  Instituto de Computação - UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Borges, Marcos Augusto Francisco

B644d O Design centrado no aprendiz no sistema Jonas: uma experiência de desenvolvimento de um sistema para formação na empresa / Marcos Augusto Francisco Borges -- Campinas, [S.P. :s.n.], 1997.

Orientadora: Maria Cecília Calani Baranauskas

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

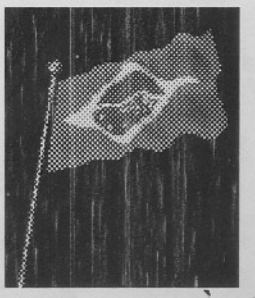
1. Sistemas especialistas (Computação). 2. Projeto de sistema. 3. Ensino auxiliado por computador. 4. Construtivismo (Educação). I. Baranauskas, Maria Cecília Calani. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Tese de Mestrado defendida e aprovada em 08 de setembro de 1997 pela Banca Examinadora composta pelos Professores Doutores

Prof. Dr. José Armando Valente

Profa. Dra. Ariadne Maria Brito Rizzoni Carvalho

Profa. Dra. Maria Cecília Calani Baranauskas



À vida, o maior de todos os bens e
ao Brasil, o melhor país do mundo.

Agradecimentos:

À minha orientadora M. Cecília C. Baranauskas, que me iniciou na ciência e que, apesar de ter todos os motivos possíveis, nunca deixou de ser uma orientadora presente, crítica e amiga.

À minha Marisa, por todo carinho, apoio e paciência.

À minha família, que me deu oportunidades de chegar até aqui.

Ao grupo do Nied, pelo apoio técnico e metodológico.

Aos professores e colegas do Instituto de Computação-Unicamp.

À Harrison Thermal Systems, pela oportunidade de efetuar pesquisas dentro de sua fábrica.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq e a Fundação de Amparo ao Ensino e Pesquisa - FAEP/UNICAMP e ao Instituto de Computação pelo apoio financeiro.

"Quem sabe que o tempo está fugindo descobre, subitamente, a beleza única do momento que nunca mais será"
Rubem Alves, Tempus Fugit, pag.11, 3.ed., Paulus, 1990, São Paulo-SP, Brasil.

Sumário

Resumo

O mundo moderno exige das pessoas muito mais conhecimento. Por este motivo, a educação vem sendo repensada, seja nos seus métodos, seja nas ferramentas utilizadas. No ambiente industrial, esse novo nível de formação é ainda mais necessário. As empresas precisam de funcionários cada vez mais preparados para enfrentar a crescente competição.

Neste trabalho discute-se a construção de um sistema especialista (SE) para uso na formação de pessoal em empresas: o **Jonas** (Borges e Baranauskas, 1996, 1997A, 1997B). Este sistema faz parte do ambiente **Enxuto** (Borges et al., 1995; Baranauskas et al., 1997) e objetiva auxiliar funcionários de uma fábrica a aprender conceitos de manufatura a partir de um processo de modelagem e simulação de uma linha de produção.

O sistema **Jonas** foi inspirado no personagem *Jonah*, do livro A Meta (Goldratt e Cox, 1994), um consultor que indica aos trabalhadores de uma fábrica como conseguir chegar a respostas para suas próprias indagações. Dentro dessa perspectiva, o sistema busca ser uma ferramenta para ensino que funciona segundo o paradigma construcionista (Valente, 1993A).

Como um SE, **Jonas** visa utilizar sua "inteligência" de um modo diferenciado, não imitando um ser humano, mas apenas buscando aumentar o potencial dos usuários. Deste modo, o sistema explora as características únicas da mídia computacional, de acordo com o paradigma do aumento de inteligência (Fischer, 1995).

Teve-se como objetivo construir um sistema com uma interface simples, que possa ser utilizada de um modo independente por pessoas com pouco conhecimento em informática. Optou-se por uma interface de manipulação direta que, segundo a teoria da ação, torna o funcionamento do sistema mais próximo dos modelos mentais dos usuários. Para se construir uma interface com estas características, o *design* do **Jonas** foi centrado no usuário (Norman, 1986; Norman e Draper, 1986).

O trabalho descreve o processo de *design* do sistema, discute as opções efetuadas, os ambientes de implementação e implantação e a metodologia de desenvolvimento. Ao final o protótipo resultante é apresentado e são feitas propostas de continuidade para o projeto.

Abstract

The global world requires much more prepared people. In this sense, education is being rethinked regarding to its methods and tools. In the industrial environment, this new level of knowledge is even more required. Factories need better prepared workers to face an increasing competition.

In this work, the development of an expert system for industrial workers training is discussed. **Jonas** (Borges and Baranauskas, 1996, 1997A, 1997B) is part of **Enxuto** environment (Borges et al., 1995; Baranauskas et al., 1997), which aims to help shopfloor workers to learn manufacturing concepts using a production line modeling and simulation process.

Jonas system is inspired on the character *Jonah*, from "The Goal A Process of Ongoing Improvement" book (Goldratt and Cox, 1994). *Jonah* is a counselor who suggest industrial workers how to solve problems by themselves. In this way, the system intends to be a learning tool supported by the constructionist paradigm (Valente, 1993A).

Jonas is an expert system, but uses its "intelligence" in a different way. It does not mimic a human, but intends to increase the user's potential. The system explores the computational media qualities, according to the intelligence augmentation paradigm (Fischer, 1995).

The purpose of the work is to build a system with an interface, which can be used by people with little computer knowledge, in an independent way. The system has a direct manipulation interface that makes it closer to the user mental models, according to the action theory. In order to build an interface with this characteristics, the process of **Jonas** design was user centered (Norman, 1986; Norman and Draper, 1986).

This work describes the system design process and discuss its options, the implementation environments and the development methodology. The resulting prototype is presented, as well as continuity project proposals.

Lista de abreviaturas e siglas

AI - Aumento de inteligência;

BC - Base de conhecimento;

BD - Banco de dados;

CAI - *Computer-aided instruction* (instrução assistida por computador);

DCA - *Design* centrado no aprendiz;

DCT - *Design* centrado na tecnologia;

DCU - *Design* centrado no usuário;

DLL - *Dynamic Link Library*;

GMB - General Motors do Brasil;

IA - Inteligência artificial;

ICAI - *Intelligent computer-aided instruction* (instrução assistida por computador inteligente);

ITS - *Intelligent tutoring systems* (sistemas de tutoramento inteligente);

JIT - *Just-in-time*;

Nied - Núcleo de Informática Aplicada à Educação;

ODBC - *Open data base connectivity* (conectividade de banco de dados abertos);

OMT - *Object modelling technique* (modelagem orientada a objetos);

OO - Orientado a objetos;

QT - Qualidade total;

SE - Sistema especialista;

SQL - *Structured query language* (linguagem estruturada de consulta);

Unicamp - Universidade Estadual de Campinas;

UFSC - Universidade Federal de São Carlos;

USP - Universidade de São Paulo;

WWW - *World Wide Web*.

1 - Introdução

O mundo está constantemente evoluindo, desafiando indivíduos e organizações a mudar e escolas e universidades a prepará-los para estas mudanças. Com a abertura do mercado gerada pela globalização, a crescente exigência de competitividade da indústria nacional fez com que a necessidade de se contar com uma mão-de-obra qualificada aumentasse. Para atender esta demanda, estudos vêm sendo feitos de modo a identificar como formar funcionários que atendam às novas necessidades das empresas e, em especial, como a informática pode colaborar neste desafio.

Em paralelo com estas novas necessidades sociais, e almejando supri-las, está acontecendo uma revolução: o surgimento de uma educação centrada no aprendiz e baseada em problemas, onde os estudantes aprendem de um modo construtivista. O aprendizado construtivista baseia-se na idéia de que se aprende melhor quando se necessita e se busca novos conhecimentos para resolver um problema específico. Os aprendizes desejam e precisam aplicar os novos conhecimentos em uma experiência autêntica e relevante para a vida (Nicol, 1990; Woolf, 1996). Uma informação que nunca é utilizada durante o processo de aprendizado pode ser difícil de recuperar e usar quando se necessitar dela no mundo real (Schank e Kass, 1996). Segundo estes estudos, o aprendizado deve ser um processo para a vida inteira permitindo a pessoas dedicar-se às atividades que as transformem em pensadores, trabalhadores, colaboradores e jogadores (Eden et al., 1996, p.40).

O vice-presidente dos EUA, Al Gore, prevê, para a próxima década, que os novos meios de comunicação divertirão tanto quanto informarão. Mas, mais importante, eles criarão muitos empregos enquanto educarão, promoverão a democracia e salvarão vidas (Scientific American, 1995). Como a informática pode colaborar neste sentido? Os aprendizes não adotam novas práticas sem que, claramente, haja vantagens para eles. Para isso, se requer um incentivo concreto para motivar o uso inicial do *software* (Edelson et al., 1996). Deve-se considerar que o maior problema no aprendizado não é o acesso difícil ou a má apresentação das informações, mas sim a impossibilidade do aprendiz poder fazer algo interessante com esta informação (Schank e Kass, 1996).

Este trabalho propõe uma nova estratégia de construção de sistemas para formação em empresas, que possa ser utilizada como um padrão de desenvolvimento nesta área. Como exemplo, é apresentada a construção de um sistema computacional com base nessa estratégia.

A Harrison Thermal Systems[1] é uma empresa pequena (menos de cem funcionários), nova (pouco mais de dez anos) e que desde sua criação baseia-se em técnicas modernas de produção: possui uma implementação exemplar de programas de qualidade e de produção *just-in-time*. Devido a todas estas características esta empresa investe na formação de sua mão-de-obra de forma mais ampla ao que o que ocorre normalmente: contrata funcionários mais qualificados (todos os funcionários possuem no mínimo segundo grau), e preocupa-se bastante com formação e treinamento. Por todas estas características, esta empresa mostrou ser um ambiente interessante para um projeto de cooperação com o Nied[2] , no âmbito de formação de funcionários do chão-de-fábrica. A proposta de se construir um ambiente de aprendizado baseado em computador surgiu a partir deste projeto de cooperação (Nied, 1995).

Um dos projetos de pesquisa derivados desta cooperação foi o ambiente computacional **Enxuto** (Borges et al., 1995), onde um usuário pode modelar uma linha de produção e simular seu funcionamento. O objetivo é possibilitar que, enquanto modelando e simulando, o usuário experimente soluções para melhorar os resultados da simulação e, assim, aprenda estratégias de melhoria de processos produtivos. O ambiente **Enxuto** viabiliza a utilização de conhecimentos de um modo prático e desafiador. Acredita-se que o usuário pode sentir-se motivado a construir modelos eficientes de linhas de produção no **Enxuto**, o que torna o sistema interessante para ele. Como outro resultado, os usuários podem se sentir incentivados a tomar a iniciativa de efetuar melhorias na linha real de produção, algo muito desejado segundo as novas técnicas de manufatura.

Durante as discussões iniciais do projeto, verificou-se que poderiam existir situações onde o usuário não conseguisse identificar alguma melhoria possível no modelo, o que o desmotivaria. Buscando minimizar a possibilidade de ocorrência deste tipo de situação, foi

---

[1] Subsidiária da General Motors Corporation (GMC) localizada em Piracicaba-SP.
[2] Núcleo de informática aplicada à educação, da Unicamp.

proposta a construção de um sistema que apoiasse o usuário quando este achasse necessário. Esse sistema é o **Jonas**, descrito neste trabalho.

O sistema **Jonas** funciona como um conselheiro do usuário do **Enxuto**, ajudando-o a enfrentar o desafio de formar funcionários do chão-de-fábrica em modernas técnicas de produção. Acredita-se que este auxílio torne a utilização do **Enxuto** mais interessante, uma vez que devem ser minimizados os casos em que o usuário não consiga identificar problemas no modelo ou oportunidades de melhoria.

O **Jonas** é um sistema especialista (SE) que usa uma base de conhecimento e resultados de modelagens e simulações feitas no **Enxuto**, buscando apresentar informações que ajudem os usuários a identificar as conseqüências de uma alteração ou partes do modelo que possam ter seus resultados melhorados.

Buscou-se construir um sistema que contribuísse com a idéia de aprendizado construcionista no **Enxuto**. Deste modo, o método de *design* do **Jonas** e a estratégia de uso da inteligência artificial (IA) no seu funcionamento foram baseadas em teorias que, em conjunto, poderiam levar ao paradigma do construcionismo. Em especial, o funcionamento do sistema baseou-se no personagem *Jonah*, do livro A Meta (Goldrat et al., 1994). Este personagem é um professor universitário que oferece consultoria a uma empresa que enfrenta problemas em manter-se competitiva. Nesta consultoria, o *Jonah* nunca indica as soluções dos problemas identificados, mas leva os próprios funcionários da empresa a encontrar as soluções, baseados em informações passadas por ele. O objetivo do **Jonas** é prover o usuário com um tipo de apoio similar.

O capítulo 2 deste trabalho indica os objetivos gerais do trabalho.

No capítulo 3 é apresentado um histórico dos principais sistemas computacionais para ensino/aprendizado existentes, classificados segundo a metodologia de ensino em que se baseiam e as técnicas computacionais utilizadas.

No capítulo 4 é discutida a base teórica deste trabalho. São apresentados os paradigmas de aprendizagem mais importantes, as diferentes formas de se efetuar o *design* de um sistema e estratégias de exploração de IA, modelagem e simulação em sistemas de ensino/aprendizagem.

O capítulo 5 discute as técnicas de produção modernas e como elas podem fazer parte da cultura de uma empresa. Ao final é discutido como a formação de pessoal deve ser repensada de modo a preparar pessoas para as novas necessidades criadas pelas metodologias modernas de produção.

O capítulo 6 apresenta o funcionamento do **Enxuto** e do **Jonas**, baseado em suas funcionalidades. As interfaces são apresentadas e um exemplo de interação usuário-**Enxuto**-**Jonas** é descrito.

No capítulo 7 é discutido o processo de desenvolvimento do **Jonas**. São apresentadas as etapas do *design* da interface e a construção do SE e do subsistema onde a base de conhecimento é construída.

O último capítulo mostra as conclusões do trabalho e discute possíveis oportunidades de continuidade para a pesquisa.

2 - Objetivos

Os objetivos deste trabalho são:

- identificar como concretizar, em um ambiente computacional, um sistema com comportamento que preserve estratégias usadas pelo personagem *Jonah* do livro "A Meta";
- buscar um conjunto de técnicas que levem à construção de sistemas motivadores, efetivos e viáveis para uso na formação de pessoal em empresas. O trabalho utilizou-se do *design* centrado no usuário, embora esta técnica não garanta, isoladamente, que o usuário poderá aprender com o sistema (Norman e Spohrer, 1996). É preciso utilizar-se de outras estratégias para que o sistema seja efetivo e viável no sentido da formação de pessoal;
- construir um protótipo de sistema que atenda aos objetivos indicados e exemplifique a utilização das estratégias propostas;
- identificar como efetuar o *design* do **Jonas** de modo a definir um sistema que possua uma interface simples e direta, que não direcione a atenção do usuário para aprender sobre o sistema, mas sim, com o sistema;
- identificar oportunidades de pesquisa no ambiente industrial, assim como estratégias de desenvolvimento que contemplem todas as peculiaridades existentes para a formação de pessoal nesse ambiente, pouco estudadas academicamente.

3 - Histórico sobre sistemas computacionais para ensino/aprendizagem

A construção de sistemas de aprendizagem baseados em computador foi iniciada na década de 50. Dessa data até a atualidade surgiram vários paradigmas indicando formas de se utilizar o computador na educação, alguns apontando para direções opostas entre si. Com o objetivo de situar este trabalho, neste capítulo é apresentado um histórico com exemplos de sistemas construídos segundo vários destes paradigmas. Entre eles está o sistema **Jonas**.

Os primeiros sistemas computacionais para ensino/aprendizagem construídos foram os de instrução programada (computer aided (ou assisted) instruction, CAI). A construção de sistemas CAI foi iniciada na década de 50: eram programas lineares, que não proviam individualização, baseados no paradigma de ensino instrucionista. Neste tipo de sistema, pouca ou nenhuma iniciativa é reservada ao estudante, uma vez que o sistema funciona como uma máquina de ensinar interativa (Perez e Seidel, 1987). Todos os estudantes são apresentados à exatamente ao mesmo conteúdo, na mesma seqüência (Nunes et al.,1993).

Na década de 60 surgiram os sistemas CAI ramificados (branching programs) que selecionam o material a ser apresentado, baseando-se em uma análise das respostas anteriores dos estudantes. Os sistemas CAI adaptativos (generative or adaptative systems) surgiram na década de 70, sendo capazes de formular suas próprias questões em áreas específicas - como a matemática - combinando elementos de sua base de dados. Este tipo de sistema pode gerar o material de ensino e resolver os problemas avaliando as necessidades dos estudantes (Nunes et al., 1993), mas só é capaz de construir questões do tipo "exercício e prática" (Perez e Seidel, 1987). Como o conhecimento não é organizado de modo similar ao que é observado no homem, os sistemas CAI não conseguem responder certas questões (Yazdani,1987). Para representar o que, para quem e como está sendo ensinado, assim como explicar a lógica seguida pelo sistema, há uma demanda por uma representação explícita do conhecimento em três domínios diferentes: especialista (domínio), instrução tutorial e conhecimento do estudante

(Nunes et al.,1993), o que não existe nos CAI. Na década de 80, com a disseminação dos microcomputadores os sistemas CAI se difundiram.

O DENDRAL, desenvolvido na década de 70, teve um grande sucesso industrial e comercial. Este sistema tem como objetivos auxiliar um químico no processo de identificação de estruturas moleculares. Usa técnicas de IA em manipulações simbólicas de dados. O grupo que desenvolveu o DENDRAL buscou um método para codificar as regras e procedimentos usados por químicos na interpretação e previsão de espectros (Gray, 1988). A experiência de construir sistemas como o DENDRAL fez com que uma ênfase em sistemas especializados em um certo domínio emergisse em IA, em contraposição à ênfase inicial dada a métodos gerais de resolução de problemas (Kulikowski, 1988). Com isto, o paradigma de IA moveu-se da generalidade para sistemas baseados em conhecimento (Feigenbaum e Buchanan, 1993; Kulikowski, 1988). Unindo-se a esta linha de pesquisa, novos sistemas educacionais foram construídos usando SE. Estes sistemas foram chamados tutores inteligentes (intelligent tutoring systems, ITS) ou sistemas de instrução programada inteligentes (ICAI) (neste trabalho todos são denominados ITS).

O desenvolvimento dos ITS preocupa-se com aspectos relativos a técnicas de representação de conhecimento, diálogos em linguagem natural e mecanismos de inferência, enquanto a construção de sistemas CAI, que baseia-se em teorias instrucionistas, fixa-se em características funcionais ou de domínio (Kearsley, 1987A). O interesse inicial dos pesquisadores de ITS era avaliar IA usando processos de instrução e não melhorar instrução com IA (Perez e Seidel, 1987). Um ITS completo age como um tutor gerando problemas, comparando as respostas dos estudantes com as de especialistas, diagnosticando fraquezas, associando explicações e exemplos com erros, decidindo como e quando intervir e fornecendo correções (Gladwin, 1984) (vide figura 1). Segundo Wenger (1987), os ITS podem ter dois tipos de estratégia de ensino:

- oportunista: explora oportunidades de ensino que aparecem durante alguma atividade na qual o estudante está envolvido;
- planificada: baseado em metas pedagógicas, o sistema controla e organiza as atividades e as interações com o estudante.

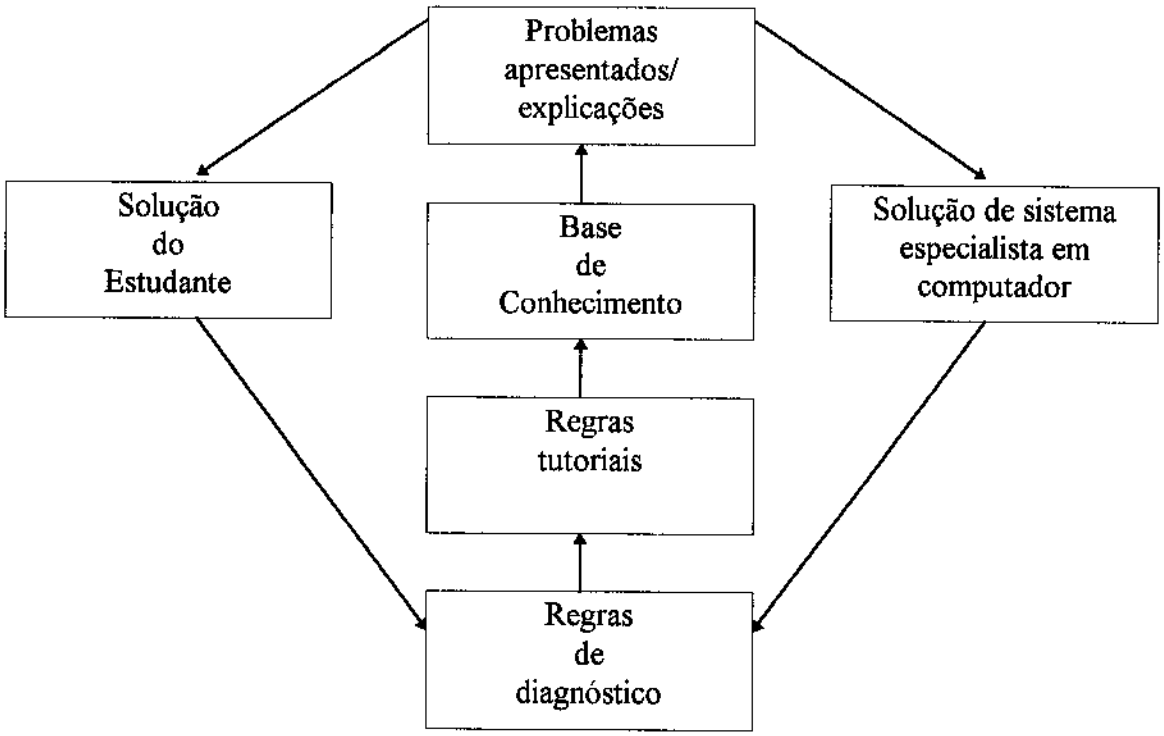


Figura 1: Modelo geral de funcionamento de ITS (Perez e Seidel, 1987, p.36)

O sistema SCHOLAR (desenvolvido em 1970) estimulou o desenvolvimento de ITS. Baseado em um banco de dados complexo e com estrutura bem definida, o sistema usa o estilo socrático de diálogo para ensinar a geografia da América do Sul (Perez e Seidel, 1987). No método socrático, o tutor não ensina uma matéria por exposição direta, mas leva o estudante, através de uma seqüência de questões, a formular princípios gerais baseando-se em casos específicos, a examinar a validade de suas hipóteses, a descobrir contradições e, finalmente, a extrair inferências de fatos que ele conhece (Wenger, 1987).

SOPHIE é o mais conhecido dos primeiros ITS. SOPHIE também utiliza-se do método socrático de descoberta guiada para ensinar. Sua interface é baseada em linguagem natural, o que habilita uma iniciativa mista entre estudantes e computador, com perguntas e respostas provenientes dos dois lados (conversação bidirecional) (Brown et al.,1982; Kearsley, 1987B). A área envolvida é a de circuitos eletrônicos. Seu propósito pedagógico é ser um ambiente de aprendizado reativo onde os estudantes podem testar suas idéias, tê-las criticadas e receber conselhos, sendo encorajados a experimentar e aprender com seus erros. Como o DENDRAL,

o SOPHIE é composto por uma ferramenta de *software* controlada por um especialista: utiliza-se do SPICE (simulador de circuitos eletrônicos) para avaliar hipóteses e responder questões. SOPHIE I (1973-1975) é um laboratório automatizado com um instrutor que, baseado em um simulador de circuitos, apresenta críticas inteligentes. No SOPHIE II, um especialista em resolução de problemas é adicionado ao laboratório. O SOPHIE III (1976-1978) é capaz de construir um modelo do estudante, o que possibilita atividades orientadas as características de um certo aprendiz..

Sistemas do tipo *coaches* objetivam identificar deficiências na compreensão dos estudantes, observando o comportamento destes e identificando estratégias de tutoramento explícitas, de modo que o sistema tutor diga a coisa certa na hora certa (Perez e Seidel, 1987). Um exemplo deste tipo de ITS é o WEST. WEST busca ensinar manipulações apropriadas de expressões aritméticas em um ambiente de jogo em computador. No WEST, o sistema não controla o ambiente, mas reage as ações do usuário de um modo instrutivo (Kearsley, 1987B; Perez e Seidel, 1987).

Na decorrer da década de 70, os avanços em IA passaram a ser usados mais efetivamente em sistemas educacionais. Começam a surgir sistemas de aprendizagem que funcionam baseando-se em ferramentas inteligentes (diferentemente do que ocorria, por exemplo, com o DENDRAL e o SOPHIE). Um exemplo é o projeto GUIDON, que investigou a possibilidade de usar o MYCIN no ensino de estratégias de diagnóstico para estudantes de medicina. O MYCIN é um sistema baseado em conhecimento que recomenda a medicação adequada para certas doenças infecciosas. Este projeto foi continuado na década de 80 na Stanford University, gerando novos sistemas, como a nova versão do MYCIN (NEOMYCIN), uma interface gráfica de janelas (GUIDON-WATCH), entre outras (Richer e Clancey, 1987).

Na linha do SOPHIE III, outros ITS tentaram modelar os estudantes, identificando seus problemas de entendimento. Como exemplos podem ser citados o BUGGY e o PROUST.

O BUGGY (desenvolvido em 1975) baseia-se em uma biblioteca de erros (Wenger, 1987), possuindo um mecanismo para identificar porque um estudante está cometendo um certo erro. BUGGY tenta auxiliar no ensino de estratégias de resolução de problemas algébricos (Perez e Seidel, 1987).

PROUST é um sistema que encontra erros não sintáticos de programação pascal feitos por programadores não experientes. Quando encontra um erro, PROUST indica a linha onde este foi encontrado e sugere como corrigi-lo. Para conseguir fazer isso, os objetivos dos programas devem ser descritos em uma linguagem própria do ambiente, antes do aluno usar o sistema para identificar os problemas existentes em sua implementação. PROUST efetua uma análise baseada em intenções, usando para isso entradas, saídas, fluxo de dados e reconhecimento de padrões de erro em codificações (Johnson e Soloway, 1987).

Explorando os avanços da informática na área de interfaces, surgiram sistemas que usam mais interação e gráficos. STEAMER é uma ferramenta para treinamento de engenheiros, cujo objetivo é instruir sobre a operação de uma usina de energia, possibilitando um desenvolvimento dos modelos mentais dos estudantes. O *design* teve como meta a construção de uma interface de manipulação direta, com uma simulação interativa e inspecionável, baseada em gráficos coloridos que representam com razoável fidelidade painéis existentes em uma usina, além de representações de um modelo de usina funcionando como um todo (Hollan et al., 1987; Hutchins et al., 1986; Richer e Clancey, 1987; Wenger,1987).

Sistemas baseados na abordagem de micromundos têm-se tornado surpreendentemente efetivos (Wenger, 1987). Apesar de existirem autores que classificam os micromundos como ITS (Kearsley, 1987B), as diferenças entre esses paradigmas com relação à perspectiva educacional e funcionamento operacional são tão profundas (Perez e Seidel, 1987) que preferiu-se, neste trabalho, classificar micromundos como um novo tipo de sistema. Diferentemente dos sistemas CAI e ITS tradicionais, a abordagem educacional é voltada para a exploração e investigação, em um processo de construção de conhecimento (Feurzeig, 1987), onde os erros são vistos como uma oportunidade de aprendizado. Sistemas neste paradigma são como sistemas físicos estudados por cientistas: não ensinam nem instruem, apenas têm um determinado comportamento. É o aprendiz (como um cientista) que aprende os princípios analisando o comportamento do sistema em experimentação (Thompson, 1987).

Uma implementação de sucesso da abordagem de micromundos é a linguagem LOGO. No final dos anos 70, Papert explorou o uso de linguagens de programação como o LOGO para ensinar de um modo diferente. O LOGO é uma linguagem onde são descritos os

movimentos de uma "tartaruga" na tela do computador. O aprendizado ocorre quando um usuário tenta desenhar algo com esta tartaruga. A tartaruga e o ambiente onde ela pode movimentar-se podem ser considerados um micromundo (Papert, 1987). Kafai (1996) também explorou a utilização de linguagens de programação no aprendizado, fazendo com que crianças criassem jogos educativos. Com isso as crianças aprendem a se expressar no domínio tecnológico, aprendem conteúdos usando-os e aprendem como aprender pensando nos futuros usuários de seus jogos.

Este novo paradigma, onde o usuário aprende fazendo experimentos em um ambiente computacional que não provê apoio instrucional explícito, é chamado construcionismo. Sistemas construcionistas apontam para uma direção diversa dos ITS e influenciaram muitas pesquisas na área de informática aplicada a educação.

O surgimento do construcionismo fez com que a pesquisa a respeito de *software* usados em formação se dividisse em linhas distintas, conforme indicado na figura 2.

Apesar do surgimento do construcionismo, os ITS continuam sendo muito pesquisados, além de possuirem atualmente um grande sucesso comercial. Muitas pesquisas em ITS contrapõem-se às idéias construcionistas. Atualmente, grandes projetos com ITS estão sendo desenvolvidos. No início da década de 80 foi construído o tutor de recuperação de caldeiras (Recovery Boiler Tutor, RBT). Neste ambiente o usuário tem acesso a uma ferramenta completa de simulação interativa. Para o grupo que desenvolveu o RBT, uma simulação sem um componente tutor não testará se um estudante realmente melhorou sua habilidade de enfrentar uma situação (Woolf et al.,1987). O grupo acredita que a habilidade de um sistema de IA em dirigir os estudantes no experimento é um apelo motivacional, divergindo do paradigma construcionista. Nos anos 90, um grande projeto conjunto europeu preocupa-se com a construção de ITS para ambientes industriais. Um dos sistemas resultantes deste projeto é o treinamento para manutenção de equipamentos elétricos (Electrical Equipment Maintenance Training, EEMT), composto por especialistas no domínio e em instrução, capazes de identificar incorreções nos conceitos em que se baseiam os estudantes. Tecnicamente, o EEMT inova ao usar programação orientada a objetos e interface de manipulação direta (Bertin et al.,1993).

Metáfora de Aprendizado	Ensino assistido por computador	Ambientes interativos de aprendizagem		Aprendizado socialmente distribuído	
Tipos de sistemas	Tutores (CAI e ITS)	Jogos e ambientes de simulação	Ferramentas de propósito geral	Aprendizado colaborativo	Sociedade de aprendizes
Exemplos de sistemas	SOPHIE, GUIDON, SCHOLAR, WEST, BUGGY, PROUST, STEAMER,RBT, EEMT, Cardiac Tutor	SimCity, **Enxuto, Jogo do alvo,** HyperGami, Broadcast news	Logo e outras linguagens. Editores. Planilhas Eletrônicas.	Colaborative Notebook, CSILE	WWW, Kidlink (?)
Controle	Computador	Misto (computador e aprendiz)	Aprendiz	Misto (computador e aprendiz)	Não existe um controle formal

Figura 2: tipos de *software* educacionais para estudantes

(estendido e adaptado de Riel et al., 1987)

Alguns projetos atuais mesclam o construcionismo com o instrucionismo. Segundo Eden et al. (1996), busca-se, com isso, evitar o pouco espaço para criatividade existente nos ITS e também os problemas decorrentes da falta de apoio oferecido em ambientes construcionistas. São ambientes computacionais que auxiliam os usuários a criar artefatos baseados em atividades onde subdividem os problemas e avaliam alternativas. Estes ambientes são orientados a um certo domínio, o que permite que o sistema ofereça um alto grau de apoio instrucional aos estudantes. Para Eden, neles, os usuários motivam-se e aprendem efetivamente. São sistemas controlados pelo aprendiz mas que provêem um apoio para este. Um exemplo deste tipo de sistema é o jogo SimCity™, onde o aprendiz modela cidades e simula seu crescimento. Outro exemplo é o HyperGami, onde o aprendiz cria e visualiza objetos em três dimensões na tela.

Na segunda metade da década de 90, passa a ser estudada a teoria do *design* centrado no aprendiz (DCA). A pesquisa de DCA aponta para sistemas inteligentes de simulação em multimídia. Neles, os estudantes fazem o papel de um profissional no trabalho (Norman e

Spohrer, 1996). Um dos trabalhos que seguem esta linha é o Broadcast News, que possuí uma arquitetura de aprendizado de cenários baseados em metas (goal-based scenario, GBS). No Broadcast News os estudantes aprendem estudos sociais definindo o conteúdo de um programa de notícias para televisão. O computador age como o produtor executivo, definindo um resumo da estória a ser descrita pelo estudante. O computador também provê críticos, que avaliam a estória escrita. São vários críticos com diferentes visões, podendo inclusive ter opiniões diferentes a respeito de um único trabalho. É o estudante que decidirá se alterará ou não seu texto baseado nas opiniões destes críticos (Schank e Kass, 1996). O sistema Cardiac Tutor (Woolf, 1996) ensina ressuscitação cardíaca em um paciente simulado. O tutor compara as ações dos estudantes com as de especialistas, definindo o problema de acordo com o nível do estudante e retornando informações sobre as ações corretas e incorretas. Apesar de tanto o Broadcast News quanto o Cardiac Tutor basearem-se no DCA, sendo sistemas de simulação em multimídia inteligente, com aprendizado baseado em metas, percebe-se que o primeiro aproxima-se mais do paradigma construcionista, enquanto o segundo, sendo um tutor, é mais instrucionista.

Com o advento das redes, em especial da rede mundial de comunicação (World Wide Web, WWW), muitos trabalhos voltam-se à identificação de como utilizar redes na educação. Redes viabilizam atividades onde estudantes desenvolvem seu próprio conhecimento de um modo colaborativo (Kay, 1995). Erich Neuwirth (1996) defendeu a disponibilização gratuita neste ambiente de pequenos programas com alguma habilidade instrucional. Para ele, com isso, os professores podem, em vez de ensinar, selecionar e indicar os programas que os alunos devem explorar. Utilizar a WWW na formação exige conhecimentos ainda não dominados, tais como técnicas de psicologia social e antropologia necessárias para se estudar a dinâmica de um grupo (Grudin, 1990). Devido a esta deficiência no conhecimento do funcionamento deste tipo de sistema, inúmeras pesquisas/experiências tem sido feitas, tais como:

- o projeto **Kidlink**, que visa disponibilizar uma rede mundial para crianças entre 10-15 anos (Presno, 1995);
- experiências com a disponibilização de ITS já construídos (Pimentel, Hagui, 1996);

- base de dados em rede onde os estudantes constroem informações em um trabalho conjunto, o que resulta em um aprendizado colaborativo. Um exemplo deste tipo de sistema é o Collaborative Notebook, que sugere uma estrutura básica de telas para auxiliar os estudantes a definir os passos a serem seguidos (Edelson et al., 1996). Outro exemplo são os ambientes de aprendizado intencional com apoio computacional (computer-supported intentional learning environments, CSILE), que provêem ferramentas de busca de outros pontos de vista de determinados assuntos (Scardamalia e Bereiten, 1996).

No Brasil, atualmente, várias linhas de pesquisa estão sendo desenvolvidas. O grupo da USP-São Carlos desenvolve pesquisas com ITS e sua possível disponibilização na Internet (Hasegawa e Nunes, 1995; Pimentel e Hagui, 1995). Na linha de micromundos, pode-se destacar o grupo da UFSC (Cardozo e Ramos, 1995; Cardozo e Mortari, 1996). O **NIED**, da **Unicamp**, desenvolve, entre outros projetos, sistemas voltados à aprendizagem baseados em jogos e em modelagem e simulação. Um exemplo de ambiente educacional baseado em jogo é o **Jogo do Alvo** (Fernandes et al., 1995). Um exemplo de sistema de modelagem e simulação, que baseia-se no construcionismo de Papert, é o **Enxuto**, do qual o sistema **Jonas** faz parte.

No próximo capítulo os paradigmas utilizados pelos sistemas apresentados são descritos, assim como toda a fundamentação teórica em que foi baseado o trabalho.

4 - Fundamentação teórica

Ao se construir um sistema computacional com objetivos educacionais devem-se analisar diversos fatores. Neste capítulo são apresentados os paradigmas de aprendizado mais importantes, estratégias para o *design* de interfaces e a diferença entre inteligência artificial (IA) e aumento de inteligência (AI). Também é discutido o uso da modelagem e simulação no aprendizado e interagindo com IA. Ao final são apresentadas as opções feitas no desenvolvimento do **Jonas**.

Inicialmente deve-se identificar o paradigma educacional a ser tomado por base, ou seja, qual a estratégia planejada para que um usuário aprenda interagindo com um sistema. Estes paradigmas são discutidos na seção 4.1. Identificado o paradigma educacional, deve-se selecionar o método de *design* do sistema. O método de *design* utilizado no desenvolvimento de um sistema influencia decisivamente no seu comportamento geral. Três destes métodos são discutidos na seção 4.2. Na seção 4.3 são apresentadas as principais vantagens de se utilizar modelagem e simulação na construção de sistemas para aprendizado. Na seção 4.4 analisa-se as vantagens de se utilizar conceitos de inteligência artificial em ambientes de modelagem e simulação. Ao se decidir pela construção de um sistema "inteligente", deve-se definir como melhor explorar essa "inteligência". Na seção 4.5 são discutidas duas diferentes maneiras de se utilizar "inteligência" em um sistema. Ao final (seção 4.6) são apresentadas as estratégias utilizadas na construção do **Jonas**.

4.1 - Paradigmas de aprendizagem

O desafio do conhecimento dos seres humanos tem se tornado maior a cada dia. As mudanças se aceleram e o que se aprende na infância pode não ser mais aplicável vinte anos depois. Deve-se rapidamente absorver novos paradigmas e visões do mundo (Kay, 1995).

Apesar de existirem formas variadas de se possibilitar a aprendizagem, não há como se classificar qual é a melhor. Não se pode perder de vista que estudantes e disciplinas são

diferentes entre si, requerendo diferentes abordagens desde as mais tradicionais às mais modernas (Norman e Spohrer, 1996).

Nas próximas seções são discutidos os paradigmas básicos de formação (instrucionismo e construcionismo) e como eles são transpostos para um ambiente de formação baseado em um sistema computacional.

4.1.1 - Instrucionismo e construtivismo

O ensino instrucionista, influenciado pela teoria psicológica *behaviorista* de Skinner, tem como objetivo transferir um conhecimento pronto, hierarquizado e compartimentalizado ao aprendiz, que funciona com um repositório de informações. Os conteúdos são apresentados de acordo com um plano prévio de ensino (Norman e Spohrer, 1996). Como o conhecimento resultante é desconectado, tende a manter-se inerte, podendo, posteriormente, vir a ser esquecido (Guimarães,1996). Nesta abordagem os erros são punidos e o aprendiz é passivo (Valente, 1993A). O instrucionismo é a base do sistema adotado na maior parte das escolas atuais.

O construtivismo opõe-se ao paradigma instrucionista. Segundo a teoria construtivista de Piaget, uma criança possui um mecanismo de aprendizagem próprio antes de ir para a escola. Para Piaget, a criança desenvolve sua capacidade intelectual interagindo com objetos do ambiente, sem ensino explícito. A meta é uma exploração ativa, onde se constrói o conhecimento, ao invés de se instruir através de aulas e lendo livros (Norman e Spohrer, 1996; Valente, 1993A). A construção do conhecimento é um produto da interação social (Scardamalia e Bereiter, 1996), onde a educação se baseia em um conjunto de problemas motivadores e realistas. Problemas reais têm um enorme potencial para aprendizado, porque, tipicamente, requerem um grande número de conhecimentos e habilidades para sua resolução e encorajam o aprendizado ao mostrar onde o conhecimento é útil (Guzdial et al., 1996). A resolução de problemas é uma forma de aprendizagem interessante porque aprende-se mais

profundamente técnicas para resolver problemas que são importantes, questionando, buscando respostas em diferentes fontes, considerando diversas perspectivas, trocando visões com outros e construindo seu próprio conhecimento (Kay, 1995). O construtivismo defende que os erros ajudam a entender ações e conceitualizações. O aprendiz construtivista é ativo no processo. A construção do conhecimento nem sempre é simples, mas esta dificuldade é produtiva: um sistema educacional que tenta tornar tudo fácil e agradável acaba evitando que ocorra importantes progressos na aprendizagem (Kay, 1995).

4.1.2 - O uso do computador

O uso do computador no ensino/aprendizagem pode ser interessante porque, dependendo de como é explorado, pode estimular a criatividade, a personalidade, o pensamento lógico, a formação de estratégias de resolução de problemas e o pensamento algorítmico (Kalas, 1996). Outros possíveis benefícios enumerados por Kay (1995) são: a interatividade, a habilidade de utilizar as diversas formas de mídias existentes, a disponibilização de informações em diferentes perspectivas (quando em rede, torna-se uma biblioteca universal) e de simulações, onde pode-se testar teorias conflitantes. Com o uso de redes de computadores, ambientes de aprendizado colaborativo tornaram-se mais viáveis. O aprendizado colaborativo habilita o estudante a resolver problemas e entender conceitos que ele, provavelmente, não conseguiria sozinho. Isto deve-se ao conhecimento distribuído e as múltiplas perspectivas. Apesar disso, o ensino/aprendizado colaborativo é complexo e conflita com a cultura de transmissão de conhecimento da escola atual (Edelson et al., 1996). Explorar estas vantagens do uso do computador na educação é algo interessante. Por exemplo: as crianças adoram a excitação advinda de jogos, especialmente daqueles que possuem fases com níveis crescentes de dificuldades (Hakansson, 1990). Isto torna os jogos uma fonte de pesquisa importante na identificação de como construir sistemas para ensino com os benefícios do uso da informática na educação.

Entretanto, a mera existência de computadores não melhora o ensino. Os computadores não substituem os professores: são apenas mais um material, que pode servir como amplificador, estendendo o alcance e a profundidade do aprendizado (Kay, 1995).

Baseando-se nas teorias existentes, podemos identificar duas direções de ensino baseadas em computador. Na primeira, o *software* ensina o aluno, em um estilo instrucionista: neste grupo podem ser incluídos os sistemas tutores (CAI e ITS). Na outra direção, baseada no construtivismo, é o aluno que "ensina o computador" via *software*: neste caso, o computador é uma ferramenta que resolve problemas ou realiza tarefas. Como exemplos de ferramentas podem ser citadas as linguagens de programação. Quando o aluno *ensina* o computador, este é usado como uma nova mídia educacional, propiciando condições para que os estudantes procurem e selecionem informações, resolvam problemas e aprendam de um modo independente (Valente,1993A).

Para Nicol (1990), com os computadores sendo integrados nas salas de aula, os *software* que funcionam como ferramenta serão utilizados freqüentemente em muitas partes do programa de ensino. Mas Nicol acredita que a curiosidade e a motivação das crianças não são suficientes: é preciso um contexto que facilite a descoberta e modelos disponíveis que auxiliem a descobrir um método para a resolução dos problemas apresentados. É preciso também encorajar os estudantes a correr riscos.

Um exemplo de sistema onde o *software* não "ensina" explicitamente é o ambiente Logo, composto originalmente por uma linguagem de programação, onde o aprendiz controla os movimentos de um objeto no chão ou um ícone na tela (a *tartaruga*). A construção deste ambiente foi profundamente influenciada por conceitos e metodologias de IA e pelo construtivismo de Piaget. Para Papert, em vez de ensinar as teorias consideradas corretas, o mais importante é ajudar as crianças a desenvolver e verificar suas próprias teorias. O objetivo do Logo é transformar o computador em um simulador experimental universal, onde os aprendizes podem manipular e explorar conceitos em diversos domínios que não são previamente estabelecidos. O aprendizado deve vir de um processo de formulação de hipóteses,

teste e avaliação do resultado (Valente,1993B, Wenger,1987). Um exemplo: imagine que uma criança resolva desenhar um quadrado usando Logo. Inicialmente ela vai, usando uma linguagem de programação, descrever uma seqüência de passos necessários para atingir este objetivo. Em seguida, o computatador executará cada um dos passos. A criança avaliará, então, se os passos resultaram realmente em um quadrado. Por fim, se o resultado não foi o esperado, tentará identificar qual o erro existente na sua descrição inicial, corrigindo-a. Fazendo isso, a criança voltará ao primeiro passo, de uma forma cíclica. Sistemas que buscam o aprendizado através deste ciclo de descrição-execução-reflexão-depuração são ditos possuir a "estética Logo" (Valente, 1993B, 1996). O tipo de aprendizado resultante da interação com este tipo de sistema é classificado como construcionista.

Para que uma certa estratégia de formação seja utilizada em todo seu potencial, é importante que o usuário não necessite dispender muito esforço cognitivo no entendimento da interface do sistema com o usuário. Caso contrário, o "aprender sobre o sistema" será um processo que dificultará o processo de "aprender com o sistema", objetivo real de um sistema para formação. Por este motivo, é necessário identificar como construir um sistema com uma interface que minimize este problema. Na próxima seção discute-se o que é e como fazer o *design* de interfaces.

4.2 - *Design* de interfaces

O *design* de interfaces de sistemas deve se basear em três critérios básicos (Vertelney et al., 1990):

- usabilidade: pode o usuário facilmente aprender e interagir eficientemente, conseguindo as informações desejadas e atendendo suas expectativas?

- funcionalidade: o que deve ser disponibilizado na interface? Muitas vezes oferecer menos funcionalidades é melhor: existem casos onde é mais interessante apresentar as informações mais importantes de um modo claro, coerente e padronizado;

- estética e comunicação visual: organização espacial (*layout*), uso de ícones, gráficos e botões. Definição de quando usar texto e quando usar figuras.

Normalmente, no *design* das interfaces, o programador faz suas críticas "simulando" um usuário. Mas esta simulação sempre tem falhas. O melhor é realmente envolver os usuários no *design*, pois estes têm um conceito diferente do problema (Sellen e Nicol, 1990). Jogos são bons exemplos a serem avaliados, uma vez que as interfaces destes precisam ser não apenas funcionais e fáceis de usar, mas também divertidas (Crawford, 1990).

Mas o mais importante é a interface atender as necessidades dos usuários: de outro modo de nada adiantaria ela ter boas estética e consistência. Para isso, antes de iniciar a implementação, os desenvolvedores devem identificar as necessidades dos usuários (Vertelney et al., 1990). Se um sistema oferece para seus usuários características que não lhes sejam familiares, mesmo que poderosas, acaba sendo substituído por outras ferramentas mais familiares (Laurel, 1990B).

Interfaces para aprendizado possuem necessidades especiais de *design*, pois quando o usuário não conhece os conceitos básicos do domínio em questão (e não apenas o uso do programa) a interface precisa refletir princípios de ensino e modelos que sirvam de base. Neste tipo de interface, a animação pode ser aplicada com sucesso no apoio ao entendimento dos conceitos mais importantes. Deve-se preocupar em fazer com que a interface incentive o usuário a aprender com seus erros e não apenas a trabalhar segundo uma estratégia de "tentativa e erro" (Nicol, 1990).

4.2.1 - Engenharia cognitiva: teoria da ação

"Engenharia cognitiva é um tipo de ciência cognitiva aplicada que busca aplicar o que é conhecido da ciência no *design* e construção de máquinas" (Norman, 1986, p.31). Sua meta é estender o tema, mostrando como fazer melhores escolhas quando elas existem e quais análises devem ser feitas quando, como ocorre comumente, uma melhoria em um certo domínio leva a resultados inferiores em outro (Norman, 1986).

Segundo a teoria da ação, a interação do usuário com um sistema computacional envolve um ciclo de atividades relacionadas com a execução de uma ação e a avaliação dos efeitos resultantes desta ação (Norman, 1986). Para executar uma ação é necessário formar uma intenção e especificar a seqüência de passos necessária para se atingir os resultados intencionados (figura 3). A continuidade de uma interação depende da interpretação do usuário do estado atual do sistema, comparando-o com suas intenções iniciais. Isto envolve a percepção, interpretação e avaliação do estado do sistema depois de executada uma ação.

Nessa teoria, uma pessoa interagindo com um computador tem suas metas expressas em termos relevantes para si (psicológicos), enquanto os mecanismos e estados dos sistemas são expressos em termos físicos. A discrepância entre as variáveis físicas e psicológicas são chamadas golfos. Existem dois golfos: o da avaliação e o da execução (figura 3). Estes golfos devem ser analisados e minimizados no *design* e uso de um sistema.

4.2.2 - Interface de manipulação direta

Interfaces tradicionais funcionam como intermediárias entre o sistema e o usuário, reforçando para o último a idéia de que ele está usando um computador. Isto porque é exigida uma participação descritiva (informa-se o que se deseja e o "intermediário" executa). Mas os

usuários finais não desejam trabalhar com representações: eles trabalham mais motivados em um sistema que simule um ambiente real, onde eles possam agir diretamente em um contexto simulado (Laurel, 1986).

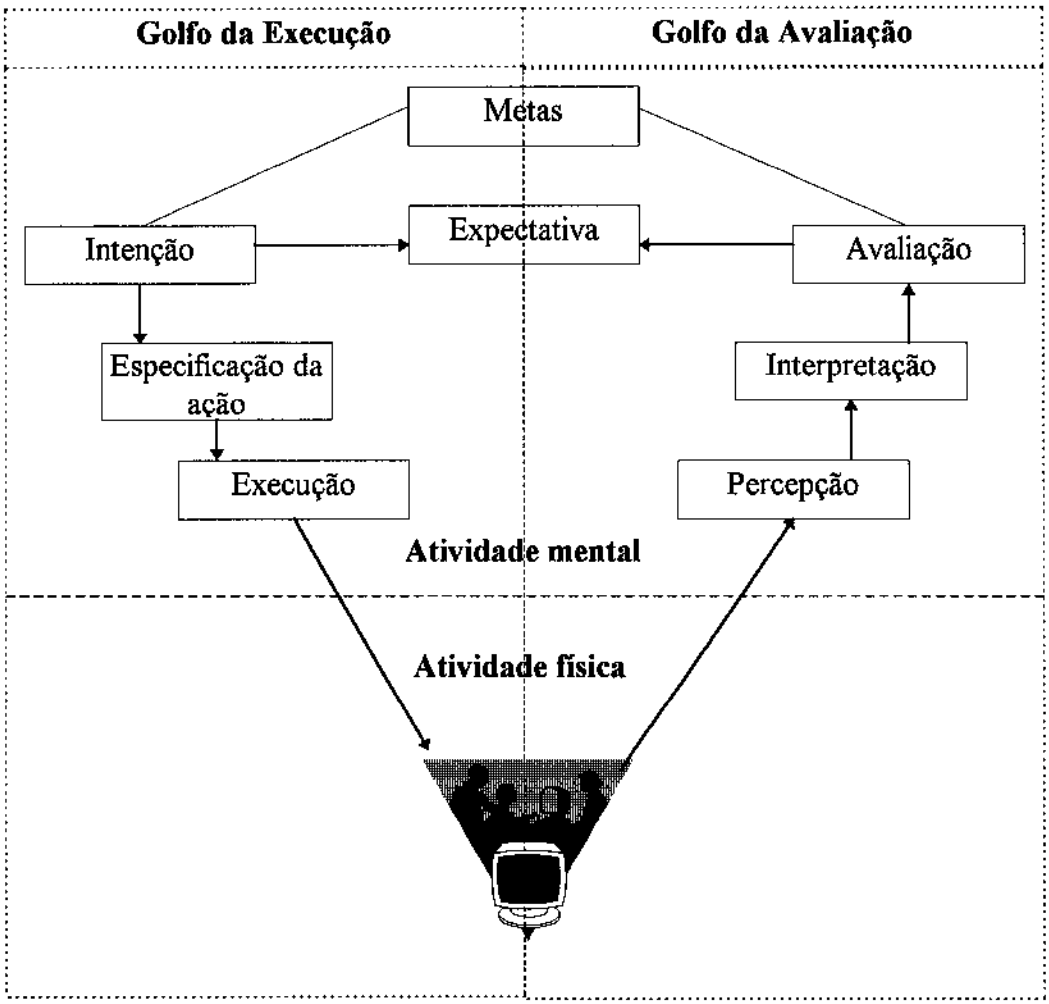


Figura 3: Os golfos (adaptado de Norman (1986), pags.40 e 42).

Objetivando tornar as ações dos usuários mais próximas às ações reais, os sistemas usam metáforas. Uma metáfora é uma rede invisível de termos e associações em que se baseia como se pensa e fala sobre um conceito. Metáforas funcionam como modelos naturais, permitindo estruturar-se conceitos abstratos baseando-se em objetos e experiências concretas e

familiares. Deste modo, as metáforas são partes poderosas e essenciais do pensamento. Uma boa metáfora é essencial para uma interface fácil de usar, pois provê um modelo que o usuário pode usar como base no entendimento do funcionamento do sistema. Identificar uma metáfora conveniente deve ser um trabalho cuidadoso, de modo a se escolher uma que provenha uma estrutura que realmente facilite o entendimento, sendo fácil de representar e conhecida pelos usuários, mas que não leve a expectativas incorretas. Uma boa metáfora deve também permitir extensões. De modo a verificar se a metáfora é realmente conveniente para o usuário, este deve ser envolvido em todo o processo de *design* (Erickson, 1990).

O conceito de interface de manipulação direta foi posto em prática inicialmente pela Xerox de Palo Alto e popularizado com o computador Macintosh, da Apple (Norman e Draper, 1986). A manipulação direta busca fazer com que, em vez de o computador servir como uma mídia computacional abstrata, todo o uso do computador seja feito graficamente, de um modo similar a como se pensa nos problemas. As operações desejadas devem ser feitas simplesmente movendo e conectando os ícones apropriados na tela. Não devem existir operações escondidas, nem comandos ou sintaxes para aprender. As representações de entrada e saída devem ser unificadas. Para operar um sistema de manipulação direta, é necessário conhecer o domínio, mas muito pouco de computação.

Como regra geral, conceitos e imagens são melhor representados através de sons e imagens em vez de textos, uma vez que gráficos expressam um conceito mais diretamente (Crawford, 1990). Como um sistema é um ambiente de informação, ele deve ser atrativo, provendo os usuários com todo tipo de informação sensorial que ele precisar (texto quando necessário, mas também imagens, animações e sons em conjunto) para interpretar o programa e responder a ele (Hakansson, 1990).

O sistema deve ter respostas rápidas, viabilizando um circuito de interação que aparente ser uma comunicação contínua. Qualquer demora pode quebrar a atenção do usuário e atrapalhar a continuidade da tarefa (Crawford, 1990).

As vantagens de um sistema com uma interface de manipulação direta são: novatos podem aprender as funcionalidades básicas rapidamente, mantendo os conceitos operacionais mesmo sem usar o sistema com freqüência; usuários experientes trabalharão de um modo eficiente; mensagens de erro são raramente necessárias; usuários têm um retorno praticamente imediato de suas ações, podendo avaliar se suas metas foram atingidas e, se não foram, podendo mudar a direção da atividade; ansiedade dos usuários é minimizada pela maior compreensão e facilidade em reverter suas ações.

O sentimento de que uma interface é *direta* é inversamente proporcional à quantidade de esforço cognitivo necessária para manipular e avaliar um sistema, ou seja, quanto mais *direta* é uma interface, menores os golfos da avaliação e execução (Hutchins et al.,1986).

4.2.3 - Tipos de *design*

Por mais de trinta anos o *design* de interfaces foi limitado pela tecnologia. Uma vez que não havia poder computacional suficiente para manipular a tarefa, pouco poder era "desperdiçado" com a interface (Soloway e Prior, 1996). Este era conhecido como o *design* centrado na tecnologia (DCT).

Em oposição ao clássico DCT, surgiu o *design* centrado no usuário (DCU), cujo objetivo é buscar a "humanização" do computador ao invés da mecanização do ser humano (Turcsányl-Szabó, 1996). Hoje, o DCU é o padrão desejável no desenvolvimento de interfaces.

Usando o DCU, pode-se considerar que o propósito de um sistema é servir ao usuário, e não usar uma tecnologia específica ou ser um programa elegante. Buscando atingir este objetivo, deve-se começar com as necessidades dos usuários, identificando como estes fazem normalmente a atividade a ser trabalhada através do computador. Como, para os usuários, a interface é o sistema, as necessidades destes usuários controlam o *design* da interface (participando de cada estágio de seu desenvolvimento) e as necessidades da interface controlam

o *design* dos outros módulos (Norman, 1986; Sellen e Nicol, 1990). O DCU baseia-se em duas questões importantes:

- Como o sistema pode facilitar ao usuário a formação da intenção e sua transformação em ações usando os mecanismos da interface (ou seja, diminuir o golfo da execução)?
- Como o sistema pode tornar mais simples para o usuário a interpretação dos resultados (*output*) (ou seja, diminuir o golfo da avaliação)?

Recentemente, surgiu um novo paradigma de *design*: o centrado no aprendiz (DCA), que busca auxiliar o usuário não apenas a efetuar uma tarefa, mas também a aprender enquanto a executa. Sistemas com este tipo de *design* necessariamente devem possuir as seguintes características (Soloway e Prior, 1996):

- adaptar-se ao conhecimento do usuário;
- adaptar-se à heterogeneidade dos usuários;
- tornar a interface atrativa de modo a que os usuários dediquem atenção à tarefa;
- focar nas necessidades, habilidades e interesses dos usuários, freqüentemente acompanhando a abordagem baseada em problemas.

A figura 4 compara os três tipos de *design* de acordo com o poder computacional disponível.

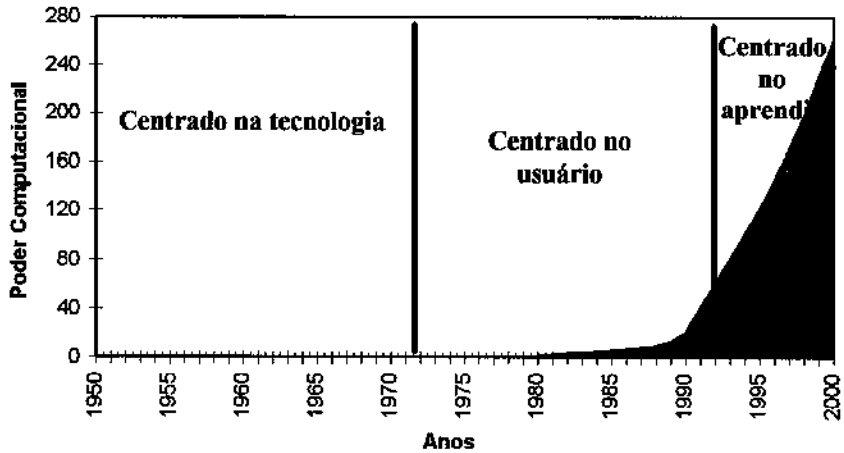


Figura 4. Os tipos de *design* de acordo com a evolução do poder computacional (adaptado de Soloway e Prior, 1996, p.17)

Interfaces de manipulação direta se adequam às atividades de modelagem e simulação. Este é um dos motivos pelos quais sistemas de modelagem e simulação estão sendo explorados com sucesso como ambientes de aprendizado baseados em computador. Na próxima seção discute-se modelagem e simulação e como estas técnicas podem ser exploradas para formação de pessoal.

4.3 - Modelagem e simulação

Chamamos "modelagem computacional" a construção de modelos e exploração de suas conseqüências usando computadores. Um modelo pode ser considerado uma representação que assemelhe-se a um certo conjunto de propriedades de um objeto ou sistema (Lee,1993), ou a representação de um sistema real que inclui as entidades deste sistema, seus comportamentos e interações (Pollacia, 1989). Através do processo de modelagem, o usuário pode avaliar não somente o modelo mas o seu próprio conhecimento sobre o sistema ou fenômeno alvo.

Simulação computacional é um processo pelo qual um sistema, baseado em um certo modelo, simula o comportamento dos objetos do modelo e as interações entre estes objetos através do tempo. Segundo Sol (1986), a simulação é vantajosa para efetuar pesquisas práticas no campo de tomada de decisão em organizações, sistemas de suporte a decisão e sistemas de informação, pois possibilita:

- enfatizar a atividade de conceitualização;
- experimentar, avaliando os dados disponíveis;
- gerar alternativas e analisá-las em comparação com uma especificação inicial.

Apesar destas vantagens, a indústria tem sido lenta em adotar a simulação como um meio de analisar problemas de decisão complexos (Fox et al., 1989). O problema é o tempo necessário para se adquirir as habilidades essenciais para a simulação (para O'Keefe, um ano de estudo e dois anos de prática (1989)). Os ambientes comercialmente disponíveis para a criação e simulação de modelos em empresas pressupõem uma competência formal em matemática que favorece os trabalhadores de nível gerencial, com uma formação mais completa (como é o caso dos *software* STELLA (Lindfield, 1992), ROBSIM (Wloka, 1989) e SSS (Muro et al.,1991)).

4.3.1 - Modelagem e simulação no aprendizado

A modelagem é um dos processos mais fundamentais da mente humana. Um modelo pode ser considerado o uso de algo em lugar de outro por algum propósito cognitivo, com menores custos (Rothemberg, 1989). Enquanto se constrói um modelo, aprende-se sobre o que está sendo modelado. Um ambiente de modelagem é um ambiente de aprendizagem *per se*.

A simulação de um modelo é um aspecto importante do raciocínio (Lee,1993). A simulação é uma estratégia de modelagem poderosa para entender fenômenos complexos (Rothemberg,1989), podendo ser considerada uma metodologia de resolução de problemas (Sol,1986). Em uma simulação, o usuário pode visualizar natureza, propriedades e funções dos objetos no mundo, além de poder tornar visíveis *objetos escondidos* como taxas, forças, relações de igualdade e proporcionalidade. Há evidências de que perceber estas entidades conceituais como objetos é vital para o desenvolvimento de uma capacidade de resolução de problemas (Wenger,1987).

Os benefícios de se usar simulação em contextos educacionais são bem conhecidos (De Jong, 1991; Pagano, 1992; Hebenstreit, 1991). Estes benefícios são ampliados se o usuário puder construir o modelo que ele quer simular. Em um ambiente de modelagem e simulação com objetivos educacionais, o objetivo não é aprender sobre modelagem e simulação *per se*,

mas aprender sobre o domínio através do processo de criação e simulação de modelos, onde idéias são testadas. Envolve um processo cíclico de construção de um modelo, análise dos resultados da simulação e reflexão sobre o modelo. Um trabalhador com competência para modelar e simular uma fábrica e seus problemas pode criar e testar melhorias no modelo e, posteriormente, implantar estas mudanças em uma fábrica real com um risco menor.

Sistemas de modelagem e simulação são potencialmente bons ambientes de aprendizado, uma vez que envolvem o aprendiz no ciclo básico de expressão, avaliação e reflexão sobre o domínio sendo modelado. Como o computador exige uma descrição formal de um modelo, o usuário precisa buscar um entendimento mais preciso de seu conhecimento sobre o domínio. Adicionalmente, a execução do modelo habilita uma avaliação que pode motivar o aprendiz a questioná-lo, reavaliando e reexpressando seu conhecimento, continuando o ciclo de ações em um estilo construcionista de aprendizagem (Papert, 1986; Valente, 1993b).

Apesar do desenvolvimento de ambientes comerciais de modelagem e simulação, ainda há pouca literatura sobre o uso destes sistemas no treinamento de funcionários em empresas.

Integrar os conceitos de IA é a tendência atual da pesquisa em modelagem e simulação. Na próxima seção são analisadas as vantagens da interação entre estas áreas.

4.4 - Modelagem e simulação interagindo com inteligência artificial

Enquanto que, a princípio, IA preocupava-se com a resolução de problemas, a modelagem e simulação concentrava-se em representações matemáticas de sistemas reais (Kulikowski, 1988). Unir estas duas linhas é algo que demonstra ter um grande potencial.

A IA pode ajudar a tornar os ambientes de modelagem e simulação mais amigáveis, auxiliar os usuários a construir modelos mais complexos, disponibilizar SEs que possam explicar ou justificar os resultados obtidos, etc.. A teoria de IA propõe novos paradigmas de programação para a simulação, como a orientação a objetos e sistemas baseados em conhecimento (Pollacia, 1989; Wang e Bell, 1992).

Por outro lado, ambientes de modelagem e simulação podem disponibilizar uma quantidade expressiva de valores a ser avaliada na busca por explicações e sugestões. Esta disponibilidade de um grande volume de dados pode ser explorada em testes de novas técnicas de IA (Widman e Loparo, 1989), servindo como uma fonte de conhecimento que programas de IA podem usar, por exemplo, fornecendo certos valores a um simulador e analisando suas respostas (Cao et al., 1993).

O aumento de inteligência foi proposto como uma alternativa ao uso tradicional da IA. Na próxima seção são analisadas as diferenças entre essas duas linhas. São comparados sistemas para formação tradicionais que usam IA e possíveis sistemas que explorem o aumento de inteligência.

4.5 - Inteligência artificial *versus* aumento de inteligência

IA é a exibição por computadores de um comportamento que seria considerado inteligente se realizado por seres humanos (Gladwin, 1984, p.46). Deste modo, tradicionalmente, sistemas inteligentes buscam imitar especialistas em suas habilidades. Em 1995, Fischer propôs uma nova forma de usar IA: o aumento de inteligência[1] (AI). Neste paradigma, a meta mais importante é construir um ambiente colaborativo de resolução de problemas, explorando as propriedades únicas da mídia computacional em vez de imitar a capacidade humana (Fischer, 1995).

No paradigma de AI a interação entre homem e computador explora as diferentes habilidades de cada um, dividindo a responsabilidade assumida entre eles. Busca organizar elementos que possam fazer a comunicação humano-computador mais fluida, deslocando o foco da atenção, em termos dos mecanismos de interação, do sistema para o usuário. Um ambiente de AI difere de SE e ITS tradicionais porque é um artefato computacional centrado no homem, e não possui uma arquitetura controlada pelo sistema. Os programas de AI usam

freqüentemente a abordagem crítica[2], identificando problemas que poderiam não ser percebidos sem o apoio do sistema. O aconselhamento é feito baseando-se em conhecimentos do domínio, examinando as ações dos usuários e os produtos criados por estas ações. Sistemas críticos auxiliam usuários que querem entender fórmulas e resolver seus próprios problemas, enquanto que, em ITS, os usuários querem resolver um problema proposto pelo sistema. Em sistemas AI o conhecimento não é fixo, mas é socialmente construído enquanto o sistema está sendo utilizado.

4.6 - O embasamento teórico do Jonas

O ambiente **Enxuto,** do qual o sistema **Jonas** faz parte, é um ambiente de modelagem e simulação desenvolvido com fins educacionais. Mesmo sem a existência do **Jonas,** poderia ser considerado um ambiente para formação. O sistema **Jonas** visa facilitar o aprendizado do usuário frente ao ambiente **Enxuto**, apenas entrando em ação quando o usuário desejar. Como é o usuário quem controla a interação com o sistema e como o **Jonas** não objetiva transferir conhecimentos prontos, mas apenas auxiliar o usuário a construir seu próprio entendimento, o paradigma de ensino seguido é o construcionista.

A integração do **Jonas** com o ambiente **Enxuto** é uma proposta diferente de utilização conjunta de modelagem e simulação e sistemas inteligentes: o sistema **Jonas** tem como objetivo aumentar o poder dos seus usuários e não imitá-los, uma vez que visa aumentar a possibilidade de aprendizagem no ambiente e não testar técnicas de IA. Ou seja, nesta proposta integra-se o AI (ao invés da IA tradicional) com modelagem e simulação.

De modo a se conseguir atender a todos esses objetivos, a interface precisou ser desenvolvida de um modo cuidadoso. O sistema, de um modo metafórico, funciona como sendo uma pessoa na fábrica simulada, que auxilia os usuários (que estão na posição de gerentes) a melhorar os resultados da empresa. O objetivo foi construir uma interface de manipulação direta e para isso baseou-se na teoria da ação. O processo de *design* foi baseado

---

[1] Intelligence augmentation
[2] critical approach

na teoria de *design* centrado no usuário. No entanto, como o usuário do **Jonas** é um aprendiz, e como durante o desenvolvimento houve a preocupação de se construir um ambiente de aprendizado efetivo, pode-se considerar o processo de *design* resultante como sendo centrado no aprendiz.

No próximo capítulo é apresentado com mais detalhe o domínio de conhecimento relacionado com o trabalho.

5 - O domínio da aplicação

O objetivo deste capítulo é traçar um panorama geral do domínio de conhecimento envolvido no trabalho, contextualizando e justificando a necessidade de formação de pessoal nos novos sistemas de produção.

Os desafios impostos à área de manufatura brasileira nunca foram tão urgentes e difíceis de serem superados. A cada dia novas teorias (ou filosofias) de gerenciamento nascem e a empresa que não se adequar a tais mudanças rapidamente corre alto risco de perder seu mercado para a concorrência. Hoje, *a velocidade empresarial tornou-se impressionante* (Veja, 1994, pág. 93).

Antes do advento da nova e poderosa indústria japonesa, programas de melhoria em empresas normalmente eram constituídos por projetos de modernização tecnológica (normalmente restritos a automação) definidos pelos níveis superiores da organização ou por consultores externos. O treinamento de funcionários constituía-se literalmente em ensinar uma reação adequada em um certo momento, reação esta definida pelos níveis superiores.

Mas a competitividade da indústria japonesa não será igualada se as empresas ocidentais continuarem apenas com este tipo de programas de melhoria. Os funcionários japoneses não são apenas bem treinados: eles têm uma postura diferenciada. Também as empresas japonesas portam-se de um modo completamente diferente.

O Brasil tem condições de tirar vantagens de seu atraso nesta área. *Deve olhar para os países desenvolvidos e copiar o que fizeram certo. Em alguns casos, deve agir diferentemente para reduzir os efeitos negativos causados pela modernização da economia* (Abranches, 1994, p. 95). O país conseguiu grandes progressos nos últimos anos. Entre outros exemplos, em "empresas enxutas" o prazo de entrega caiu de trinta e quatro para treze dias entre 1992 e 1994. Em 1994, das mil maiores fábricas, novecentos e cinqüenta e duas desenvolviam programas de qualidade (Beting, 1994). Mas ainda há um longo caminho a seguir, caminho este muito importante para a definição do futuro da indústria nacional.

Qualquer empresa que queira se manter competitiva no mercado mundial deve ser ágil ao implantar todas as melhorias conhecidas. Uma das mais difíceis barreiras a ser vencida é

romper a resistência às mudanças inerente ao ser humano, e mais especificamente às pessoas que trabalham na área de produção - o chão-de-fábrica - de uma indústria. Sabe-se que *mudanças comportamentais exigem alto esforço, dedicação e perseverança* (Telovi, 1994, p.7), o que torna estas mudanças o maior desafio a ser vencido.

É necessário desenvolver, em curto espaço de tempo, uma nova cultura industrial e uma nova postura profissional. Os funcionários devem assumir estas mudanças com entusiasmo, como lição de vida. A indústria ocidental atual ainda trabalha segundo a metáfora militar, baseada em controle, disciplina e responsabilidade. A nova indústria, diferentemente, deve enfatizar a informalidade, o "fazer" individual e a evolução (Peter e Waterman, 1984).

O pilar mestre do diferencial de competitividade da indústria japonesa é a filosofia de qualidade total (QT), criada por Deming (1992). Na busca pela qualidade total (QT) é necessário reduzir custos e, para isto, as estratégias de *just-in-time* (JIT), sistema de puxar e *kanban* são comumente utilizadas[1]. O conjunto de técnicas que permeiam a QT, o JIT, o sistema de puxar e o *kanban* é a base da produção enxuta (principalmente quando relacionado a indústria automobilística). Algumas características do funcionamento das empresas japonesas decorrentes da cultura nacional também fazem parte deste tipo de produção.

Uma discussão sucinta destas técnicas é mostrada nas seções 5.1 até 5.4. Na seção 5.1 discute-se qualidade total com base em Ciampa (1992), Cole (1989), Deming (1992), Peter e Waterman (1984), Telovi (1994), Wood e Urdan (1994) e 3M (1988A); na 5.2 o *just-in-time* com base em Moura (1989), Plantullo (1994) e Vollman et al. (1988); na 5.3 o sistema de puxar e o *kanban*, com base em GM (1989) e Moura (1989); e na 5.4 a produção enxuta com base em Mazzone (1993), Mazzone (1995) e Womack et al. (1992). Não há teoria melhor: a atitude mais produtiva é selecionar de cada uma o que ela oferece de mais adequado e com implantação mais viável para uma dada realidade. É importante observar que a bibliografia nesta área é extensa e, não raro, divergente ou contraditória. Nenhuma destas teorias possui um limite definido: a característica básica desta área é a alta taxa de intercambialidade e

[1] Além destas, também são utilizadas a reengenharia e o *benchmarking* (Coopers & Lybrand), os custos baseados em atividades (Marques,1994), o controle estatístico de processos (GM Brasil) e a teoria das restrições (Goldrat e Cox, 1994; Plantullo, 1994).

combinação entre todas. Outro problema é que as implantações de programas que baseiam-se nestas teorias nem sempre as seguem em todas as suas prescrições. Um exemplo comum é a divulgação constante pela imprensa de demissões geradas por programas de qualidade total, apesar de a demissão de funcionários ser fortemente desencorajada por Deming.

As melhorias decorrentes da QT só poderão ser percebidas quando esta filosofia alterar ou se inserir no conjunto de valores organizacionais existentes, definidos como cultura organizacional (Ferro,1995). Observa-se que mais que treinar, o que é necessário é mudar a cultura existente nas fábricas. Uma análise mais detalhada da cultura organizacional é feita na seção 5.5, com base em Ciampa (1992), Ferro (1991), Schein (1985), Semler (1988) e Womack et al. (1992).

Em 5.6 será discutido como a formação de pessoal deve ser repensada, de modo a suprir as necessidades do mundo moderno e, em especial, das empresas "enxutas" com base em Hakansson (1990), Kay (1995), Mazzone (1993), Mazzone (1995), Semler (1988) e Veja (1994), uma vez que os trabalhadores precisam se tornar tão flexíveis e aptos quanto as empresas.

5.1 - Qualidade total

Segundo Peter e Waterman (1984) qualidade total (QT) é a "total dedicação ao cliente". Para a 3M (1988B, p.3) é a "conformidade consistente com as expectativas do cliente". Para Feigenbaun (1988) a qualidade é um modo de vida para as empresas, uma filosofia de compromisso com a excelência. Existe muita bibliografia a respeito de QT e cada autor tem uma visão particular a respeito desse conceito. O que se pode afirmar, com certeza, é que QT é um conjunto de princípios de gestão de empresas criado pelo americano Deming (1992) que, depois de desprezado nos Estados Unidos, foi implantado com entusiasmo pela indústria japonesa, sendo o grande responsável pela alta competitividade desta indústria. Há mais de uma década a indústria ocidental busca implantar a QT, mas enfrenta problemas porque esta implantação envolve mudanças comportamentais, ou seja, uma mudança na cultura organizacional, o que é sempre complicado, delicado e demorado.

Hoje, muitas críticas estão sendo feitas à QT. Mas deve-se destacar que os insucessos relatados devem-se a falhas de implementação ou a programas cujo único vínculo com a QT é a denominação. Para se ter sucesso na implantação da QT é importante que seus princípios sejam incorporados aos processos organizacionais, em todos os níveis.

Uma empresa com QT deve visar as necessidades do consumidor atual e futuro, buscando a melhoria de produtos e serviços. Esta metodologia tem como objetivo manter a empresa, protegendo investimentos e assegurando dividendos e empregos: apesar de ser dirigida para o cliente, a QT é orientada para o crescimento e para o lucro.

Não há um limite para a qualidade. É um processo (sem fim) de melhorias contínuas[2], não um programa. Envolve flexibilidade, velocidade e economia de recursos. O objetivo é sempre minimizar o tempo de resposta a problemas e, se possível, antecipá-los e evitá-los; fazer as coisas corretamente desde a primeira vez e somente quando necessário; buscar o *zero defeitos.*

O engajamento total na melhoria contínua dos processos é fator essencial, pois a qualidade não se compra nem virá com a automação: somente poderá ser feita pelo homem, envolvendo todos os níveis da fábrica. Em empresas com QT, o trabalho em equipe, inclusive interdepartamental, é incentivado. Os funcionários recebem informações mais acuradas e participam da resolução de problemas e das decisões, correndo riscos e sentindo-se motivados e importantes. Utiliza-se o alto grau de criatividade na solução de problemas organizacionais largamente distribuído entre os funcionários, incentivando o espírito de descoberta. Nenhum funcionário deve perder o emprego havendo melhoria na qualidade, pois, sem segurança, não há interesse dos trabalhadores. Deve-se considerar que uma pressão negativa pode produzir uma mudança comportamental, mas freqüentemente estranha, imprevisível e indesejada. Já uma pressão positiva causa uma mudança comportamental normalmente na direção desejada. Não se pode esperar que os funcionários contribuam com suas idéias se não forem recompensados por isso.

Dividir a empresa em pequenos grupos é uma estratégia da QT. Em pequenos grupos o trabalhador pode dividir seus conhecimento e habilidade, e desenvolvê-los. Nestes grupos, os

---

[2] Em japonês, *kaizen*

indivíduos participam significativamente das decisões, o que é uma forte fonte democratizadora. O poder deixa de estar centralizado e passa a estar dividido entre todos os níveis da organização.

Uma empresa com qualidade total (QT) mantém uma constância de propósitos, num compromisso de longo prazo. Foca sua atenção em uma visão de um ideal realista, centrado em uma área específica de atuação, baseada em valores importantes para os empregados, motivando-os. Como este tipo de empresa busca uniformidade e confiabilidade e não apenas preço, o fornecedor deve participar dos projetos e garantir um aprimoramento constante, numa relação de confiança e lealdade de longo prazo.

O sucesso de um processo de QT é medido em termos da sobrevivência das inovações ou contribuições para melhorar a qualidade e a produtividade, do aumento da participação dos funcionários e de sua satisfação no emprego, da redução do giro de mão-de-obra e das faltas.

Programas de QT buscam, entre outras coisas, diminuir custos, como os de armazenamento de produtos pelas empresas. O *just-in-time*, descrito na próxima seção, surgiu a partir desta necessidade.

5.2 - *Just-in-time*

O *just-in-time* (JIT) é uma filosofia sem nenhuma técnica ou metodologia específica que visa um programa de produção consistente, estimulando a produtividade, eliminando estoques ociosos e melhorando a qualidade do produto (Guimarães, 1991).

O JIT busca minimizar o desperdício de tempo, energia e materiais na manufatura, eliminando tudo o que não agrega valor ao produto. O objetivo é fornecer exatamente o necessário, na quantidade necessária e no tempo necessário. Métodos de manufatura repetitivos de alta escala implicam em:

- diminuição e eliminação de lotes, substituídos por taxas de produção. O ritmo da produção é determinado pela demanda do mercado. Para viabilizar isto, é necessário diminuir os tempos de *setup* ou preparação;

- diminuição do estoque em processo;
- produção de vários produtos (ou modelos de) simultaneamente;
- implantação de sistemas visuais - tais como o sistema de puxar e, em particular, o *kanban* - ao invés de complexos sistemas de controle;
- estabelecimento de forte integração com o ambiente externo, comprando e vendendo produtos de alta qualidade com grande freqüência. Por exemplo, os fornecedores devem entregar diretamente na linha de montagem.

A implantação do JIT normalmente faz com que as fábricas agrupem os equipamentos em células de manufatura e trabalhem em um sistema de puxar. Uma célula de manufatura é um grupo de máquinas dedicado à fabricação de um grupo de peças (*layout* orientado ao produto). As células têm freqüentemente o formato de U, para minimizar a necessidade de transporte e de estoque entre as máquinas.

Como não são mais mantidos grandes estoques intermediários, a implantação do JIT exige uma melhoria contínua da linha. Por este motivo, é comumente implantado em conjunto à programas de QT. Na produção JIT, os trabalhadores devem ser multifuncionais, podendo efetuar diferentes funções conforme a necessidade de produção, além de participar da busca por melhorias.

O sistema de puxar e o *kanban*, ferramentas usadas na implementação do *just-in-time*, são descritos na próxima seção.

5.3 - O sistema de puxar e o *kanban*

O sistema de puxar (*pull system*) é um sistema de provimento de materiais que autoriza a manufatura a produzir e transportar apenas o material requerido na próxima operação. Ou seja, o consumidor autoriza o fornecedor a fabricar e mover os materiais na mesma taxa em que estes estão sendo consumidos (entenda-se consumidor e fornecedor em sentido amplo, podendo ser internos ou externos). Neste sistema só se produz quando houver vendas.

A implantação do sistema de puxar direciona uma fábrica para a manufatura sincronizada, eliminando as complexidades de programação de produção, organizando-a e viabilizando um controle visual das necessidades de manufatura. Como resultados, pode-se destacar: melhoria da qualidade dos produtos, eliminação de desperdícios, elevação do nível de controle da fábrica através da descentralização, redução do *lead-time*[3] e respostas rápidas da fábrica às necessidades dos clientes.

Um sistema de puxar bastante difundido é o sistema *kanban*. Sua tradução literal é registro visível de controle da produção e inventário no chão-de-fábrica. É um instrumento de controle que transmite informações da produção aos postos de trabalho interligados, com a função de avisar se há a necessidade ou não de se fabricar mais peças.

Estas novas técnicas de manufatura levam a uma mudança no paradigma de produção: da produção em massa para a produção enxuta. A produção enxuta é descrita em maior detalhe na próxima seção.

5.4 - A produção enxuta

Este tipo de produção foi desenvolvido pela Toyota e incorporado pelas outras indústrias japonesas nas décadas de 60 e 70. É normalmente colocada como uma evolução da produção em massa, como uma tendência para a indústria moderna, conforme ilustra a figura 5.

O desafio da produção enxuta (*lean production*) é minimizar a utilização de todos os recursos. Visa um melhor produto com maior variedade e menor preço. Para isso almeja a perfeição, buscando eficiência e flexibilidade.

[3] Lead-time é o tempo despendido entre a entrega da matéria-prima e a saída do produto. Também pode ser interpretado como o período para a renovação total do estoque (Vollman et al.,1988)

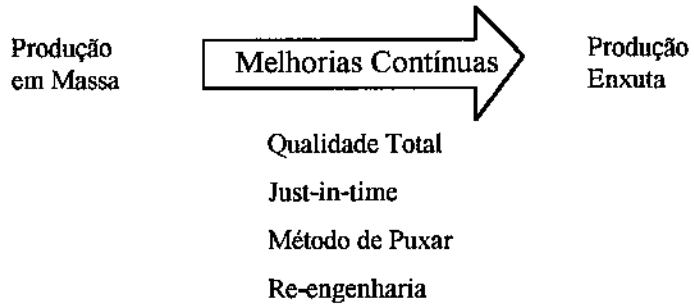


Figura 5: O caminho até a produção enxuta (Borges et al.,1995).

Um emprego *enxuto* é mais desafiador e gratificante, mas também mais *estressante*, exigindo maior responsabilidade. Neste tipo de produção não são valorizadas habilidades isoladas de funcionários, mas sim o quanto eles contribuem para seu grupo. A hierarquia é substituída pelo trabalho em equipe e a integração de atividades. A avaliação de resultados é feita por equipe (não se avalia o desempenho individual). Deste modo, a carreira, tal como se conhece no ocidente, perde seu sentido. Os funcionários devem possuir diversas qualificações. Precisam saber pensar, analisar e decidir por eles próprios. Devem estar familiarizados com o processo total de produção. Como exige uma formação mais completa de seus funcionários, a produção enxuta fatalmente pode levar ao desemprego de trabalhadores menos qualificados. Apesar de a teoria da produção enxuta definir os trabalhadores como "custo fixo", pregando evitar-se ao máximo demissões devido a ciclos de queda de vendas, todo o mundo observa um preocupante declínio no nível de emprego decorrente de sua implantação na indústria. Para se evitar este problema, a única solução, segundo Mazzone (1993), é melhorar a formação de toda a população com uma educação ampla, integrando-a à nova realidade enxuta mundial.

A produção enxuta tem como base a constância de propósitos. Em vez de incentivar a concorrência entre fornecedores, este tipo de produção prega uma parceria entre empresa e seus fornecedores, inclusive, com uma participação acionária mútua, formando grupos industriais integrados (em japonês, *keiretsu*).

Womack et al. (1992) defendem que, se um país subdesenvolvido incentivasse a implantação correta da produção enxuta em suas fábricas, conseguiria uma rápida capacitação em sua indústria, tornando-a de classe internacional, sem a necessidade de investimentos

maciços de capital. Mas, para a implementação desta forma de produção, é necessária uma mudança na cultura dos países e, mais particularmente, na cultura organizacional atual. Na próxima seção estas questões relativas a cultura são discutidas.

5.5 - Cultura organizacional

A cultura do homem foi formada durante centenas de milhares de anos em que este viveu como caçador, alguns milhares de anos em que viveu como agricultor e pouco mais de um século como industrial. Para entender este homem deve-se levar em conta todos estes componentes e não somente os últimos anos. A cultura tem vida longa e é pouco mutável.

Segundo Schein (1985, p.12) cultura é *um padrão de pressupostos básicos - inventados, descobertos ou desenvolvidos por um determinado grupo na medida em que aprende a lidar com seus problemas de adaptação externa e integração interna - que funcionou suficientemente bem para ser considerado válido e, portanto, para ser ensinado aos novos membros como o modo correto de perceber, pensar e sentir em relação a esses problemas.*

Em uma empresa as realidades organizacionais são socialmente construídas. Estas realidades resultam em um conjunto de significados compartilhados que formam uma subcultura industrial. Influenciam na formação desta subcultura muitos fatores, tanto internos quanto externos, entre eles: história e fundadores, cultura do país e de outras organizações com as quais se relaciona, tecnologia e tipo da indústria, estrutura e líderes e sistemas de controle.

Segundo Ferro (1991), uma cultura organizacional pode ser decomposta em três níveis:

- artefatos: as manifestações visíveis da cultura, como prédios, equipamentos, organização espacial;
- valores: tendência das pessoas da fábrica a preferir certas situações, baseados numa percepção subjetiva da realidade;
- pressupostos básicos: quando um valor é testado e percebido como válido, ele acaba tornando-se um pressuposto básico.

Ao buscar-se uma mudança de cultura organizacional, é necessário perceber estes três níveis. Se os valores e práticas da organização são congruentes com os pressupostos básicos, acaba-se formando um senso comum de direção e missão. Os valores explicitados sofrem grande influência do fundador e, se respeitados, tornam-se pressupostos básicos. Se esses valores permearem toda a organização, em todas as suas ações, acabam trazendo aos funcionários a sensação de que a empresa é justa (pois os pressupostos básicos são válidos). Tentar mudar estes pressupostos é sempre difícil, pois eles são usados como a referência do que é certo por toda a organização.

É muito difícil mudar a cultura organizacional de uma empresa. As mudanças encontram *barreiras monstruosas erguidas pelo tempo e pela acomodação* (Semler, 1988, p.52). As tentativas de mudanças devem ser efetuadas com critério: ao se anexar uma nova empresa, por exemplo, deve-se atentar para sua cultura anterior, pois ela tem mais a ensinar do que a aprender (afinal, segundo Schein, são padrões de pressupostos que foram testados e funcionaram bem).

Tentativas de se transplantar técnicas que funcionaram em outras empresas, em geral, não funcionam. Mudanças só podem ser buscadas respeitando-se as diferenças de cultura e dentro de um processo longo, com *encaixe cultural e orgânico com o resto do que se faz na empresa* (Semler, 1988, p.170).

Por todos estes motivos, as indústrias japonesas que implantam subsidiárias nos Estados Unidos, por exemplo, não reabrem fábricas antigas. Elas preferem construir fábricas novas a partir do zero (conhecidas como *greenfields*), utilizando-se de gerentes com anos de experiência em empresas enxutas. Isto ocorre devido a crença de que pessoas com formação em massa (baseando-se em pressupostos que adquiriram na produção em massa) não implantarão a produção enxuta, a não ser em casos extremos, quando a existência da empresa está seriamente ameaçada (como aconteceu com a Ford americana).

Uma melhor formação pode ser uma ferramenta poderosa na construção de uma nova cultura organizacional que contemple as necessidades das empresas atuais. A próxima seção analisa o processo de formação praticado atualmente e discute possíveis alternativas de atualização para este processo.

5.6 - A formação de pessoal

A nova realidade de trabalho, em empresas enxutas, exige dos funcionários muito mais preparo e flexibilidade do que a produção em massa. A escola do século XX, atrasada em relação a indústria, assemelha-se à produção em massa, uma vez que fragmenta o ensino, centraliza a informação e hierarquiza o conhecimento e as decisões. Isto deve ser revisto, pois esta escola:

- estimula a concorrência e a individualidade, ao privilegiar avaliações individuais e ao valorizar os alunos cujas notas se destaquem em relação ao grupo. O aluno formado nesta escola poderá ter dificuldades em participar de grupos e dividir responsabilidades;
- valoriza a passividade. O aluno é incentivado a "aprender" tudo o que seus mestres acharem ser importante[4]. É um processo tipicamente instrucionista. Neste tipo de ensino os estudantes são apresentados superficialmente a grandes descobertas, mas não são preparados para consegui-las. Este aluno, ao chegar ao mercado de trabalho, será desafiado a encontrar soluções próprias e a selecionar qual informação é importante, construindo seu conhecimento de modo independente. Como nunca pode escolher e discutir sobre os conhecimentos que lhe eram transmitidos na escola, existe a possibilidade deste aluno não saber como reagir frente a esta nova realidade. A distinção entre aluno e professor deve ser menor. Os dois devem dividir poderes e responsabilidades, com os professores sendo capazes de aprender com seus alunos;
- utiliza-se de ferramentas antiquadas (cadernos, lousa, etc.), enquanto poderia incentivar o uso de novos meios de informática e telecomunicações. Perde-se uma grande oportunidade de melhorar a formação, habituando os alunos a utilizarem-se da tecnologia. Pesquisas indicam que em salas de aula onde os estudantes usam computadores existe maior comunicação social e resolução cooperativa de problemas. Usando ambientes com muitas mídias também se desenvolve novas capacidades. Hoje, é mais importante saber e poder utilizar a tecnologia que criá-la, pois a supremacia tecnológica tem diminuído. Por exemplo,

[4] no caso do Brasil, é ainda pior: o MEC (Ministério da Educação e Cultura), em Brasília, decide o que é importante ser ensinado em todo o país, ignorando as grandes diferenças regionais.

sem saber informática não se consegue ser atualmente um bom profissional em áreas tão diversas como medicina, direito ou comunicação. Existe uma resistência por parte dos professores ao uso destas tecnologias, provavelmente devido ao temor de perder a autoridade frente a um aluno com liberdade e poder individual maior. Há também o medo de se perder o emprego, como ocorre com a indústria atual;

- segundo Mazzone (1993), requisita mais o pensamento dedutivo (do geral para o particular) que o indutivo (do particular para o geral). Para Mazzone, baseando-se no pensamento dedutivo, pode-se não usar todo o potencial das novas tecnologias, ao repetir-se velhos paradigmas;
- desperdiça muito tempo na memorização de detalhes. Mazzone (1993,1995) defende que em vez de perder tempo memorizando detalhes, deve-se ensinar como encontrar as informações quando necessário. O aumento da informação disponível e a facilidade em obtê-la tornam a ênfase à memorização algo indesejável e, muitas vezes, inviável. O mais importante é ser capaz de fazer um acesso crítico às informações (muitas vezes contraditórias), reconhecendo onde elas estão disponibilizadas;
- normalmente privilegia as disciplinas tecnológicas. No mundo globalizado atual é importante saber relacionar-se com outras culturas, o que fica em muito facilitado com uma formação ampla, não somente nas áreas de tecnologia, mas também nas humanas (especialmente em línguas estrangeiras) e nas artes.

Antes, as indústrias supriam as deficiências de formação de seus funcionários com treinamento. Mas hoje, os funcionários devem ser generalistas, não especialistas. Para isto é necessário uma formação ampla, o que exige, diferentemente da especialização, tempo e grande variedade de conhecimentos, o que é melhor adquirido fora da empresa. O sistema educacional e a própria pessoa devem ser os responsáveis por esta formação. Algumas empresas brasileiras estão investindo neste sentido: por exemplo, em 1989, 65% dos trabalhadores da Autolatina não possuíam o 1° grau completo. Este número havia caído para 45% em 1994, sendo que a maioria continuava os estudos em cursos mantidos pela própria empresa (Veja, 1994).

A ênfase às microempresas, o autoemprego, deve ser apoiada pelo sistema educacional, pois a participação destas empresas no mercado têm aumentado muito com a terceirização de processos.

O conhecimento mais importante é aquele que ajuda a se obter mais conhecimento: é o saber aprender e o saber descobrir. As pessoas mais bem sucedidas serão aquelas com pensamento livre, criativo, integrado e sem preconceitos, que olham mais para o presente e o futuro que para o passado. Neste ambiente, jovens bem capacitados e com muita criatividade abrirão suas microempresas e terão sucesso se tiverem acesso à informação e facilidade em aprender

O mundo passou da agricultura à indústria e, agora, passa para a era tecnológica, onde o bem vital é o conhecimento. Nesta nova realidade, surge a "indústria da aprendizagem", que virá a ser a maior de todas (Mazzone,1993).

Mas a indústria da aprendizagem precisa alinhar-se ao mundo moderno. As escolas devem se tornar menores, com mais qualidade e agilidade. Alunos, professores e escola devem buscar melhorar continuamente. O conhecimento dever ser *puxado* pelo aluno, e não *empurrado* pelo professor, uma vez que o melhor jeito de aprender é conseguir a informação exatamente quando ela é necessária (é quando o cérebro indexa mais rapidamente essas informações (Ellis, 1994)).

O sistema **Jonas** tem como objetivo auxiliar na formação de funcionários nas técnicas de formação no contexto de fábricas descritas neste capítulo, estimulando-os a ter a postura demandada pelas empresas. Também deve ser visto como parte de uma proposta de formação de pessoal diferenciada, que se utiliza das novas tecnologias e incentiva a participação e a criatividade.

No próximo capítulo os sistemas **Enxuto** e **Jonas** são apresentados, assim como um exemplo de seu funcionamento.

6 - Um ambiente computacional para formação em técnicas de manufatura

Neste capítulo, apresenta-se o ambiente **Enxuto** e faz-se uma descrição funcional do sistema **Jonas**. Buscando uma utilização efetiva na fábrica, este ambiente teve suas funcionalidades voltadas para a utilização por pessoal desse contexto, com todas as particularidades existentes. O projeto considerou o padrão de equipamento comum em fábricas, utilizando-se de *software* conhecidos e disponíveis.

Após estudos iniciais, foi proposta a criação de um ambiente baseado em modelagem e simulação. Este ambiente foi denominado **Enxuto**. A partir das atividades de modelagem e simulação e, contando com a ajuda de um SE (o **Jonas**), os funcionários têm a possibilidade de enriquecer seus conhecimentos sobre as novas filosofias de manufatura interagindo com o ambiente. Concomitantemente, os funcionários podem exercitar uma nova e desejável postura: testar mudanças na fábrica e analisar as conseqüências, refletir sobre o domínio, correr riscos.

6.1 - O ambiente Enxuto

Enxuto é um ambiente computacional de modelagem e simulação voltado para o contexto de treinamento em manufatura, desenvolvido para suprir as necessidades de trabalhadores do "chão-de-fábrica" (Borges et al., 1995). Como não é o objetivo do usuário aprender sobre o ambiente, mas sim aprender sobre o domínio usando modelagem e simulação, **Enxuto** foi desenvolvido de modo que o usuário possa focar nos objetivos realmente relevantes para ele. O ambiente cria um distanciamento do mundo real que faz com que os usuários trabalhem em um nível de abstração necessário para entender como o seu trabalho funciona. Busca-se estimular os trabalhadores a correr riscos através do processo de expor e testar suas próprias idéias, usando o computador. Neste ambiente, o usuário pode aprender técnicas de produção modelando e simulando, além de poder pedir ajuda ao SE **Jonas** quando não

entender ou não conseguir identificar como melhorar os resultados da simulação. Devido à preocupação em construir-se um ambiente educacional, acredita-se que o tempo necessário para que um usuário possa utilizar-se de todo o potencial de aprendizagem decorrente da modelagem e simulação, reforçado com a presença do **Jonas**, será sensivelmente menor àquele identificado por O'Keefe (como descrito em 4.4).

Enxuto é um ambiente de aprendizado baseado em computador. Nele o usuário pode construir o modelo de uma fábrica utilizando-se de tipos de objetos previamente definidos e disponíveis na interface tais como estações de trabalho, transportes, funcionários, técnicos de manutenção, etc.. Uma vez selecionados todos os objetos necessários, o usuário interliga-os, formando uma linha de produção. O usuário pode alterar atributos de modelagem de cada objeto, em uma tela específica. São atributos de modelagem de uma estação de trabalho, por exemplo, a velocidade de produção, o índice de defeitos médios, a quantidade de matéria-prima necessária para a produção de uma peça, etc.. Uma vez modelada a fábrica, o usuário solicita ao **Enxuto** que seja realizada uma simulação por um período de tempo específico, alterável. Ao final, ele pode analisar os atributos específicos de cada objeto resultantes da simulação. São atributos de simulação de uma estação de trabalho, por exemplo, a quantidade de matéria-prima consumida e de produtos fabricados durante a simulação, a proporção do tempo em que a estação ficou produzindo, ou quebrada, etc.. Fingindo ser o gerente da fábrica, o usuário analisa os resultados, tentando alterar o modelo de modo a obter melhores resultados na próxima simulação.

Diferentemente dos ITS tradicionais, neste tipo de sistema o aprendiz não é guiado por um professor especialista, mas está livre para uma exploração independente. O usuário aprende manipulando modelos simples de coisas importantes, idéias e seus relacionamentos, buscando e selecionando informações para resolver um problema, de acordo com a abordagem de descoberta ativa da realidade (Lawler e Yazdani, 1987; Lawler, 1987; Valente, 1993A.; Valente, 1993B). Uma característica que faz este sistema diferir bastante de um ITS tradicional é que ele não possui um módulo capaz de realmente resolver os problemas no domínio (Yazdani, 1987), apenas indica possíveis melhorias que podem ou não ser utilizadas pelo aprendiz.

O ambiente **Enxuto** pretende ser uma ferramenta de auxílio na construção de uma nov cultura nas empresas, a cultura enxuta. O objetivo básico é fazer parte do novo sistema d formação, viabilizando a implantação das técnicas modernas de manufatura. Este ambient estimula os funcionários a descobrir, a buscar melhorias, de acordo com a nova postur esperada. O sistema **Jonas**, em paralelo, ajuda os funcionários a construir seu conhecimento, identificando quais técnicas podem ser utilizadas com sucesso na fábrica.

A arquitetura do **Enxuto** é composta basicamente pelos módulos de modelagem, de simulação e de aconselhamento (**Jonas**), conforme apresentado na figura 6.

O primeiro módulo é composto de elementos que dão suporte à fase de modelagem, monitorando as ações dos usuários e apresentando *feedback*. Representa a interface de entrada e saída do sistema. Possui um editor onde o usuário pode criar modelos de processos de manufatura através de uma abstração do mundo real. Este editor disponibiliza elementos que permitem ao usuário construir modelos em um domínio específico. Estes elementos são chamados elementos atômicos ou primitivas e expressam o conhecimento de um subdomínio (Baranauskas e Oliveira, 1994; Oliveira, 1995). A atividade de construção de modelos pode ser vista como a ação de agregar primitivas de modo a construir estruturas mais complexas.

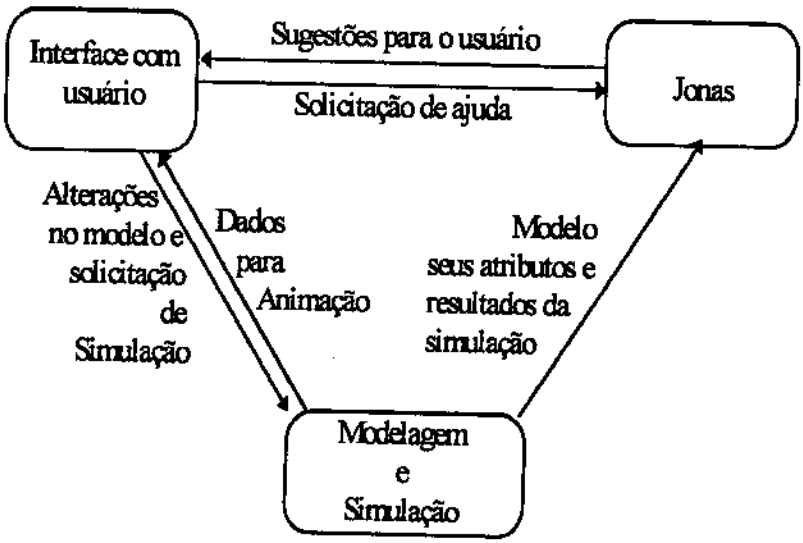


Figura 6: A arquitetura do **Enxuto**

O editor tem um conjunto de primitivas específico que permite ao usuário criar modelos de sua rotina diária, fazendo conexões entre primitivas, representadas por figuras na tela. O conjunto de primitivas definido para o domínio do **Enxuto** são: estações de trabalho (máquinas), operadores, técnicos de manutenção, clientes, fornecedores, conectores e estoques. A figura 7 ilustra a interface do **Enxuto**, apresentando um modelo exemplo.

O módulo de simulação é o responsável pela simulação do modelo. O método de simulação é discreto e orientado a eventos. Como apresentado na figura 6, o simulador interage com os outros módulos. Da modelagem ele obtém sua entrada (o modelo e seus atributos). Para a interface, retorna dados estatísticos e informações utilizados na apresentação visual de resultados. Para o **Jonas,** são transferidos os resultados da simulação.

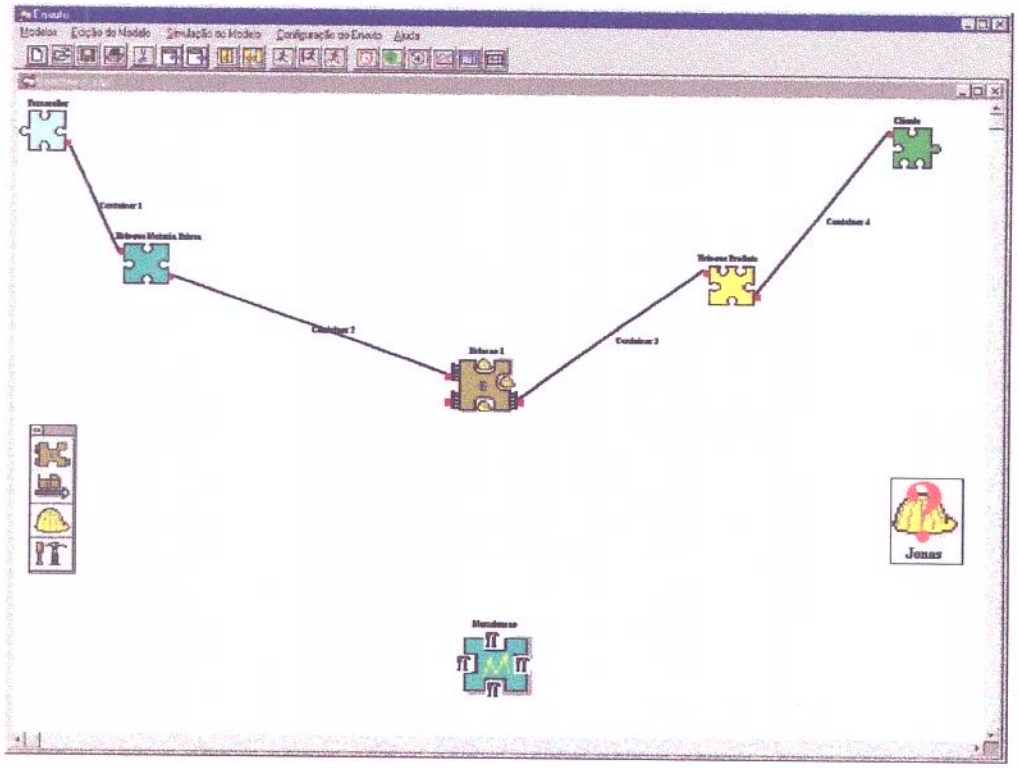


Figura 7: Interface do **Enxuto**

O componente de saída do **Enxuto** fornece ao usuário um *feedback* do estado simulação, essencial para o usuário analisar os resultados. O ciclo de atividades em que usuário se envolve durante o processo de modelagem e análise da simulação é uma fase críti que requer uma atenção especial, uma vez que, em muitas situações, os resultados não s auto-explicativos. Em muitos casos as representações dos resultados não são suficientes pa entendimento da situação; neste caso o usuário pode interagir com o **Jonas**, que prove explicações de resultados da simulação e sugestões de melhorias no modelo.

6.2 - O sistema *Jonas*

Jonas é um sistema que integra o ambiente **Enxuto** para auxiliar o usuário quando est acredita ser necessário. Seu funcionamento baseou-se na observação das heurísticas utilizada pelos especialistas da fábrica quando estes desejam apresentar explicações a outros funcionários. Como também há uma clara separação entre sua base de conhecimento e seu módulo de controle, **Jonas** pode ser considerado um SE (Ahamed e Roman, 1986; Luger e Stubblefield, 1989; Widman e Loparo, 1989). **Jonas** tem como objetivo aumentar o poder de seus usuários, e não gerar informações de um modo independente: funciona baseado no paradigma do aumento de inteligência.

Jonas não imita um professor, não testa e nem controla o estudante. Este detém o controle da interação, decidindo se precisa de ajuda e quando. Como o objetivo do sistema não é transferir conhecimentos prontos para os usuários, mas ajudá-los a construir o próprio conhecimento de um modo independente, baseia-se no paradigma de aprendizagem construcionista.

No desenvolvimento utilizou-se como premissa a não necessidade de uma ferramenta de ajuda explícita ao uso do sistema ("*help*"). As interfaces devem ser simples e auto-explicativas, de modo a que o usuário não necessite de ajuda (Sellen e Nicol, 1990).

O *design* foi centrado no usuário, objetivando a construção de uma interface de manipulação direta. Como a interface é a apresentação do sistema, qualidades como facilidade

de usar e atração podem ser cruciais para que os estudantes aceitem o sistema. O uso de um mídia poderosa pode fazer com que a representação externa seja a força que dirige o *design* d sistema.

O sistema busca sugerir conceitos de produção enxuta que podem ser usados par melhorar o modelo e explicar as mudanças nos resultados da simulação. Para isso, analisa o componentes do modelo, suas relações funcionais e características operacionais (Naylor et al. 1971).

Simplificadamente, o sistema **Jonas** é composto por:

- um sistema especialista: máquina de inferência e base de conhecimento;
- uma interface gráfica;
- um subsistema administrativo;
- uma representação orientada à objetos (OO) dos modelos e dos resultados das simulações.

Na figura 8 está esquematicamente representada a arquitetura do sistema.

A máquina de inferência pesquisará a base de modelagem e simulação, identificando quais alterações foram efetuadas pelos usuários no modelo e quais resultados mudaram significativamente na simulação. Então, a máquina de inferência usará a base de conhecimento de modo a identificar sugestões de como o modelo pode ser manipulado para obter-se melhores resultados ou porque os resultados da simulação mudaram após a última alteração no modelo.

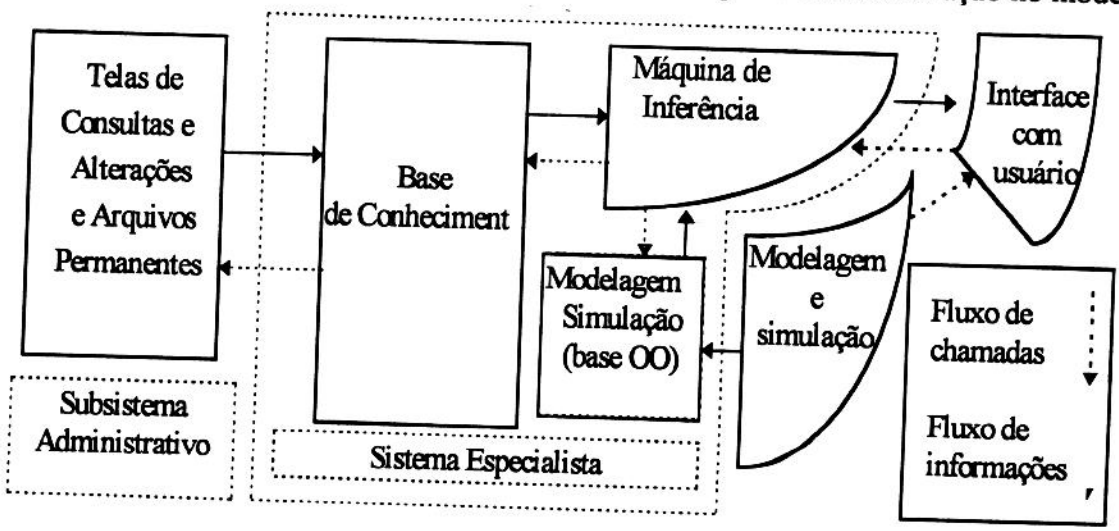


Figura 8: A arquitetura do Jonas

A construção da base de conhecimento mereceu uma atenção especial. O conhecimento é representado como um conjunto de instruções para resolver um problema, armazenadas em um banco de dados. Estas instruções, organizadas segundo um esquema de representação procedural são baseadas em regras e operações lógicas sobre atributos de modelos formando um sistema de produção (Luger e Stubblefield, 1989; Guida e Tasso, 1989). Neste tipo de representação, o trabalho de refinar e estender o conhecimento do sistema fica em muito simplificado (McDermott, 1993).

Facilitar alterações na base de conhecimento é um fator muito importante para o sucesso de um SE (Luger e Stubblefield, 1989). Por este motivo existe o subsistema administrativo, chamado por outros autores de subsistema de aquisição do conhecimento (Schlunzen, 1992). Este subsistema será utilizado pelo especialista no domínio da fábrica para construir e manter a base de conhecimento. Ele é composto por ferramentas para consulta e manutenção na base. Permitir ao especialista do domínio adicionar conhecimento e alterá-lo facilmente é crucial para o sucesso de um SE, pois torna possível construir uma base de conhecimento mais abrangente e atualizada por um período de tempo indefinido (Feigenbaum e Buchanan, 1993; Luger e Stubblefied, 1989).

A base, armazenada em uma estrutura relacional, proporciona o uso de inferência abdutiva. A abdução parte do princípio de que a partir de $P \Rightarrow Q$ e Q é possível inferir P (Luger e Stubblefield, 1989, p.309). Com este tipo de inferência pode-se testar hipóteses a partir de associações causa-efeito. Em problemas de diagnósticos baseados em modelos o uso da inferência abdutiva tem sido defendida (Boutilier e Becher,1995). Embora com resultados nem sempre corretos, a abdução é essencial para resolver problemas como os identificados nos modelos pelo **Jonas**: diagnóstico que parte dos sintomas buscando uma causa.

6.2.1 - Descrição funcional

O sistema **Jonas** é carregado quando o usuário clica sobre o botão[1] correspondente na interface de modelagem do sistema **Enxuto** (figura 7). Uma vez solicitado, é apresentada a tela inicial do sistema (figura 9). Esta tela apresenta uma descrição básica do sistema, estilizada como um pensamento de um personagem desenhado (**Jonas**). Entre as informações apresentadas, é importante observar: "Precisando, estarei em todas as telas indicando sua função. Em caso de dúvida, clique em mim !!!". Ou seja, a figura com o personagem funciona como sendo um botão a se recorrer em caso de dúvida.

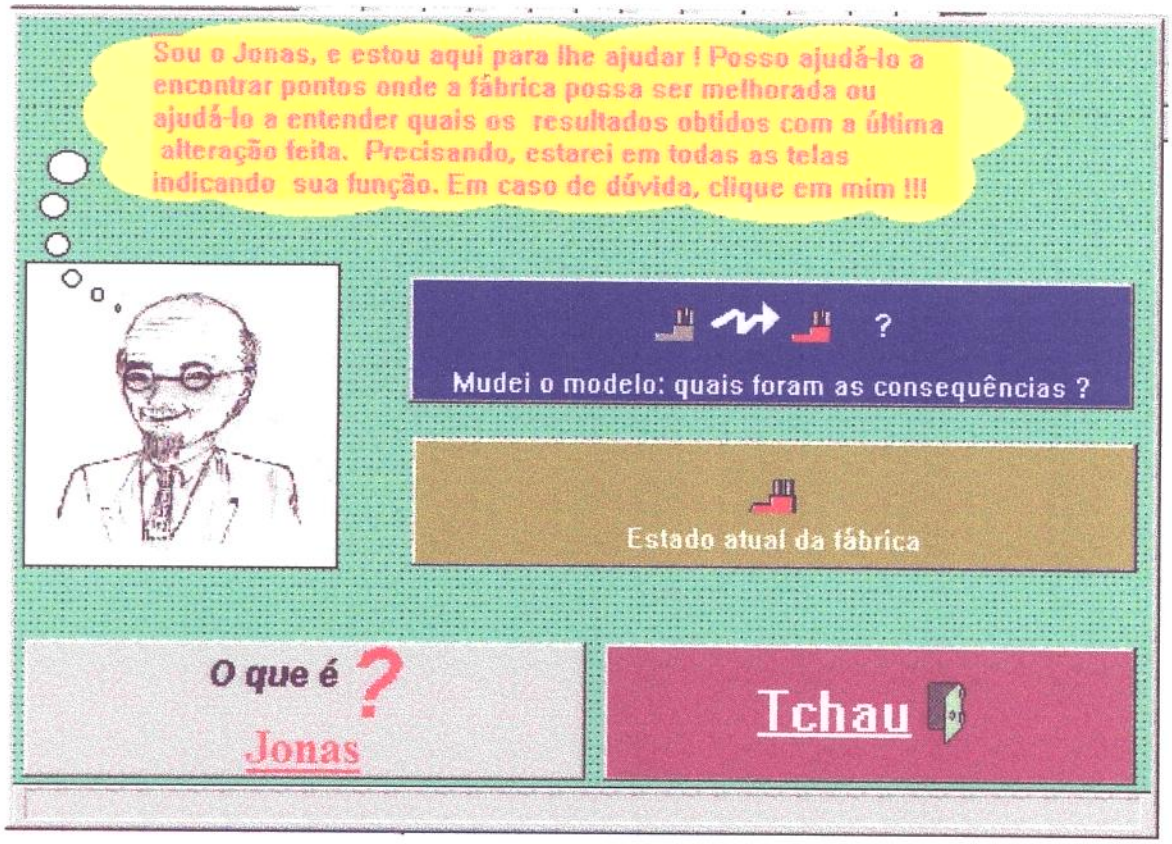


Figura 9: a tela inicial do sistema **Jonas**.

Estão também disponíveis na primeira tela quatro botões. Dois específicos e dois que serão apresentados em toda a interface do sistema. Os botões gerais são:

[1] Elemento de interface que simula o funcionamento de um botão de pressão.

- *O que é?*: um botão que leva a uma tela onde é explicado o termo indicado. Na tela inicial o termo é *Jonas;*
- *Tchau*: o usuário pode deixar o **Jonas** em qualquer ponto de seu funcionamento.

Os botões específicos são utilizados para se selecionar um dos dois possíveis auxílios que o sistema pode fornecer ao usuário:

- *Mudei o modelo: quais foram as conseqüências?*: apresenta listas com as mudanças que o usuário fez no modelo e as diferenças existentes entre os resultados das duas últimas simulações efetuadas. São apresentadas, também, informações que podem auxiliar o usuário a perceber como as alterações no modelo podem ter influenciado os novos resultados das simulações.
- *Estado atual da fábrica*: apresenta uma lista com atributos do modelo, de modelagem ou simulação, que devem ter seus valores avaliados, além de informações que auxiliam o usuário a perceber o que pode ser melhorado.

Na tela inicial, uma vez selecionado o *botão* "Mudei o modelo: quais foram as conseqüências?" é apresentada uma tela que possui (figura 10):

- um desenho com o personagem e informações básicas sobre a tela;
- os botões genéricos anteriormente descritos "O que é?" e "Tchau";
- outro botão genérico, o "Volte uma tela", que em qualquer ponto volta à tela anteriormente apresentada;
- duas listas de informações, uma com o que foi alterado no modelo e outra com as *alterações significativas*, ou seja, com os resultados da simulação que foram significativamente diferentes nas duas últimas simulações. Na segunda lista, o usuário pode selecionar uma linha;
- se o usuário não houver selecionado uma *alteração significativa*, estará sendo apresentado o botão "Estado atual da fábrica", que leva ao mesmo ponto do botão homônimo existente na primeira tela;
- se o usuário houver selecionado uma alteração, estará sendo apresentado o botão "Explica a alteração selecionada", que leva o usuário a uma tela de informações descrita a seguir.

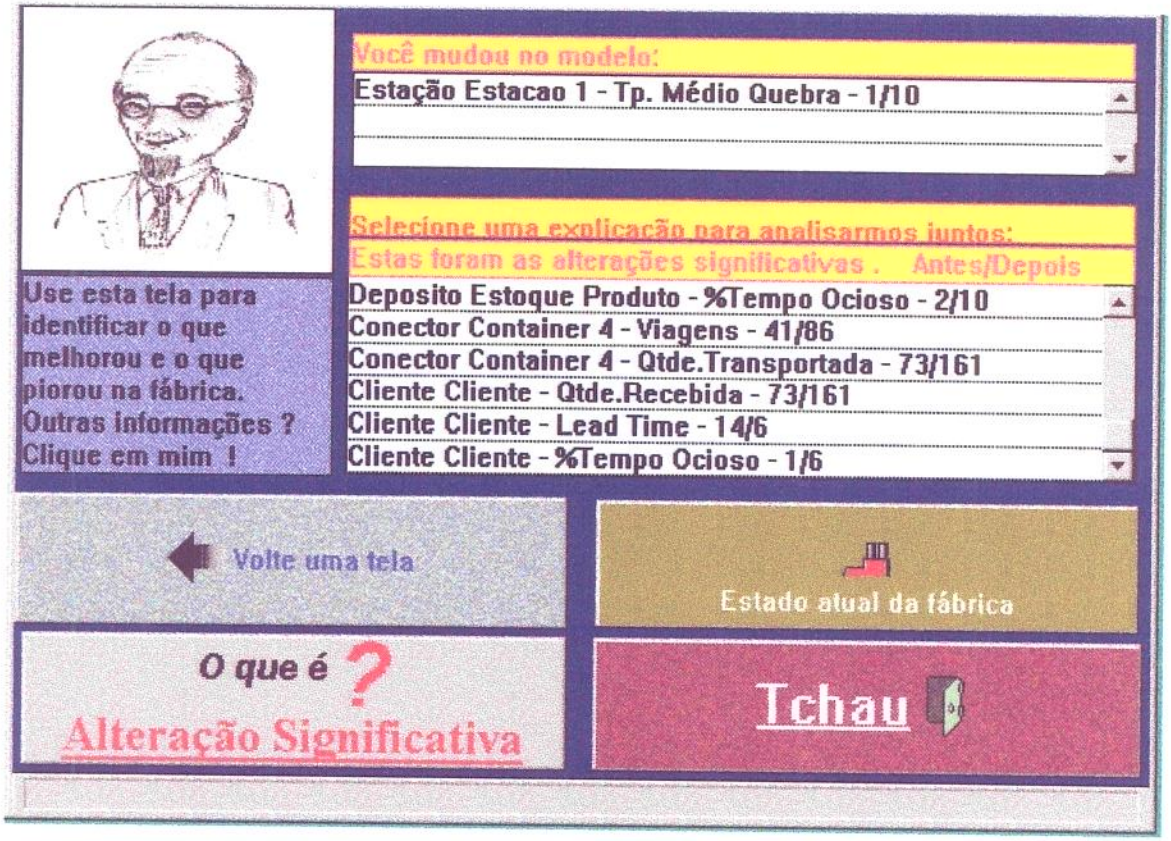


Figura 10: tela "Mudei o modelo: quais foram as conseqüências?".

A tela de informações apresentada (figura 11) possui os seguintes objetos:

- um desenho com o personagem;
- os botões genéricos anteriormente descritos "O que é?", "Tchau" e "Volte uma tela";
- um *balão* de texto apresentando uma informação;
- uma indicação da alteração significativa selecionada na tela anterior;
- outro botão, o "Continue": se ele for selecionado, novas informações serão passadas ao usuário, em telas iguais. Quando não houver mais nenhuma informação, o botão "Continue" não será apresentado;

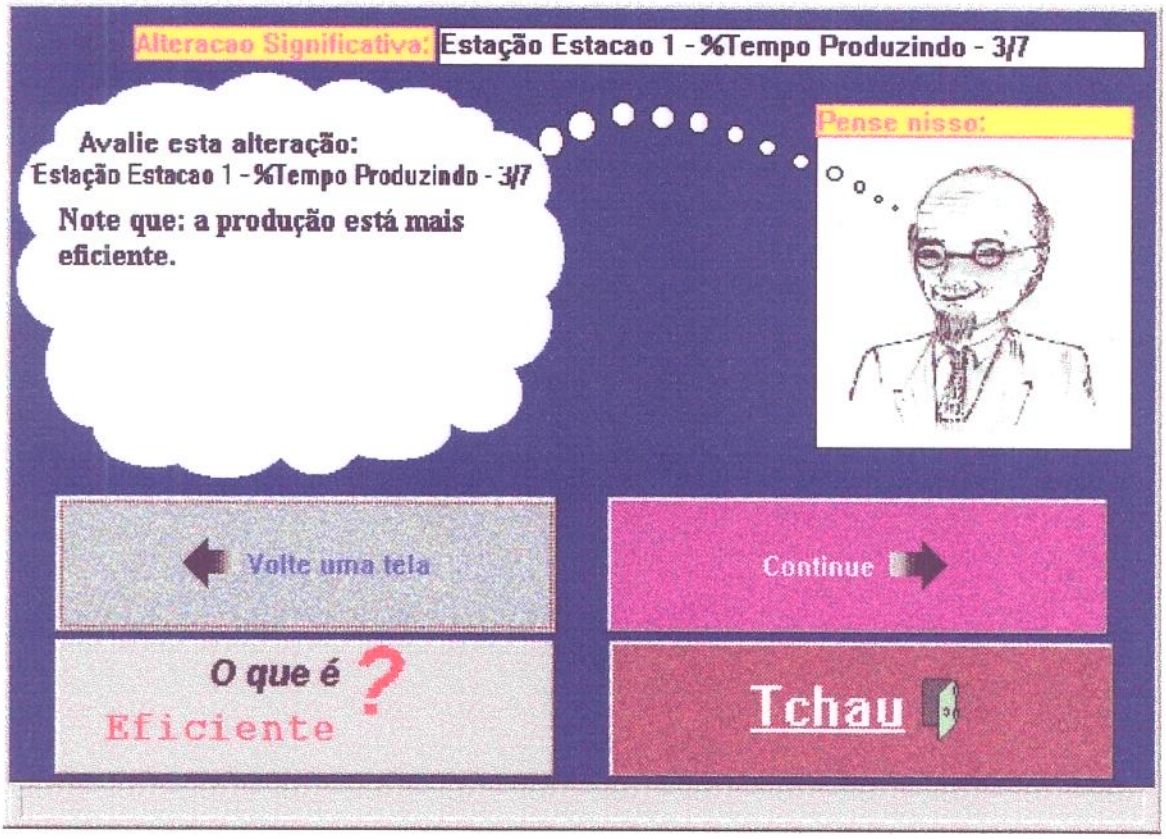


Figura 11: tela "Informações sobre conseqüências de haver mudado o modelo".

Uma vez selecionado o *botão* "Estado atual da fábrica", é apresentada uma tela (figura 12) que possui:

- um desenho com o personagem e informações básicas sobre a tela;
- os botões genéricos anteriormente descritos "O que é?", "Tchau" e "Volte uma tela";
- uma lista com informações de quais resultados da simulação/atributos do modelo devem ser avaliados mais detalhadamente pelos usuários;
- se o usuário não houver selecionado um resultado, estará sendo apresentado o botão "Mudei o mod.: quais foram as conseqüências", que leva ao mesmo ponto do botão homônimo existente na primeira tela;
- se o usuário houver selecionado um resultado, estará sendo apresentado o botão "Analisa o resultado selecionado", que transmitirá informações que têm como objetivo ajudar o usuário

a perceber a importância do resultado que ele selecionou. Esta nova tela possui as mesmas características funcionais da tela de informações descrita anteriormente.

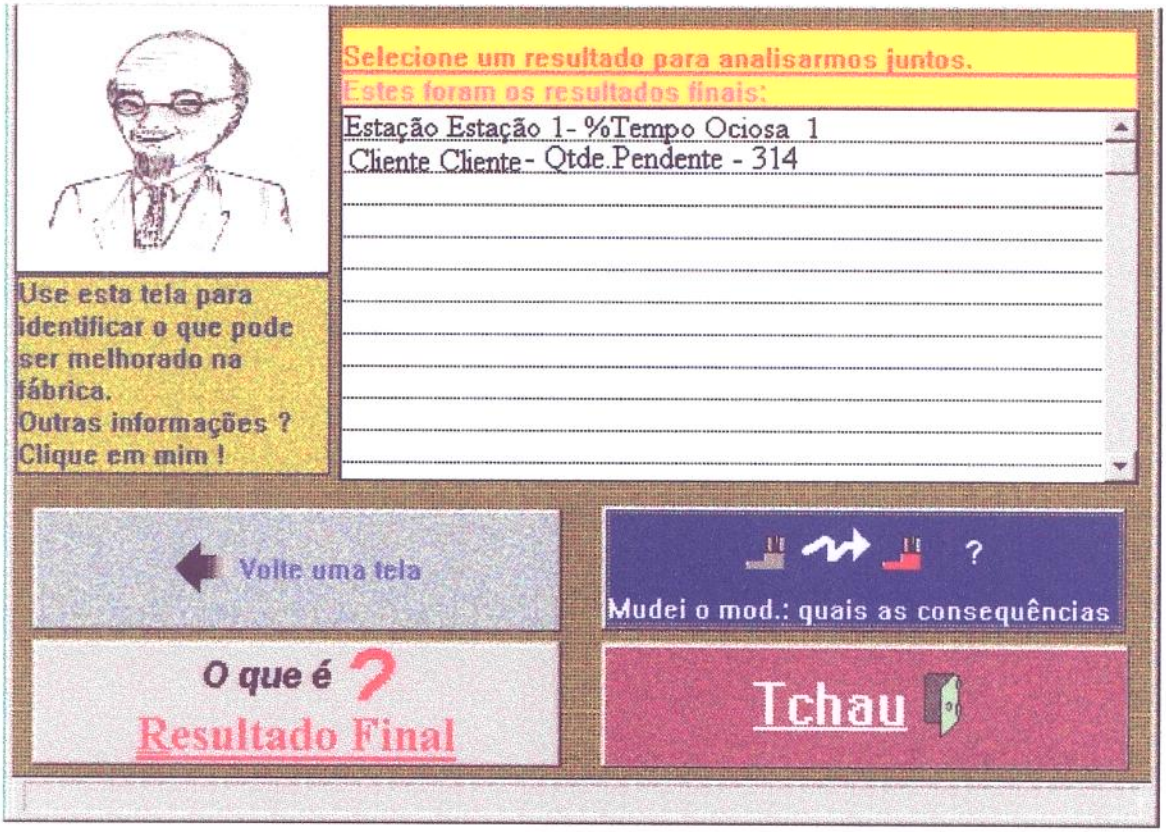


Figura 12: tela "Estado atual da fábrica"

Além das telas descritas, que compõem o corpo principal do sistema, existem ainda duas telas:

- a tela de ajuda, apresentada quando o usuário clica em algum lugar da tela não previsto (figura 13);

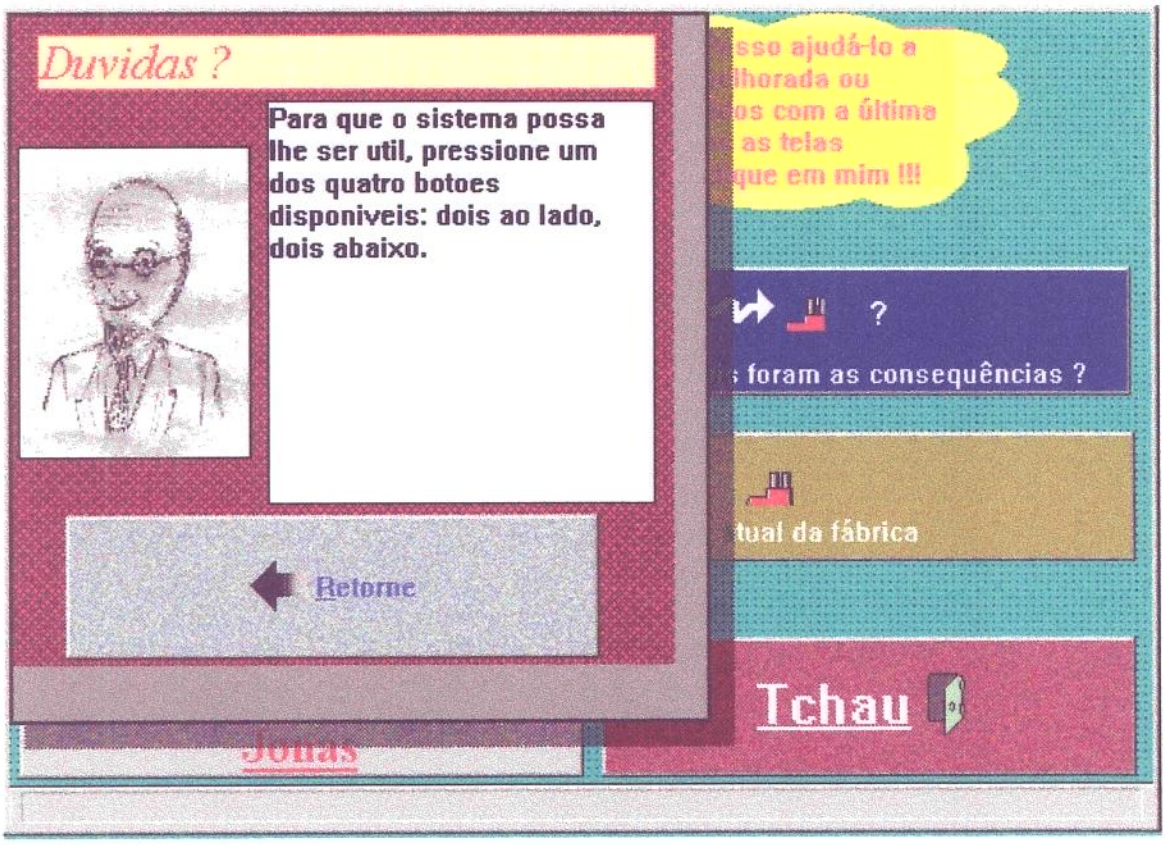


Figura 13: A tela de ajuda

- a tela "O que é?", que explica um termo. É apresentada quando o usuário pressiona o botão de mesmo nome, disponibilizado em todo o sistema para um certo termo definido (figura 14).

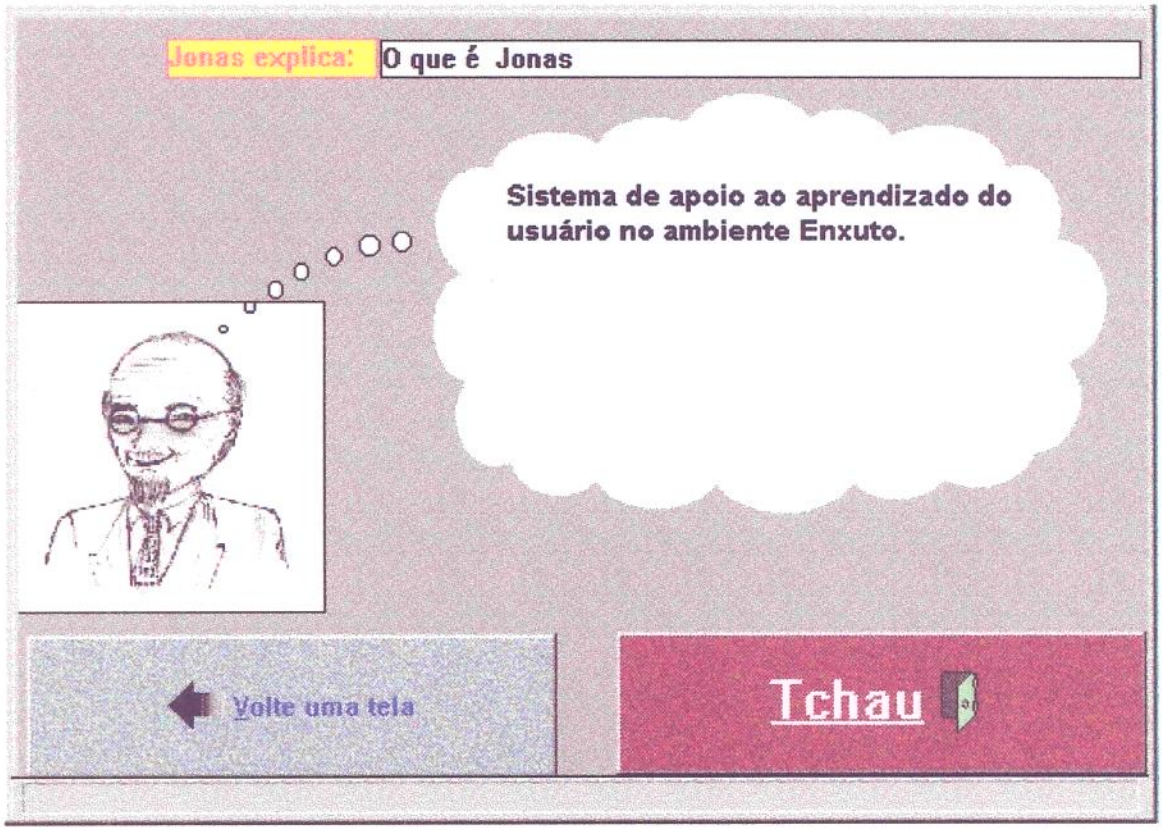


Figura 14: A tela "O que é?"

6.3 - O subsistema administrativo

O subsistema administrativo, de uso restrito ao responsável pelo sistema, funcionará independentemente do modelador/simulador. O "administrador" será alguém responsável por "alimentar" a base de conhecimento com as definições que demonstrem a veracidade dos conceitos a serem trabalhados. Este administrador deverá conhecer toda a fábrica, seu funcionamento e objetivos futuros.

O subsistema administrativo é composto por formulários construídos em Access (Microsoft,1994), que possibilitam ao usuário manter a base de conhecimento sempre atualizada. Buscou-se desenvolver um subsistema simples e amigável, utilizando-se ao máximo dos recursos de interface disponíveis na ferramenta. O subsistema pode ser dividido em três grupos distintos de telas:

- telas com listas de registros de uma certa tabela (veja exemplo na figura 15);
- telas de inserção de novos registros, com campos em branco (figura 16);
- telas de alteração/exclusão de registros cadastrados (figura 17).

Alguns formulários e relatórios do subsistema administrativos são apresentados no (apêndice 4).

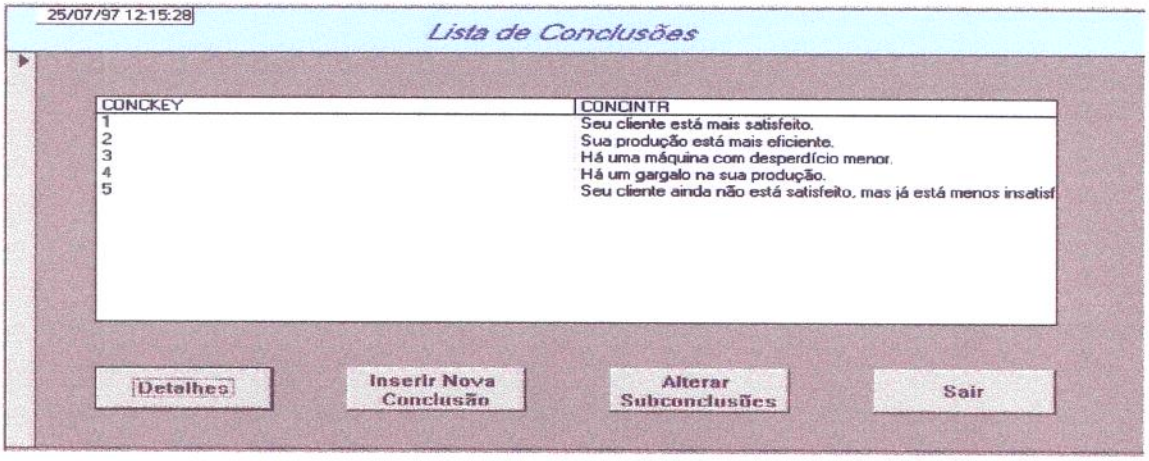


Figura 15: a tela inicial do subsistema administrativo

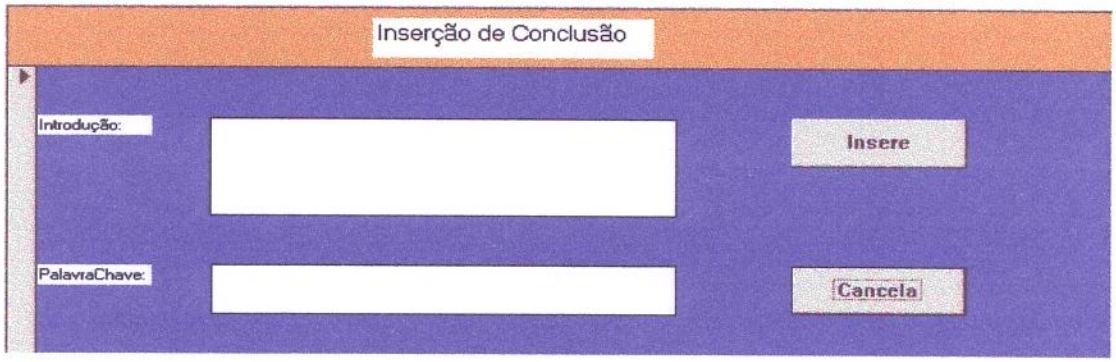


Figura 16: tela de inclusão de nova conclusão

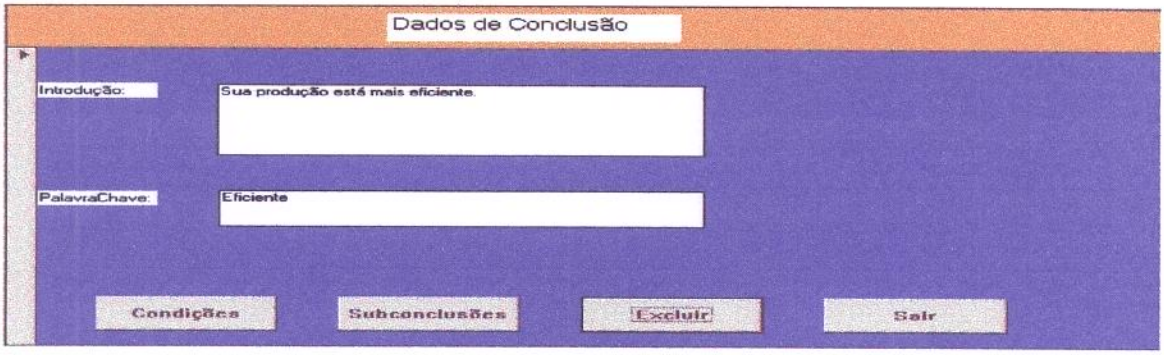


Figura 17: tela de manutenção de conclusão

6.4 - Um exemplo completo

Pode-se imaginar um usuário que tenha modelado uma linha de produção como a apresentada na figura 7, composta por um fornecedor, um cliente, estoques de matéria prima e produto, uma estação de trabalho e os transportes entre estes objetos. Ele executa a simulação do modelo e analisa os resultados dos atributos. Não conseguindo identificar nos atributos se a fábrica está em boa situação ou não, resolve chamar o **Jonas**.

O usuário recebe a tela apresentada na figura 9. Ele deseja verificar o estado atual da fábrica, e, por este motivo, seleciona o botão correspondente. Neste momento é apresentada a tela da figura 12. O usuário analisa as informações na tela e chega à conclusão que deve tentar melhorar a estação. Isto porque o cliente não recebeu muitas peças que foram solicitadas e a máquina está praticamente sem nenhum tempo livre, ou seja, deve estar sendo um gargalo.

Neste momento o usuário poderia pedir dicas para o **Jonas**, mas analisando os resultados ele chegou a sua própria conclusão. Retorna então ao **Enxuto** e analisa os atributos da estação. Identifica que o índice de defeitos da estação está muito elevado e resolve diminuí-lo, de modo a avaliar qual o impacto desta alteração. O usuário executa então uma nova simulação. Ao analisar os atributos resultantes desta simulação, o usuário não consegue identificar quais as diferenças nos resultados existentes entre esta e a simulação anterior. Volta a solicitar a ajuda do **Jonas**, desta vez entrando na primeira opção: "Mudei o modelo: quais foram as conseqüências".

A tela da figura 10 é apresentada. Pode-se perceber que acima está listada a alteração que o usuário efetuou no modelo. Abaixo estão os resultados da simulação que mudaram significativamente. O usuário percorre a lista dos resultados da simulação e seleciona "Estação Estação 1 - %Tempo Produzindo", como apresentado na figura 18. Nesta linha o sistema indica que essa porcentagem aumentou de três para sete entre as duas simulações. O usuário, desejando mais informações a respeito desse resultado, pressiona o botão "Explica a alteração selecionada".

Figura 18: tela "Mudei o modelo...", com alteração selecionada

Neste momento é apresentada a tela da figura 11, com uma informação relacionada com a diferença entre o resultado selecionado pelo usuário nas duas simulações. Se o usuário ainda não estiver satisfeito, ele solicita a opção continue. O sistema **Jonas** apresenta então a figura 19.

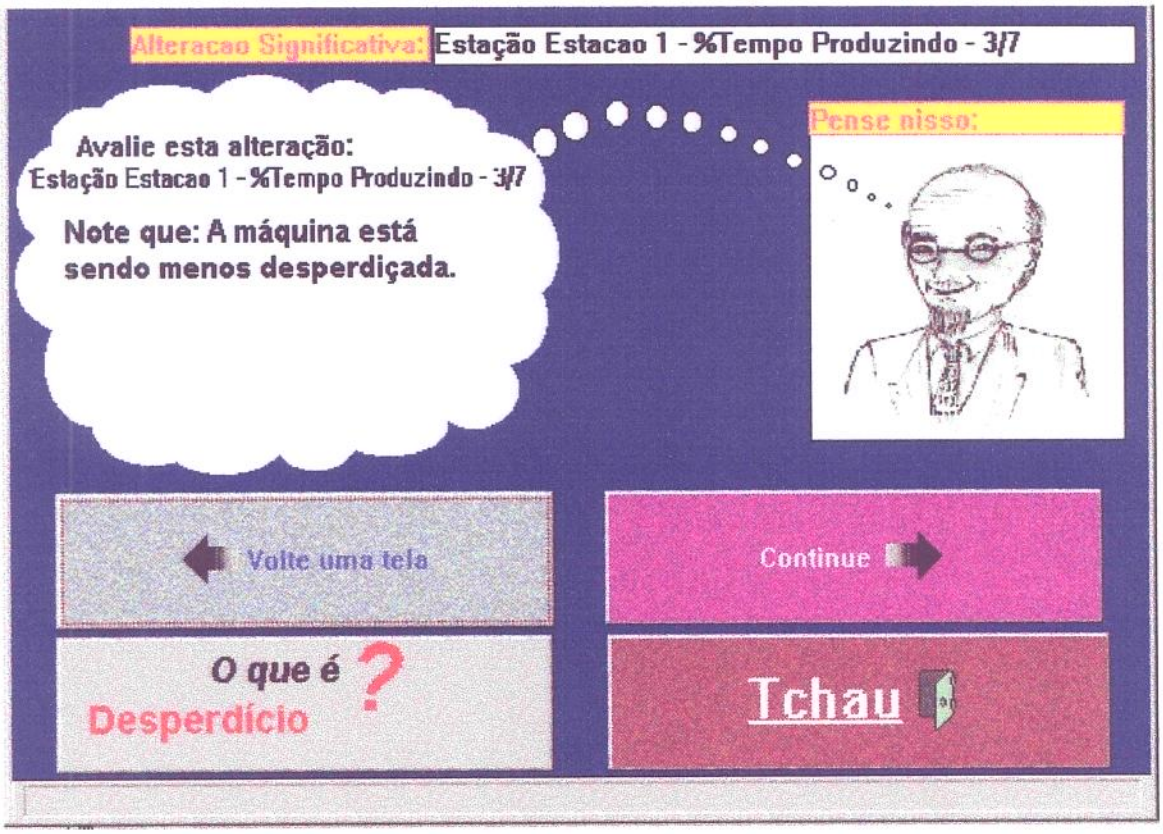


Figura 19: tela "Informações sobre conseqüências de haver mudado o modelo"

Se o usuário desejar informações adicionais, ele pressiona novamente o botão continua. Neste exemplo, o sistema **Jonas** não identificaria mais nenhuma informação relacionada e apresentaria a figura 20, apenas com um lema de qualidade da empresa.

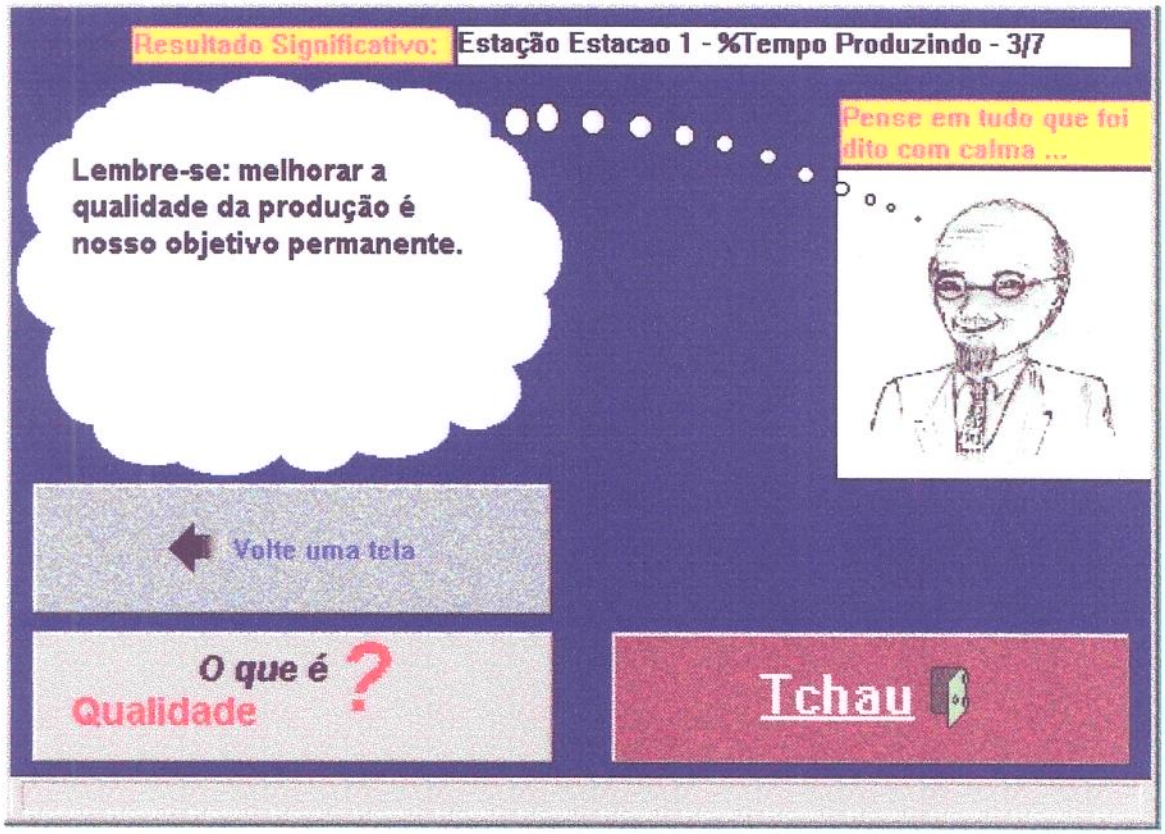


Figura 20: tela de informações com lema geral de qualidade.

7 - Desenvolvimento do sistema *Jonas*

Neste capítulo é descrito o processo de desenvolvimento do sistema **Jonas**.

Segundo a proposta original do projeto, o sistema **Jonas** tem como objetivos ajudar e dar mais poder a seus usuários, explorando as características únicas da mídia computacional. Para isso, o sistema deve ser um *ambiente colaborativo de resolução de problemas* (Fischer, 1995).

Ambientes colaborativos exigem uma interface humano-computador com maiores índices de acessibilidade, facilidade de uso e envolvimento do usuário. No desenvolvimento destes ambientes o conceito de interface inclui não somente *hardware*, *software* e como homens e computadores podem se comunicar, mas também aspectos cognitivos e emocionais das experiências dos usuários (Laurel e Mountford, 1990).

Buscando atingir estes objetivos, a abordagem ao desenvolvimento de um sistema deve ser iniciada pelo *design centrado no usuário*, testando as idéias com usuários em todos os estágios de desenvolvimento (Rheingold, 1990). No projeto do **Jonas**, o envolvimento dos usuários antecedeu o desenvolvimento do primeiro protótipo, em experimentos detalhados na seção 7.1. Ao final dos experimentos, o desenvolvimento da interface estava concluído.

Baseando-se nas funcionalidades definidas durante a construção da interface, foi projetado um banco de dados que atende as necessidades de informação do sistema. Em paralelo à definição deste banco, foram construídas telas de consulta e de atualização dos dados (o subsistema administrativo, detalhado na seção 7.3).

Uma vez definidas interface e base de conhecimento, foi construída a máquina de inferência. Como primeiro passo foram definidos os objetos e seus diagramas de transição de estados, com base na técnica de modelagem de objetos de Rumbaugh et al. (Object Modelling Technique, OMT) (1991). A partir destes diagramas foram definidos os objetos, suas funções e seus atributos. Esta fase do desenvolvimento está detalhada em 7.2.

7.1 - O *design* da interface

O *design* inicial do **Jonas** foi baseado em uma definição prévia de suas funcionalidades. Como o objetivo era construir um sistema com utilização fácil e prazerosa, seria necessário avaliar esta primeira proposta segundo teorias de *design*, de modo a iniciar o processo de desenvolvimento do sistema.

Segundo a teoria da ação (Norman, 1986) a meta de um sistema não deve ser sua eficiência, nem o seu poder computacional, mas o sistema deve transmitir um forte senso de entendimento e controle, com ferramentas que enfatizem conforto, facilidade e prazer em usar: as ferramentas conviviais (Laurel, 1990). Segundo esta teoria, seria interessante mapear as intenções dos usuários nos botões disponíveis, buscando uma imagem do sistema explícita, inteligível e consistente, com uma conceituação coerente que possibilite ao usuário a construção de um modelo mental do **Jonas**. Inicialmente, as variáveis psicológicas envolvidas não eram as dos usuários, mas sim as do seu *designer*. Foi necessário aproximar o sistema dos modelos mentais dos usuários e, neste sentido, a teoria sugere a realização de experimentos.

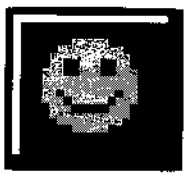
Foram efetuados dois experimentos com usuários típicos no ambiente de fábrica[1]. No primeiro, avaliou-se a proposta inicial do sistema cujo funcionamento foi simulado com pessoas e papel. Se inicialmente fosse usado um protótipo, o experimento poderia ter resultados mais direcionados, podendo não evidenciar as reais necessidades do usuário De posse destes resultados, a proposta inicial foi revista e a partir dela foi construído um protótipo, apenas com as funcionalidades da interface. Este protótipo foi submetido a um outro teste com usuários, que resultou em novas alterações. Após efetuadas estas melhorias, a interface estava construída em sua forma atual.

A seguir, são detalhados, os experimentos e as alterações a que a interface foi submetida.

[1] Operários e engenheiros da linha de produção da Harrison Thermal Systems

7.1.1 - A proposta inicial

Na proposta inicial o sistema seria acessado pelo usuário via barra de menu do **Enxuto** ou via um ícone (figura 21), que ficaria em alguma posição da barra lateral da janela. Uma vez acessado o **Jonas**, imediatamente seria apresentada uma janela com botões para que o usuário definisse se ele desejaria uma explicação dos resultados da simulação ou uma indicação de possível melhoria no modelo (figura 22). Após o usuário selecionar o que ele desejasse, o sistema apresentaria uma tela com uma idéia inicial da informação e botões para que o usuário controlasse o fluxo de diálogo com o sistema (figura 23). Os botões que estariam disponíveis seriam:



Jonas

Figura 21: Ícone de acesso ao Jonas

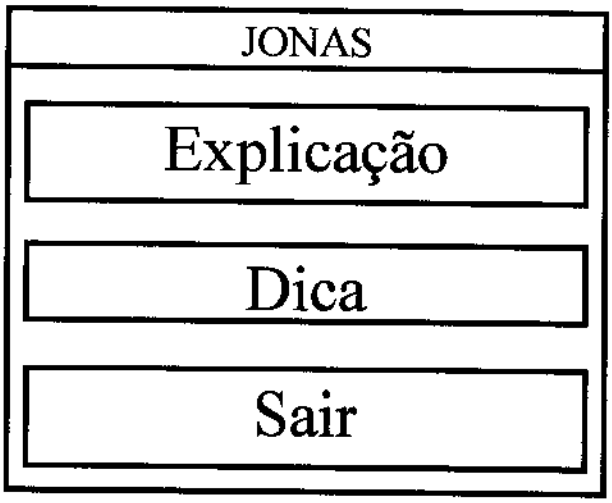


Figura 22: Janela do Jonas

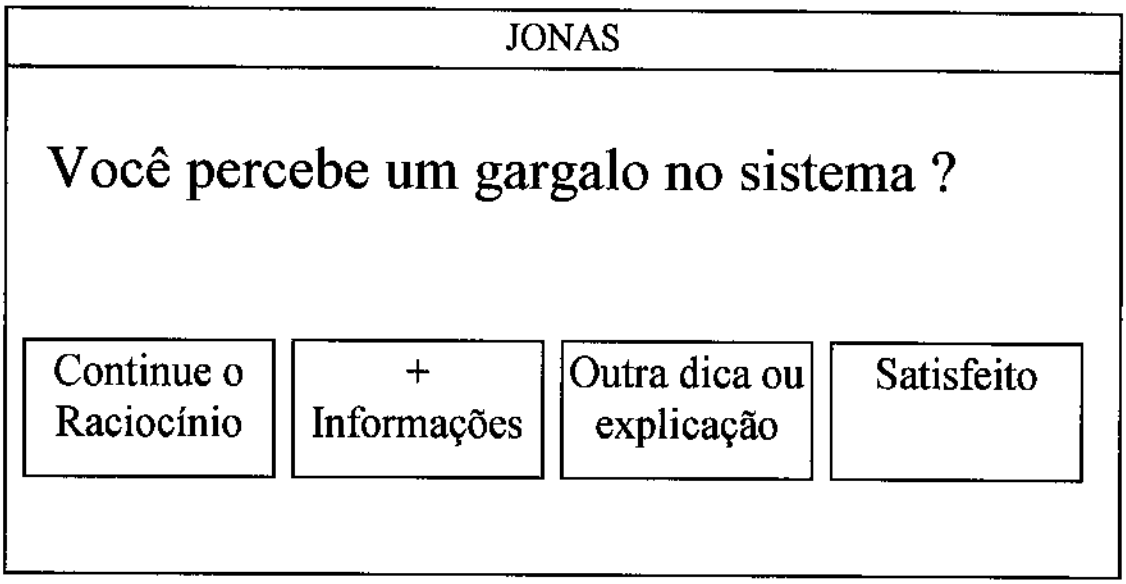


Figura 23: Tela principal de diálogo do Jonas

- continue raciocínio: onde seria apresentado um texto que "aprofundasse o raciocínio" seguido até aquele momento pelo sistema;
- mais informações: onde seria explicado mais claramente o que o sistema "quis dizer" com a frase anterior, sem aprofundar o "raciocínio";
- outra dica ou explicação: faria o **Jonas** buscar uma outra linha de "raciocínio", ignorando o anterior;
- satisfeito: fecharia a janela do **Jonas** e voltaria o controle para o **Enxuto**.

Conjuntamente com a definição da interface, foi desenhado um modelo de base de conhecimento que, pelo menos em teoria, conseguia atender as necessidades desta interface. O modelo da base não só foi influenciado pela interface, como teve influência na definição desta.

7.1.2 - Experimento I: avaliação da proposta inicial

Objetivos

De modo a tornar o sistema **Jonas** mais próximo do usuário, com base nos pressupostos teóricos de Norman (1986), foram efetuados experimentos com usuários típicos, seguindo a metodologia da Apple (Gomol, 1990). O objetivo do experimento I foi identificar as necessidades e expectativas dos futuros usuários do sistema, validando a proposta inicial.

Metodologia

Foram construídas três situações simplificadas do modelador/simulador do **Enxuto**. Estas situações estavam representadas em uma folha onde foram impressas duas *telas* simplificadas apresentando os resultados da simulação antes e depois de uma certa alteração efetuada no modelo, alteração esta indicada ao usuário na mesma folha. Para cada tela também foi apresentada uma lista dos valores finais dos atributos em cada uma das simulações. A figura 24 apresenta uma destas situações (todas estão apresentadas no apêndice 1).

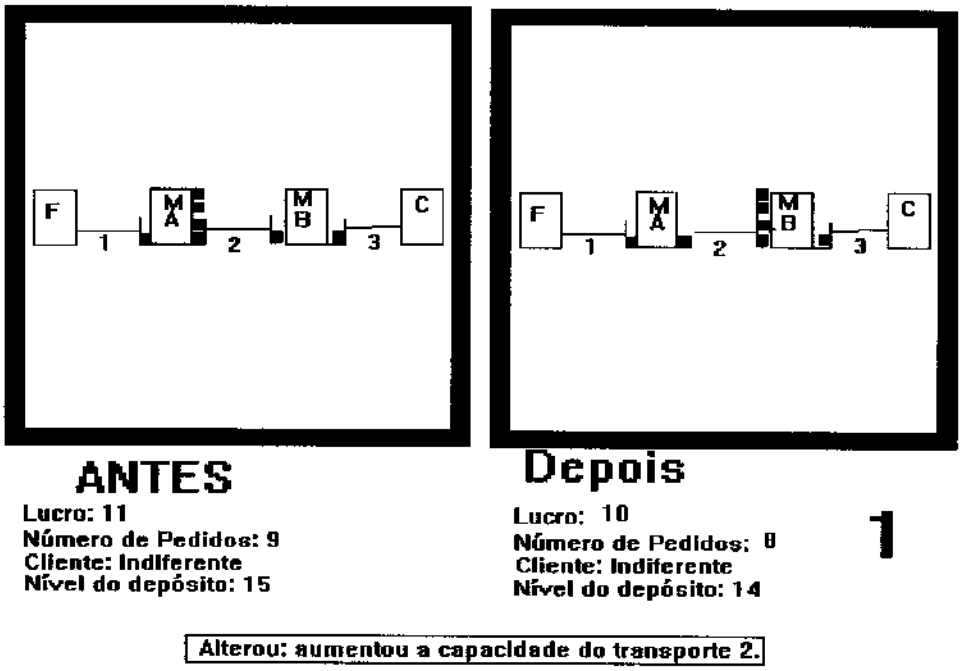


Figura 24: Situação apresentada aos usuários no primeiro experimento.

Três duplas, duas compostas por funcionários do chão-de-fábrica e uma composta por engenheiros de produção (com um conhecimento mais profundo dos conceitos envolvidos na produção enxuta), participaram da observação.

As duplas de funcionários do chão de fábrica tinham como *tarefas*:

- explicar porque a alteração indicada causou as diferenças observadas nos resultados da simulação;
- indicar como conseguir melhores resultados alterando o segundo modelo.

As respostas da dupla de especialistas foram utilizadas para se identificar como eles chegam às conclusões, de modo que, ao se projetar o **Jonas**, seu método de inferência se aproximasse ao utilizado pelos especialistas. Outro objetivo para esta dupla era interagir com uma dupla de funcionários do chão-de-fábrica para que se pudesse verificar como ocorre a interação entre funcionários típicos e especialistas na busca da melhoria do modelo, interação

esta semelhante ao almejado com o **Jonas**. Nicol (1990), defende que o primeiro passo de um *design* deve ser verificar como bons professores ensinam. Neste caso, são os engenheiros os responsáveis por ensinar os conteúdos para os funcionários.

A dupla que interagiu com os especialistas foi encarregada também de definir o tipo de acesso do sistema **Jonas** a partir do **Enxuto**.

A outra dupla de funcionários do chão-de-fábrica interagiu com um sistema simulado, seguindo a técnica de Vertelney e Booker (1990). O objetivo desta parte do experimento foi verificar como usuários se comportariam em frente ao sistema e identificar quais características da interface e do funcionamento do sistema proposto poderiam ser melhoradas.

Antes de ser executado o experimento, foram identificados e descritos os dez passos a serem seguidos, segundo Gomol (1990). Estes passos foram relacionados na forma a seguir:

1) Serão 3 duplas, duas de funcionários que serão usuários típicos e uma formada por especialistas nas técnicas de produção (da própria GMB - General Motors do Brasil). Para cada dupla haverá instruções escritas. O experimento será realizado no ambiente de trabalho, num local não sujeito a interrupção. Será utilizado algum método de gravação, preferencialmente vídeo, pelo menos áudio;

2) Indicar que o teste é do produto e não das pessoas que o farão. Exibir uma idéia geral da observação (testar uma proposta para a interface do **Jonas**);

3) Indicar que eles podem desistir a qualquer momento;

4) Apresentar o que está na sala: equipamentos para gravação e computador simulado;

5) Pedir para que se pense alto, pois todo pensamento é importante;

6) Explicar que durante o experimento não haverá ajuda e que todas as dúvidas serão sanadas ao final do teste;

7) Explicar o que é o Jonas. Explicar os passos que devem ser seguidos (separadamente por dupla). Entregar instruções escritas;

8) Perguntar se não há mais questões. Iniciar o exercício;

9) Concluir as observações:

explicar os objetivos da observação;

responder qualquer questão;

discutir qualquer coisa interessante que foi percebida, pedindo explicações;

perguntar qual é a impressão geral;

10) Avaliar os resultados obtidos.

As instruções escritas do item 7 são apresentadas no apêndice 1.

Os experimentos foram gravados em vídeo e efetuados em uma sala da própria GMB, próxima a área de produção, onde, provavelmente, será instalado o ambiente **Enxuto**.

Resultados e nova proposta

Durante os experimentos ficou claro que as opções oferecidas na tela inicial (explicação, dica) não tinham seu significado claro para os usuários. Optou-se, então, por frases mais explicativas a serem colocadas nos botões: "Não entendi o que aconteceu" substituiu "explica" e "Quero melhorar a fábrica" substituiu "dica". Nesta primeira tela também inseriu-se uma opção de ajuda. Tal opção foi denominada "O que é Jonas?" (figura 25).

A chamada ao sistema Jonas passou a ser um ícone (mais precisamente um capacete de segurança com um ponto de interrogação, figura 24) que fica junto ao modelo. Isto foi definido a partir de uma afirmação de uma das duplas: "*se o sistema vai ser chamado para ajudar a resolver um problema na fábrica, ele deve ser um elemento da fábrica, que fica lá conosco observando*".

Uma dificuldade evidente para os usuários durante os experimentos foi relacionar qual variação no resultado da simulação foi causada por qual alteração efetuada no modelo. Outra dificuldade foi identificar quais resultados poderiam ser melhorados com alguma mudança no modelo. Esta dificuldade, não prevista na proposta inicial, ficaria mais evidente quando os usuários estivessem frente ao ambiente **Enxuto** real, uma vez que os resultados obtidos com a simulação não estariam impressos, dificultando a comparação e análise. Como fazer, então, para viabilizar esta comparação? Foi definido que o **Jonas** colaboraria neste sentido.

Na explicação dos resultados de uma alteração, a primeira tela apresenta as alterações efetuadas nos modelos e os resultados obtidos da simulação antes e depois destas alterações. Só são apresentadas as "alterações significativas", ou seja, aquelas alterações que não podem ter sido causadas por pequenas variações aleatórias nos resultados do modelo. O usuário pede então para que o **Jonas** explique uma certa variação, por ele selecionada (figura 26).

Na busca de possíveis melhorias, a primeira tela apresenta os resultados da simulação que requerem uma avaliação mais cuidadosa. O usuário seleciona o resultado que ele deseja ver analisado pelo **Jonas** (figura 27).

A função do botão "+ informações", utilizado para explicar melhor algum conceito importante da frase anterior não foi compreendida durante o experimento. Resolveu-se trocar sua identificação para "o que é ...", indicando assim o que será explicado. Deste modo acredita-se que a opção tornou-se mais clara. Outra vantagem é que o usuário irá requisitá-la quando realmente desejar a descrição de determinado conceito (figura 28).

Com base nas avaliações resultantes do primeiro experimento foi definida uma nova proposta. Contrariamente ao que havia sido feito anteriormente, esta proposta não foi desenvolvida em paralelo à base de conhecimento. Segundo a abordagem adotada (Norman,1986), o desenvolvimento de um sistema deve começar pela sua interface para então se definir suas funcionalidades.

De modo a tornar o sistema mais simples e agradável de se utilizar, foi definido um tipo de tela padrão, com opções de nomes mais claros. Um exemplo foi a inserção do botão "Tchau!" em todas as telas, substituindo "Sair" e "Satisfeito". Foram colocadas também figuras nos botões (além de textos), de modo a tornar sua identificação mais rápida e a tela mais agradável para se trabalhar. Outras figuras e frases informais foram inseridas para "aproximar" o sistema ao usuário.

A possibilidade de reverter uma ação, indicada como sendo uma boa característica de sistemas de manipulação direta, não era prevista na proposta inicial. Nesta nova proposta, em todas as telas existe um botão chamado "quero voltar uma tela".

Na nova proposta de interface, baseada nos resultados do experimento, o usuário passou a ter maior controle. Na versão inicial, era apresentada uma informação definida pelo sistema e o usuário poderia pedir outra (sem definir qual); na versão atual, é o próprio usuário quem define qual informação deseja ver. Outra característica positiva desta nova interface é que ela aumenta o poder de análise do funcionário, fornecendo lado a lado os resultados das duas últimas simulações (exercitar a análise de situações é um dos objetivos do **Enxuto**).

Figura 24: Ícone de acesso ao Jonas

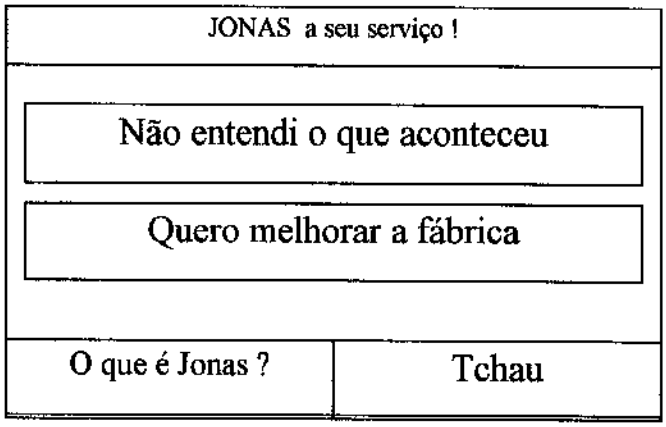


Figura 25: Janela do Jonas

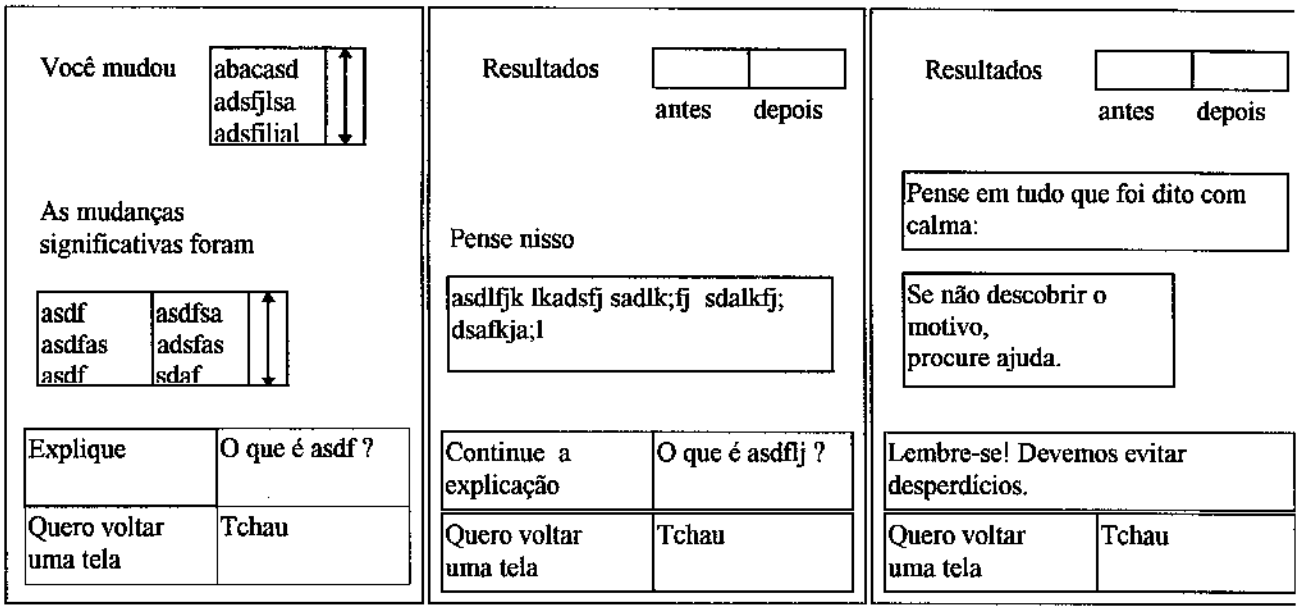


Figura 26: Telas de explicação

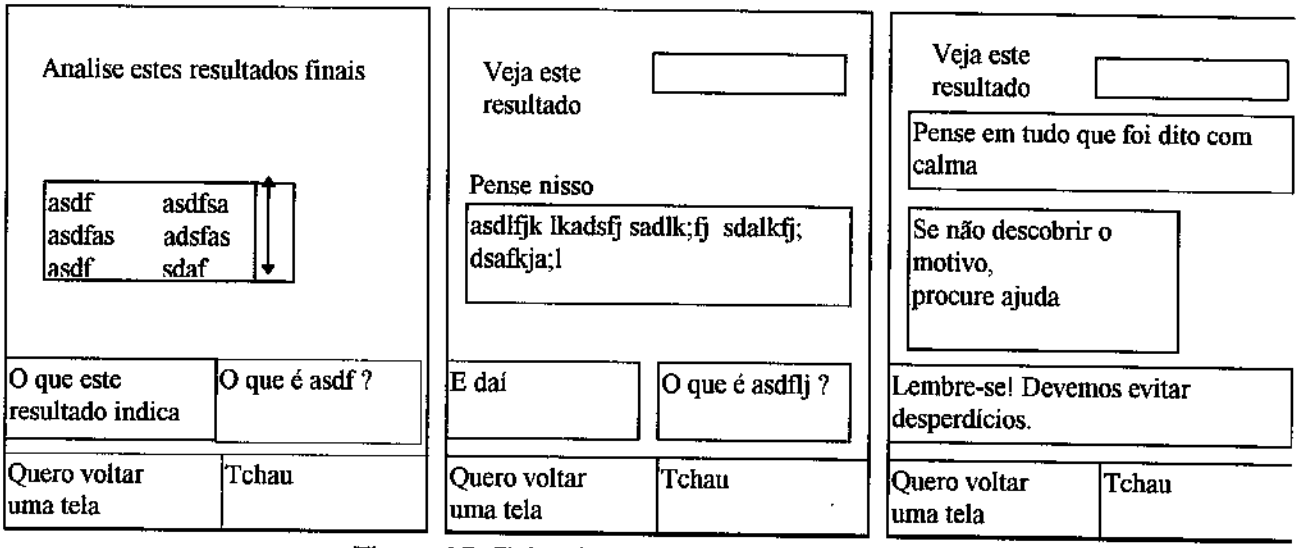


Figura 27: Telas de possíveis melhorias

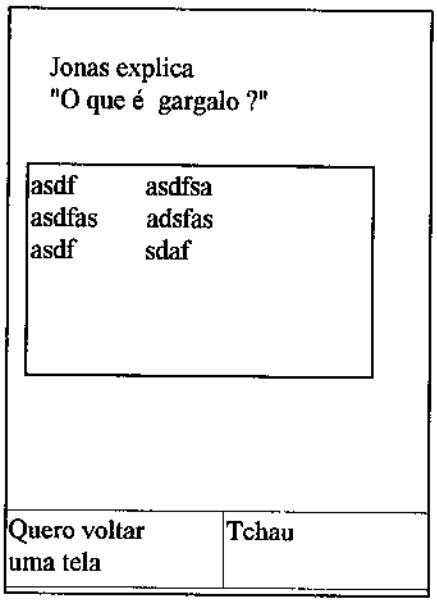


Figura 28: tela de explicação de termos

7.1.3 - Experimento II: avaliação da interface construída

Objetivos

Foi efetuado um segundo teste da interface do **Jonas** com usuários típicos seguindo as mesmas estratégias utilizadas no teste anterior. Diferentemente do primeiro, neste, os usuários utilizaram-se de uma interface real construída em computador, que simulava o funcionamento do sistema tal como será (protótipo). O objetivo principal do experimento II foi validar a interface construída, identificando onde ela poderia ser melhorada e como os usuários reagiam ao sistema computacional.

Metodologia

Para este teste, além do computador com o protótipo, foram utilizadas duas folhas onde foram impressas telas do sistema **Enxuto** com modelos de processos de manufatura (vide apêndice 2). Além disso, estas folhas apresentavam:

- na primeira, os valores dos atributos dos objetos do modelo e os resultados da simulação com estes atributos;
- na segunda, os valores dos atributos que foram alterados após a primeira simulação e os novos resultados obtidos.

Também foi utilizada uma câmera de vídeo com a qual registrou-se o experimento.

Participaram do teste dois usuários da área de manufatura. Para eles foram passadas como tarefas:

- explicar como as mudanças efetuadas no modelo após a primeira simulação originaram as alterações observadas nos resultados da segunda simulação;
- identificar possíveis melhorias que poderiam ser feitas no modelo da segunda página.

Para efetuar a tarefa, os usuários deveriam utilizar as folhas com os modelos e os valores de seus atributos e o sistema **Jonas**. A interface deste havia sido construída de modo a

apresentar as sugestões como realmente o faria se o sistema estivesse pronto, com o usuário modelando e simulando conforme o que estava indicado nas folhas.

Resultados

Avaliando o vídeo resultante do experimento foi possível tecer as considerações abaixo que algumas vezes levaram, outras não, a alterações na interface do sistema.

Houve dificuldade em se perceber a diferença entre as janelas de "explicação sobre mudanças no modelo" e "avaliação do estado atual da fábrica". Ao entrar na primeira, os usuários só saíram dela com a intervenção do pesquisador. Eles não saíam da primeira, apesar de saber fazê-lo, porque não percebiam que a outra tela poderia auxiliá-los melhor em certos momentos. A partir desta observação, foram inseridos textos explicativos nas duas telas, incentivando os usuários a entrar na outra quando for conveniente. Também foram criados botões que levam diretamente de uma destas janelas para a outra.

Com relação aos textos apresentados, houve problemas com o entendimento de certos termos. Vários destes termos são utilizados pelo **Enxuto** e, por este motivo, foi sugerido ao responsável pelo sistema de modelagem e simulação que os alterasse. Os termos que apresentaram dificuldade de interpretação foram:

- diferenciar "Produto" (material produzido, armazenado depois de uma máquina aguardando o transporte) de "Produzido" (quantidade de material produzido pela máquina durante toda uma simulação). Após analisar o experimento, foi sugerida a utilização dos termos "Aguardando Transporte" para substituir "Produto" e "Total Produzido" para substituir "Produzido";
- o termo "Nível Crítico" não é utilizado na fábrica. Após explicar o que significa (quantidade mínima aceitável de material em estoque) eles sugeriram mudar o termo para "Estoque mínimo" ou "Estoque de Segurança". Também demonstraram interesse em que seja anexado ao termo se ele se refere a peças produzidas ou matéria-prima.

Por fim, com relação a formatação das telas, surgiram dois problemas que já foram solucionados:

- nas telas com explicações, as informações apresentadas no campo localizado na extremidade superior não foram percebidas. Os usuários chegaram a se queixar da ausência destes dados na tela (apesar de eles estarem nela). Decidiu-se, a partir de sugestão deles, incluir esta informação também no *balão* onde é apresentado o texto principal (figura 11);
- no menu de alterações, na tela de alterações no modelo, os usuários sentiram falta de um título que indicasse que os dados apresentados (um número seguido de uma barra e de outro número) significavam o resultado da primeira e da segunda simulação, apesar de o texto imediatamente abaixo do menu explicar isto. Foi decidido incluir no título do menu a indicação "antes/depois" (figura 10).

Com exceção do acima indicado, a interface e as informações nela contidas atingiram os objetivos de clareza definidos durante seu *design*. Quando clicado um local inválido, os usuários conseguiram interpretar a janela informativa apresentada. Quando o significado de um termo não era claro (por exemplo, "alteração significativa"), utilizaram o botão "o que é?". Quando o botão "Continue" não foi apresentado, perceberam que tratava-se de uma explicação terminada, sem possível continuação. Também não houve dificuldade no entendimento do funcionamento dos outros botões.

Com relação ao ambiente, percebeu-se um interesse bastante expressivo com relação ao **Enxuto**, especialmente nas fases de modelagem e simulação. Durante o experimento os usuários indagaram se era possível construir um modelo da realidade deles e simular. Eles também indicaram que o sistema **Jonas** deveria ser chamado a partir de um ícone na tela do Enxuto, conforme já havia sido observado no primeiro experimento. Esta ligação já estava sendo planejada e não havia sido implementada naquela data por problemas de compatibilidade entre versões do ToolBook utilizadas na época na construção do **Enxuto** e do **Jonas** (Asymetrix, 1994, 1995). Tal problema foi solucionado com a conversão do **Enxuto** para a nova versão do ToolBook.

O modo como os usuários trabalharam demonstrou ser bem próximo do almejado com o sistema. Eles utilizaram o **Jonas** para identificar em que ponto deveriam focar a análise e, então, analisaram as folhas para entender o que havia ocorrido. Temia-se que os usuários utilizassem o **Jonas** como único meio de obter informações, e não apenas como um

conselheiro, mas isto não aconteceu. O sistema **Jonas** funcionou realmente como um conselheiro (ou consultor) indicando que aspectos do modelo deveriam ser analisados de um modo mais cuidadoso. Os usuários conferiam os resultados apresentados pelo **Jonas,** comparando-os ao modelo, e tentavam *traduzir* aquelas informações que o sistema apresentava em dados úteis para a realização das tarefas a eles indicadas. Ou seja, percebeu-se que os usuários utilizavam o modelo como ponto intermediário entre as informações do **Jonas** e a compreensão real destas informações. Demonstraram, inclusive, interesse em que o **Jonas** indicasse diretamente na tela do **Enxuto** onde as alterações apresentadas estão, facilitando esta transposição. Esta sugestão parece interessante e poderá ser utilizada numa extensão futura do sistema **Jonas**, uma vez que diminui o "Golfo da Avaliação" (Norman, 1986). O fato desta compreensão ser baseada na análise do modelo e não apenas nas informações fornecidas pelo **Jonas** está de acordo com a abordagem construcionista de aprendizagem utilizando modelagem e simulação.

Após obter a lista de informações apresentadas pelo sistema e entender seu significado analisando as folhas com os modelos, os usuários identificavam se aquela informação indicava uma mudança que representava uma melhoria ou uma piora na fábrica simulada. Para os usuários apenas as alterações que indicavam uma piora nos resultados mereciam ser avaliadas. Eles sugeriram que fosse indicado na lista o que é ruim, ordenando-se do pior para o melhor. Esta sugestão não foi acolhida, pois contraria a abordagem educacional adotada, onde esta análise e classificação, se feita pelo usuário, pode levar a uma reflexão sobre o modelo como um todo. Do modo sugerido, o **Jonas** funcionaria como um professor instrucionista indicando o que é bom e o que não é. A princípio, esta busca por melhorias e não pelo entendimento parece ser uma postura absorvida da cultura educacional de formação e da empresa em particular. Uma análise mais cuidadosa destes fatores é interessante, mas foge do escopo do teste.

Ainda nesta linha, outro pedido dos usuários, que não foi acatado, foi o de apresentar sugestões de melhoria para cada problema apresentado. Neste caso, também, o sistema funcionaria de um modo instrucionista, ensinando o que fazer em cada situação. O que se busca é que os próprios usuários, identificando um problema, discutam entre si e definam uma

alteração que lhes pareça conveniente. Após esta definição, testem esta alteração no **Enxuto** e, então, verifiquem se a alteração efetuada efetivamente melhorou os resultados. Deste modo os usuários estarão trabalhando segundo a *estética Logo* (Valente, 1993B), sendo o computador apenas uma ferramenta onde eles representam, executam e avaliam seus modelos, construindo conhecimento a partir da verificação da correção de certas suposições.

A partir de discussões efetuadas durante o experimento, os usuários escreveram explicações para as situações-problema propostas nas folhas-tarefa (apêndice 2). Pode-se perceber que os usuários utilizaram-se de informações obtidas a partir do **Jonas** nestas explicações, demonstrando entender estas informações e relacioná-las exatamente com partes do modelo (esta relação não é fornecida explicitamente pelo sistema). Por exemplo, o sistema apresentou a informação "Há um desperdício de espaço na fábrica", mas não apontou onde esse desperdício ocorria. Posteriormente ao recebimento da informação acima, os usuários justificaram uma alteração no modelo com a diminuição do desperdício de espaço. Isto parece indicar que o sistema será realmente producente na sua função de ajudar a construir o conhecimento sobre o contexto em questão. Os usuários também utilizaram em suas explicações informações relativas a realidade da fábrica, não apresentadas pelo sistema. Isto é muito interessante porque demonstra que os usuários não perceberam o sistema como a única fonte de conhecimento, mas como mais uma, incorporando o **Jonas** ao seu domínio de trabalho. Nesse sentido, fica visível no uso do sistema o paradigma do aumento de inteligência.

7.1.4 - Conclusões gerais

O primeiro experimento foi muito importante pois, como pode ser observado, o sistema proposto inicialmente não atendia às necessidades dos futuros usuários e suas potencialidades não eram totalmente compreendidas. A avaliação dos resultados não levou apenas a uma mudança na posição dos campos, mas também a uma mudança no comportamento geral do sistema, em sua funcionalidade. Acredita-se que o *design* da interface decorrente desta avaliação esteja mais próximo dos modelos mentais do usuário.

A partir das observações feitas no primeiro experimento, foi construído o protótipo do sistema. Avaliar esta interface também foi muito importante, porque:

- usuários de um sistema em computador possuem uma postura diferenciada daqueles que interagem com um computador simulado por pessoas. Isto leva a impressões sobre o sistema e dúvidas sobre o funcionamento diferentes. Pode-se perceber que as duplas que trabalharam com o computador utilizaram-se mais das ajudas disponíveis e sentiram-se mais "à *vontade*" no experimento;
- certas opções de *design*, aparentemente não significativas, influenciam bastante nos resultados obtidos pelo usuário na utilização do sistema. O segundo experimento apresentou alguns exemplos interessantes neste sentido, como a não percepção, por parte dos usuários, de uma informação apresentada na tela.

Os experimentos foram efetuados no mesmo ambiente onde será instalado o sistema: uma sala ao lado da linha de produção. Apesar de esta localização não possibilitar o isolamento sugerido por Gomol, acredita-se que deste modo os resultados obtidos podem oferecer maiores subsídios para uma avaliação de como será a utilização real do sistema. Quando o sistema estiver instalado, seus usuários certamente serão interrompidos quando houver alguma *emergência* na fábrica. Estas interrupções influenciarão no desempenho do usuário frente ao computador. Ao se desenvolver os testes no mesmo ambiente, houve a possibilidade de se avaliar a interação do usuário com o sistema sujeita às interrupções normais.

Os resultados obtidos com o segundo experimento devem considerar o fato de os usuários que participaram destes testes não serem operários, mas sim engenheiros do chão-de-fábrica. Apesar de os engenheiros apresentarem um maior conhecimento do processo de manufatura como um todo, acredita-se que o modo de trabalhar com o sistema não tenha sido muito diferente do que ocorreria com funcionários menos qualificados. Possivelmente, o maior conhecimento fez com que eles dispendessem menos tempo para tomar atitudes semelhantes às que tomariam os operários. Por outro lado, uma vez que os engenheiros estão mais habituados a trabalhar com computadores, conseguiu-se avaliar as dificuldades decorrentes do sistema, e não decorrentes de uma falta de experiência em informática.

7.2 - O sistema especialista

7.2.1 - A base de conhecimento

Segundo a proposta do **Jonas**, sua base de conhecimento (BC) deve ser simples de manipular, viabilizando sua alteração e complementação pelos próprios usuários no ambiente fabril. Este objetivo foi definido porque o poder de um SE depende primeiramente de sua base de conhecimento, que deve ser completa e flexível (Luger e Stubblefield, 1989). Para que isto seja viável, a base não pode ser muito complexa e deve haver ferramentas de manipulação simples, identificadas no projeto do **Jonas** como o subsistema administrativo (descrito em 7.3).

Buscando atender a estas necessidades, foi desenhado um projeto de BC mantida em um banco de dados (BD) relacional, do qual derivou a primeira proposta de interface e funcionalidades do **Jonas**. Após a reavaliação da interface com os experimentos descritos em 7.1, o projeto inicial da base foi refeito, também seguindo as estratégias do design centrado no usuário.

A BC gerada pode ser classificada em três grandes grupos, de acordo com suas funcionalidades:

- uma lista de *lemas de qualidade*, que são apresentados aos usuários ao final de um raciocínio, como um fechamento. É apenas uma lista de frases, sem nenhuma chave;
- os significados de algumas expressões importantes, que podem ser solicitados quando os usuários não entendem estas expressões. É constituído por uma tabela com dois campos: expressão e sua respectiva explicação;
- o grupo que caracteriza o BD como uma BC abdutiva, ou seja, uma base que possibilita identificar uma "causa" a partir de certos "sintomas" (Luger e Stubblefield, 1989). A "causa" está indicada numa conclusão e os "sintomas" são representados em uma lista de condições necessárias para que se chegue a uma conclusão. Cada condição identifica um tipo de objeto e sua posição no modelo, podendo estar relacionada com zero ou mais comparações. Nas comparações os atributos podem ser avaliados segundo a tendência (de

crescimento ou decrescimento, comparando o resultado atual com o anterior), ou confrontados com outros valores (fixos ou de outros atributos do modelo). A figura 29 apresenta a estrutura das tabelas envolvidas nesta funcionalidade e a figura 30 apresenta um exemplo de conclusão.

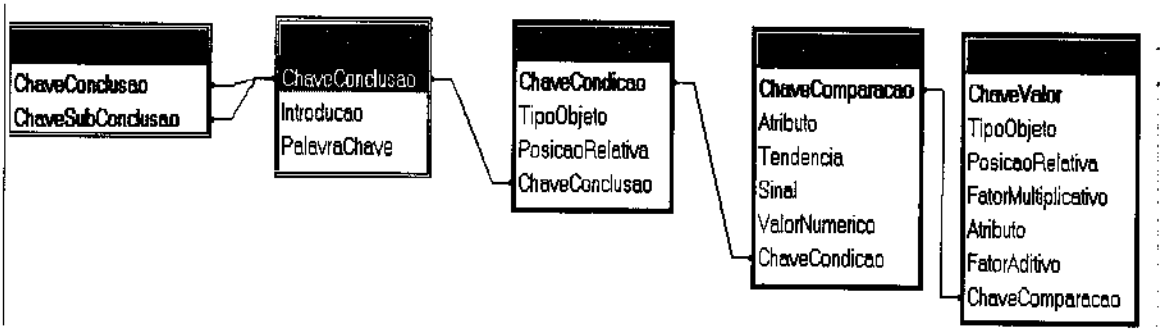


Figura 29: tabelas que originam a base de conhecimento

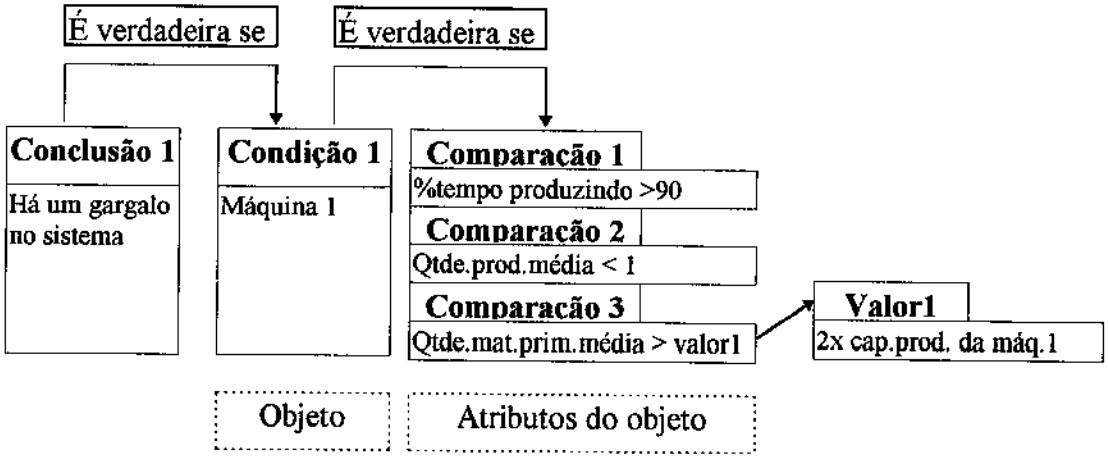


Figura 30: um exemplo de conclusão

7.2.2 - A máquina de inferência e as estruturas de objetos

A proposta do **Jonas** definia que sua máquina de inferência seria baseada numa arquitetura de sistema OO. Buscando utilizar ao máximo as potencialidades da OO, definiu-se que a BC seria convertida de arquivos relacionais em uma estrutura de objetos em memória. Esta é a primeira estrutura de objetos: objetos que representam a BC, criados automaticamente no momento em que o **Enxuto** é carregado. O objeto *BD* é o principal, sendo o responsável por criar toda a base a partir do banco de dados. Isto é feito quando o objeto *BD* é criado. O *BD* possui apontadores para três objetos controladores de listas: lemas, ajudas e conclusões.

A lista de lemas simplesmente armazena os lemas existentes no banco de dados em seqüência e, quando solicitada, retorna um deles. A lista de ajudas armazena uma seqüência de palavras e explicações. Retorna à explicação relacionada com a palavra solicitada. A lista de conclusões controla uma estrutura semelhante àquela apresentada na figura 29. Ela recebe informações sobre o estado atual do modelo e identifica quais *conclusões* podem ser obtidas a partir destes dados. São estas *conclusões* que são apresentadas ao usuário, quando este solicita informações.

Uma biblioteca de ligação dinâmica (Dynamic Link Library, DLL) não pode solicitar rotinas do programa principal, mas apenas ser solicitada por este. Como o SE faz parte de uma DLL, foi necessário construir uma outra estrutura de objetos que armazena as informações geradas pelo sistema **Enxuto** necessárias ao funcionamento da máquina de inferência: atributos da modelagem e simulação. Esta estrutura de objetos é criada e atualizada a partir de informações fornecidas pelo ambiente **Enxuto** ao final de cada simulação. É composta pelo objeto *Modelo*, responsável pelo controle e acesso a esta estrutura. Este objeto possui um apontador para uma lista de elementos que representam os objetos da fábrica (máquinas, clientes, fornecedores...). Cada elemento possui um apontador para uma lista com os valores de seus atributos nas duas últimas simulações. Esta estrutura identifica quais atributos tiveram alterações significativas nos seus resultados e quais possuem um valor final que necessita de uma avaliação mais cuidadosa pelos usuários (neste último caso, baseia-se em informações obtidas com o objeto *BD*).

A última estrutura de objetos compõe a máquina de inferência propriamente dita. O objeto principal desta estrutura é o *Expert*, que recebe todas as chamadas feitas à DLL pelo sistema **Jonas**. Além de gerenciar as chamadas aos objetos *BD* e *Modelo*, *Expert* aponta para uma subestrutura onde é armazenada uma lista com as mudanças significativas ocorridas no modelo ou os resultados finais de simulação que devem ser avaliados com atenção.

A figura 31 apresenta as estruturas de objetos de uma forma simplificada.

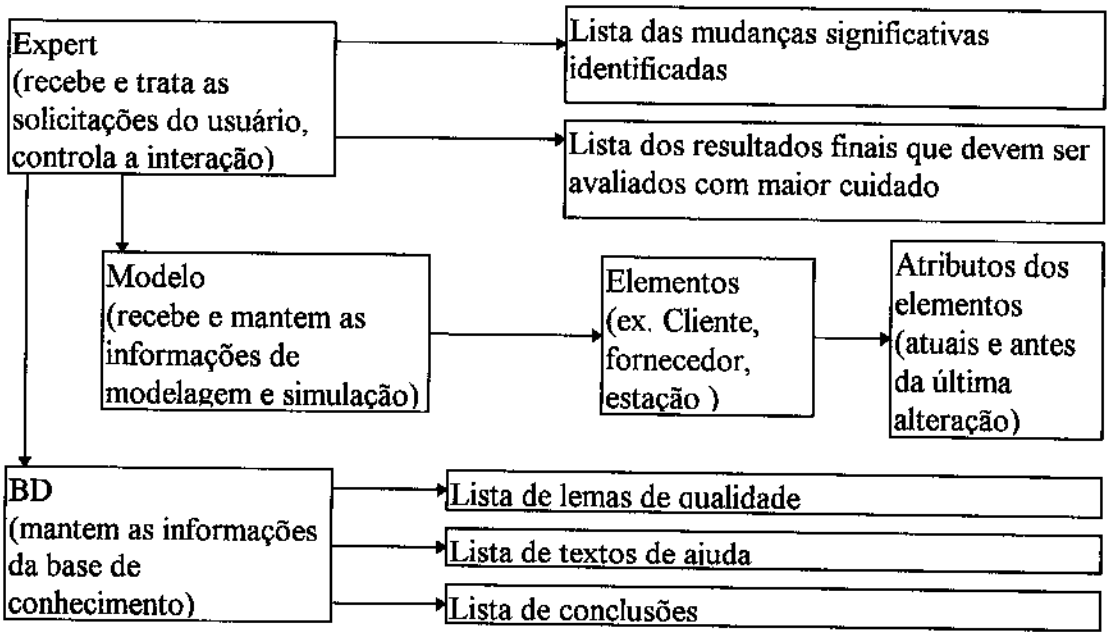


Figura 31: Visão simplificada da estrutura de objetos

Como descrito anteriormente, a máquina de inferência e sua BC estão representados em estruturas de objetos, constituindo um SE completamente OO. Identificadas as estruturas de objetos/classes, passou-se a utilizar da metodologia OMT para projeto de sistemas OO. Como primeiro passo, foi construído o modelo de objetos para as classes principais das três estruturas previamente identificadas (máquina de inferência, BC e modelo). Em seguida, foram construídos os modelos dinâmicos (ou de transição de estados) destes objetos. A partir dos dois modelos o código destas classes e seus objetos foi gerado, assim como um *driver* que simulava as chamadas a serem feitas e *stubs* (Pressman, 1984) que simulavam o funcionamento de um objeto que ainda não havia sido construído. Estando todo o funcionamento dos objetos

de acordo com os modelos da OMT, os modelos para objetos naquele momento representados por *stubs* eram definidos e codificados, além de se construir as novas *stubs* necessárias. Posteriormente, novos testes eram feitos. Esta seqüência de passos foi repetida até a construção de todo o SE. O modelo final de objetos está apresentado no apêndice 3.

Segundo a OMT, deve-se desenhar o modelo de objetos completo e os modelos dinâmicos antes de iniciar-se a programação. Tal estratégia foi inicialmente testada pelo autor, mas os resultados não estavam sendo satisfatórios. Isto deveu-se ao fato de que estava sendo muito difícil desenhar todo o modelo de objetos sem antes conseguir verificar se estes objetos funcionariam de acordo com o esperado. Também era difícil de se fazer abstrações em diferentes níveis, simultaneamente, uma vez que nada havia sido implementado. Devido a estas dificuldades, resolveu-se utilizar a estratégia mista descrita, combinando a OMT, feita então apenas para um certo nível de objetos, e *drivers* e *stubs* simulando níveis inferiores. A implementação deste nível levava a alterações no modelo de objetos, e, somente após um nível estar devidamente implementado e testado, voltava-se a OMT, buscando definir um nível inferior. Deste modo, passava-se para um novo nível de abstração com a certeza de que o nível anterior estava corretamente definido, o que diminuiu, em muito, problemas resultantes de detecção tardia de falhas nos modelos (correções).

7.3 - Subsistema administrativo

O subsistema administrativo foi construído utilizando-se as ferramentas de consultas, formulários e relatórios disponibilizadas no Access (Microsoft, 1994). Sua construção foi em muito simplificada, uma vez que este sistema gerenciador de banco de dados auxilia neste tipo de atividade, guiando a criação com seus *consultores*. Colaborou com esta simplicidade o fato de este subsistema resumir-se a telas de consulta e manipulação de registros de tabelas de banco de dados. Os formulários são apresentados no apêndice 4.

7.4 - Discussão

Devido a limitações comuns do ambiente fabril, o ambiente deve ser passível de ser instalado em um microcomputador padrão IBM-PC, com configurações não muito exigentes: um microcomputador IBM-PC 486, com 8 MB de memória RAM e 200 MB de disco rígido disponíveis deve ser suficiente. Todo o projeto teve este padrão de *hardware* como premissa básica.

A utilização de técnicas de *design* centrado no usuário demonstrou ter sido uma opção acertada. O sistema resultante dos experimentos demonstrou ser simples de manipular e com um grande potencial para seus futuros usuários.

Buscando explorar as facilidades em utilizar um paradigma de programação mais moderno que além de possuir um grande potencial é similar ao utilizado no **Enxuto**, a programação do sistema é orientada a objetos (OO).

O protótipo (e a interface resultante) foi construído em Multimedia ToolBook 4.0 (Asymetrix, 1995). Esta ferramenta foi selecionada buscando evitar problemas de compatibilidade entre o **Jonas** e a interface do ambiente **Enxuto**, também construída em ToolBook. Com exceção da vantagem de compatibilidade, para aplicações como o **Jonas**, que não exigem recursos multimídia, o ToolBook demonstrou ser uma opção não vantajosa, pois:

- a linguagem OpenScript, utilizada no ambiente ToolBook, possui uma sintaxe muito diferente das linguagens mais difundidas (como o Pascal e o C++), consumindo mais tempo no desenvolvimento;
- seus recursos de ajuda *on-line* e *debug* são muito limitados;
- é difícil a interligação com outros *software*;
- limitações da própria linguagem impossibilitam a construção de um programa OO complexo;
- gera um código executável com tamanho grande e velocidade baixa.

Em situações onde a compatibilidade não justifique a utilização do ToolBook, e onde não há a necessidade da utilização de recursos multimídia, sugere-se a utilização das ferramentas visuais da própria linguagem utilizada na construção dos programas propriamente ditos. No caso do **Jonas**, se não houvesse a necessidade de compatibilidade com o **Enxuto**,

poderia ser mais interessante utilizar-se das ferramentas visuais disponíveis no C++, onde foi construído o SE. A facilidade em trocar informações entre **Jonas** e **Enxuto** decorrente da utilização da mesma versão do ToolBook, no entanto, justificou a utilização desta linguagem.

Buscando evitar maiores problemas na integração **Jonas/Enxuto**, diminuir o custo de desenvolvimento e construir um sistema com boa performance computacional, foi decidido o uso de uma linguagem de uso geral e não uma voltada para a construção de programas de IA (Wallach, 1987; Woolf et al., 1987). Como no caso do **Jonas** há também a necessidade de se comunicar com um BD em Access (o que exige uma linguagem com facilidade de comunicação), definiu-se o uso da linguagem C++ na construção do SE. Em teoria (cuja prática demonstrou nem sempre ser correta), esta linguagem possibilita grande flexibilidade, facilitando a comunicação com desenvolvimentos em outros ambientes (Wallach, 1987). Outra vantagem é que C++ disponibiliza todas as potencialidades da orientação a objetos. A estratégia definida foi construir uma DLL que incluísse a máquina de inferência e que seria executada a partir do código ToolBook. Esta definição também levou em consideração a facilidade em se integrar o **Jonas** com o **Enxuto**. O **Enxuto** também foi construído em ToolBook, chamando uma DLL programada em C++. Ao se utilizar da mesma estratégia, buscou-se minimizar problemas com compatibilidade.

Na construção do SE, inicialmente, foi utilizado o Visual C++ da Microsoft (1995B). Sendo da Microsoft, a princípio, seria mais fácil a integração com o Access. Mas, como no desenvolvimento do **Enxuto** não se conseguiu gerar uma DLL acessível pelo ToolBook utilizando-se o Visual C++, decidiu-se utilizar o Borland C++ (Borland, 1996) tal qual já estava sendo utilizado na codificação do **Enxuto**.

A utilização do Borland C++ acabou frustrando as expectativas iniciais, pois:

- a ajuda *on-line* e a documentação impressa fornecida não são completas nem claras, havendo inclusive muitos erros. Um exemplo: ao se criar uma subclasse, incorreu-se em erro de sintaxe identificado pelo compilador. O erro indicado pelo compilador não correspondia aos possíveis problemas que podiam ter acontecido. Partiu-se para a leitura da ajuda *on-line* e dos manuais da linguagem. Após dispendidas várias horas sem sucesso, o problema foi resolvido com uma consulta a um programa exemplo em um livro didático (Entsminger,

1990). Acredita-se que programas-exemplo, assim como uma explicação correta dos códigos de erros de compilação, deveriam fazer parte de uma ajuda *on-line* de qualidade;

- a facilidade em se integrar o C++ com outros ambientes acabou revelando-se inexistente: o acesso ao BD em Access via *conectividade de banco de dados abertos* (ODBC) mostrou-se bastante complexo de ser implementado, sendo que só se conseguiu fazê-lo de modo não plenamente satisfatório[2]. A construção de uma DLL que fosse acessível ao ToolBook foi um longo exercício de compilações, alterações nas configurações, testes com versões diferentes do Borland C++, até chegar-se a uma DLL acessível (que necessitou ser construída em uma versão anterior do *software*);
- o ambiente de desenvolvimento da última versão demonstrou ser pouco robusto, gerando freqüentes problemas. Sugere-se a utilização da versão (Borland,1995).

A utilização da OO demonstrou ser uma opção muito interessante, especialmente para sistemas que envolvam um modelo, tipicamente composto por objetos, métodos e atributos. Mesmo a construção da máquina de inferência foi em muito simplificada devido à OO.

A técnica de desenvolvimento utilizada, que conjuga a OMT de Rumbaugh et al. (1991) e a Engenharia de Software de Pressman (1984), demonstrou ser uma opção interessante para um projeto OO. Isto porque, apesar de o *modelo de objetos* e o *modelo dinâmico* da OMT representarem bem classes, é difícil construir o modelo de uma classe sem, prévia e detalhadamente, se conhecer os métodos que devem ser oferecidos e a seqüência na qual eles serão solicitados. Apesar de indicadas no modelo, as mensagens recebidas por um objeto só são realmente identificadas após a codificação das classes que enviarão estas mensagens, codificação esta que sempre leva a uma reavaliação do modelo. A estratégia seguida demonstrou ser eficiente, conduzindo a um progresso natural e em paralelo nos modelos e na programação.

De modo a facilitar a construção do subsistema administrativo utilizando-se de uma ferramenta disponível em ambiente de fábrica, decidiu-se pela utilização do Access (Microsoft,

[2] O autor, sem conhecer em profundidade o Visual Basic (Microsoft,1995A), por exemplo, implantou um acesso desta linguagem ao Access sem grandes dificuldades, baseando-se apenas na ajuda *on-line* disponível.

1994). A ferramenta mostrou-se bastante interessante para este tipo de desenvolvimento, pois nela pode-se:

- definir as tabelas, seus campos e seus relacionamentos;
- definir as consultas ao banco de dados em *linguagem estruturada de consulta* (SQL), acessíveis por meio de ODBC;
- construir formulários e relatórios (subsistema administrativo).

Todas estas possibilidades foram utilizadas durante o desenvolvimento do subsistema administrativo. O administrador não precisará conhecer estas características: ele apenas irá fazer manutenções no banco de dados utilizando-se de telas de cadastramento e consulta previamente construídas.

No próximo capítulo é feita uma discussão final sobre o trabalho e são apresentados possíveis projetos de continuidade.

8 - Conclusões e trabalhos futuros

Neste trabalho foi descrito o desenvolvimento de um sistema especialista com objetivos educacionais, a ser utilizado na formação de funcionários do chão-de-fábrica de empresas. O sistema especialista resultante pode ser utilizado para formação em qualquer empresa com manufatura baseada em linha de produção.

A proposta inicial de utilizar o *design* centrado no usuário (DCU) na elaboração de um sistema construcionista baseado no aumento de inteligência (AI) acabou resultando em um *design* centrado no aprendiz (DCA), de forma independente da divulgação desta forma de *design* na literatura acadêmica. Em outras palavras, a literatura e o trabalho convergiram em paralelo para uma estratégia de *design* bastante similar. Observa-se que muitos autores ao descrever sistemas com DCA detalham o funcionamento do sistema, sem atentar para uma discussão sobre o processo de *design* em si. Este trabalho mostra com um razoável nível de detalhe um exemplo prático de DCA. A utilização das teorias de DCU, construcionismo e AI, ou em outras palavras, desta proposta de DCA mostrou ser viável e pode servir como um modelo para outros desenvolvimentos nesta linha.

A partir dos experimentos realizados com os usuários durante o processo de *design*, a observação da interação operário-**Jonas** sugere que o sistema resultante possa ser bastante útil no auxílio à formação dos funcionários de fábricas. O acompanhamento detalhado da implantação do sistema na fábrica, prevista como continuidade a este trabalho (segundo semestre de 1997 e 1998) pode viabilizar uma análise mais profunda da utilização das metodologias de ensino e desenvolvimento propostas, sendo um trabalho futuro não-trivial, interessante e necessário.

Quando o sistema **Jonas** é avaliado individualmente a interface resultante é direta, mas poderia ser repensada se for analisado o sistema **Enxuto** como um todo. Uma interface mais robusta entre **Jonas** e **Enxuto** poderia ser projetada, de modo a viabilizar ao primeiro indicar diretamente no modelo os resultados apresentados tornando o sistema **Jonas** "transparente" no ambiente **Enxuto**. Esta alteração faria com que a interface do ambiente **Enxuto** como um todo

fosse mais direta. Avaliar a possibilidade de se usar recursos de multimídia como animação e imagens de modo a tornar a interface do sistema ainda mais amigável pode ser interessante.

Fazer com que o **Jonas** possa aprender (ou seja, aumentar a quantidade ou melhorar a qualidade de seu conhecimento (Ören, 1986)) a partir de avaliações feitas nos resultados das simulações é algo que pode ser estudado com maior cuidado. O tipo de aprendizado poderia ser o chamado indutivo ou por exemplos (Bratko, 1993).

A implementação demonstrou claramente que é possível se construir, em um intervalo de tempo razoavelmente curto (por volta de um ano entre análise, projeto e programação para um desenvolvedor), um ambiente de aprendizado na empresa utilizando-se de um ambiente computacional bastante comum: computadores padrão IBM-PC, com sistema operacional Windows. Demonstrou também as vantagens de se utilizar orientação a objetos neste tipo de sistema e o potencial e as limitações das linguagens C++ e Multimedia ToolBook, além do gerenciador de banco de dados Access. A conjugação das linguagens Multimedia ToolBook, C++ e do Access mostrou ser contraproducente e deve ser evitada. Por outro lado, a proposta de desenvolvimento que conjuga a técnica de modelagem de objetos de Rumbaugh et al. (1991) e a engenharia de *software* de Pressman (1984) foi positiva.

O ambiente industrial demonstrou ser uma ótima fonte de observações e trabalhos de pesquisa na área de formação, pois tem interesse em sistemas educacionais e possui toda uma gama de particularidades que o diferenciam dos outros tipos de ambiente educacional. Devido a necessidade cada vez maior de melhor formação em empresas, a receptividade destas em desenvolver projetos conjuntos nesta área tende a ser a cada dia maior.

Este trabalho pode servir como exemplo de estratégias para que mais projetos que auxiliem a indústria do país a desenvolver-se, tornando-se mais competitiva, sejam efetuados em pesquisas acadêmicas. O desenvolvimento desta tese demonstrou que a interação entre os meios produtivo e acadêmico é rica em oportunidades e desafios ainda não explorados.

Referências bibliográficas

1. **ABRANCHES, S.** A revolução que liquidou o emprego. *Revista Veja*, Editora Abril, p.95, 19 out. 1994.
2. **AHAMED, S.V., ROMAN, E.G.** A model-based expert system for decision support in negotiating. In: ELZAS, M.S., Ören, T.I., Zeigler, B.P. (Ed.). *Modelling and simulation methodology in the artificial intelligence era.* Amsterdam, Holanda: North-Holland, 1986. p. 339-352.
3. **ASYMETRIX.** *ToolBook V.3.0.* Asymetrix, 1994.
4. **ASYMETRIX.** *Multimedia ToolBook, V.4.0.* Asymetrix, 1995.
5. **BARANAUSKAS, M.C.C., BORGES, M.A.F.** Learning by creating models: a computer-based environment for industrial application. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE IN EDUCATION, 8, 1997, Kobe. *8th Conference on Artificial Intelligence in Education*, Kobe (Japão), 1997. p. 426-433.
6. **BARANAUSKAS, M.C.C., OLIVEIRA, O.L.** Estratégias para design de ambientes computacionais para modelagem. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 5, 1994, Porto Alegre. *Anais do V Simpósio Brasileiro de Informática na Educação*, Porto Alegre:[s.n.], 1994. p. 25-37.
7. **BERTIN, A., BUCIOL, F., LANZA, C.** Intelligent training systems in industrial environments: approach and solutions to high risk task training. *Computers Education*, Grã Bretanha, v.20, n.1., p. 97-104, 1993.
8. **BETING, J.** *Jornal A Tribuna*, Santos, 24 dez. 1994. Coluna secos e molhados, p. B-3.
9. **BORGES, E.L., BORGES, M.A.F., BARANAUSKAS, M.C.C.** Da simulação à criação de modelos - um contexto para a aprendizagem na empresa. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 6, 1995, Florianópolis. *Anais do VI Simpósio Brasileiro de Informática na Educação*, Florianópolis: [s.n.], 1995. p. 154-165.
10. **BORGES, M.A.F., BARANAUSKAS, M.C.C.** Jonas: um sistema especialista para formação na empresa. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 7, 1996, Belo Horizonte. *Anais do VII Simpósio Brasileiro de Informática na Educação*, Belo Horizonte: [s.n.], 1996, p. 265-276 e conferência apresentada no VII Congresso Internacional de Informática e Aprendizagem, Rio de Janeiro-RJ, Brasil 1996.
11. **BORGES, M.A.F., BARANAUSKAS, M.C.C.** An user-centered approach to the design of an expert system for training. In: INTERNATIONAL PEG CONFERENCE, 8, 1997A, Sozopol. *Proceedings of the Eighth International PEG Conference*, Sozopol (Bulgária), 1997A. p. 74-81.
12. **BORGES, M.A.F., BARANAUSKAS, M.C.C.** The user as a partner in the design of an interface for a training system. In: IFIP WG 3.3 WORKING CONFERENCE: HUMAN-COMPUTER INTERACTION AND EDUCATIONAL TOOLS, 1997B, Sozopol. *IFIP WG 3.3 Working Conference: Human-Computer Interaction and Educational Tools*, Sozopol (Bulgária), 1997B (in press).
13. **BORLAND.** *C++ V.4.51.* Borland, 1995.
14. **BORLAND.** *C++ V.5.0.* Borland, 1996.

15.**BÖSSER, T.** *Learning in man-computer interaction: a review of the literature*, Berlim (Alemanha): Springer-Verlag, 1987.

16.**BOUTILIER, C., BECHER, V.** Abduction as belief revision. *Artificial intelligence*, v.77, n.1, p.43-94, august 1995.

17.**BRATKO, I.** Machine learning in artificial intelligence. *Artificial intelligence in engineering*, v.8, p.159-164, 1993.

18.**BROWN, J.S., BURTON, R.R., KLEER, J.** Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In: SLEEMAN, D., BROWN, J.S. (Ed.). *Intelligent tutoring systems.* EUA: Academic Press, 1982.

19.**CAO, Y., GRAHAM, J.H., ELMAGHRABY, A.S.** Communications approaches for simulation-AI interactions. *Simulation digest*, v.23, n.2, p.3-16, winter 1993.

20.**CARDOZO, C.M., RAMOS, G.M.F.** AALO - Um ambiente para a aprendizagem de lógica. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 6,1995, Florianópolis. *Anais do VI Simpósio Brasileiro de Informática na Educação*, Florianópolis: [s.n.], 1995. p.120-131.

21.**CARDOZO, C.M., MOTARI, C.A.** IA Oásis - Um ambiente de simulação baseado em agentes para aprendizagem de lógica. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 7, 1996, Belo Horizonte. *Anais do VII Simpósio Brasileiro de Informática na Educação*, Belo Horizonte:[s.n.], 1996. p. 467-469.

22.**CIAMPA, D.** *Total quality*. EUA:Addison-Wesley, 1992.

23.**COPERS & LYBRAND.** Vídeo "Os fundamentos do Change Managment", Coopers & Lybrand Consultoria.

24.**COLE, R.** *Strategies for learning*. California: University of California, 1989.

25.**CRAWFORD, C.** Lessons from computer game design. In: LAUREL,B. (Ed.). *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p.103-111.

26.**DE JONG, T.** Learning and instruction with computer simulations. *Education & Computing*, v. 6, p. 217-229, 1991.

27.**DEMING, W.E.** *Out of crisis*. Cambridge-Massachusetts: MIT, Center for Advanced Engineering Study, 1992.

28.**EDELSON, D.C., PEA, R.D., GOMEZ, L.M.** The colaborative notebook. *Communications of the ACM*, , v.39, n.1, p.32-33, april 1996.

29.**EDEN, H., EISENBERG, M., FISCHER, G., REPENNING, A.** Making learning a part of life. *Communications of the ACM*, v.39, n.1, p.40-42, april 1996.

30.**ELLIS, J.E.** Roger Schank wants your child's mind. *Business week*, p.36-37, july 18, 1994.

31.**ENTSMINGER, G.** *The tao of objects*; a beginner's guide to object-oriented programming. New York: M&T, 1990.

32.**ERICKSON, T.D.** Working with interface metaphors. In: LAUREL, B. (Ed.). *The art of human-computer interface design.* Massachusetts:Addison-Wesley, 1990. p. 65-73.

33.**FEIGENBAUN, A.V.** *Total quality control*. New York: McGraw-Hill, 1988.

34.**FEIGENBAUN, E.A., BUCHANAN, B.G.** DENDRAL and Meta-DENDRAL: roots of knowledge systems and expert system applications. *Artificial intelligence*, v.59, p. 233-240, 1993.

35.**FERNANDES, L.D., FURQUIM, A.A., BARANAUSKAS, M.C.C.** Jogos no computador e a formação de recursos humanos na indústria. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 6, 1995, Florianópolis. *Anais do VI Simpósio Brasileiro de Informática na Educação*, Florianópolis: [s.n.], 1995. p. 1-14.

36.**FERRO, J.R.** (*Decifrando culturas organizacionais*. São Paulo: EAESP-FGV, 1991. (Tese, Doutoramento em Administração de Empresas).

37.**FERRO, J.R.** *Cultura organizacional e qualidade total: o lado humano da qualidade*: Mestrado em Qualidade, Universidade Estadual de Campinas, 1995. (Notas de aula).

38.**FEURZEIG, W.** Algebra slaves and agents in a Logo-based Mathematics curriculum. In: LAWLER, R.W., YAZDANI, M. (Ed.). *Artificial intelligence and education*; Learning environments and tutoring systems. Norwood-NJ, EUA: Ablex, V. 1, 1987. p. 27-54.

39.**FISCHER, G.** Rethinking and reinventing artificial intelligence from the perspective of human-centered computational artifacts. In: BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 12, 1995, Campinas. *Anais do XII Brazilian Symposium on Artificial Intelligence*, Campinas:[s.n.], 1995. p. 1-11.

40.**FOX, M.S., HUSAIN, N., MCROBERTS, M., REDDY, Y.V.** Knowledge-based simulation: an artificial intelligence approach to system modeling and automating the simulation life cycle. In: WIDMAN, L.E., LOPARO, K.A., NIELSEN, N.R. (Ed.). *Artificial intelligence, simulation, and modeling*. New York: John Wiley & Sons, 1989. p. 447-486.

41.**GLADWIN, L.A.** The impact of artificial intelligence on training. *Training and Development Journal*, p.46-47, dezembro de 1984.

42.**GM Brasil**. Vídeo "Controle Estatístico de Processos", GM Brasil.

43.**GM (1989)** *Implementation Guide for Pull Systems*. EUA: documento interno da GMC), 1989.

44.**GOLDRATT, E.M., COX, J.** *A Meta - Um processo de aprimoramento contínuo*. 16.ed. São Paulo :Educator, 1994.

45.**GOMOLL, K.** Some techniques for observing users. In: LAUREL,B. (Ed.). *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p. 85-90.

46.**GRAY, N.A.B.** Dendral and Meta-Dendral - The myth and the reality. *Chemometrics and intelligent laboratory systems*, Amsterdam , v. 5, n.1, p.11-32, novembro de 1988.

47.**GRUDIN, J.** Groupware and cooperative work: problems and prospects. In: LAUREL,B. (Ed.). *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p.171-195.

48.**GUIDA, G., TASSO, C.** *Topics in expert system design*; methodologies and tools. Amsterdam: Elsevier Science, 1989.

49.**GUIMARÃES, A.M.** Abertura do VII Simpósio Brasileiro de Informática na Educação, Belo Horizonte-MG, Brasil, 1996.

50.**GUZDIAL, M., KOLODNER, J., HMELO, C., NARAYANAN, H., CARLSON, D., RAPPIN, N., HÜBSCHER, R., TURNS, J., NEWSTETTER, W.** Computer support for learning through complex problem solving. *Communications of the ACM*, vol.39, n.1, p.43-45, april 1996.

51.**GUIMARÃES, L.F.A.** *Um algoritmo "Branch and Bound" para um modelo de otimização de um sistema "kanban"*. Campinas: Faculdade de Engenharia Elétrica,

Universidade Estadual de Campinas, 1991. (Dissertação, Mestrado em Engenharia de Sistemas).

52. **HAKANSSON, J.** Lessons learned from kids: one developer's point of view. In: LAUREL,B. (Ed.). *The art of human-computer interface design.* Massachusetts: Addison-Wesley, 1990. p.123-130.

53. **HASEGAWA, R., NUNES, M.G.V.** Tootema: uma ferramenta para construção de sistemas tutores inteligentes em matemática. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 6, 1995, Florianópolis. *Anais do VI Simpósio Brasileiro de Informática na Educação*, Florianópolis: [s.n.], 1995. p. 315-326.

54. **HEBENSTREIT, J.** Simulation as an educational tool. In: INTERNATIONAL CONFERENCE ON TECHNOLOGY AND EDUCATION, 1991, Toronto. *Proceedings of International Conference on Technology and Education*, Toronto, Canada, 1991. **apud** PAGANO, R. *Computer simulation as an educational tool.* Bélgica: Université Catholique de Louvain, 1992. (PhD thesis).

55. **HOLLAN, J.D., HUTCHINS, E.L., WEITZMAN, L.M.** Steamer: an interactive, inspectable, simulation-based training system. In: KEARSLEY, G. (Ed.). *Artificial Intelligence And Instruction Applications and Methods.* Massachusetts: Addison Wesley,1987. p.113-134.

56. **HUTCHINS, E.L., HOLLAN, J.D., NORMAN, D.A.** Direct manipulation interfaces. In: NORMAN, D.A., DRAPER, S.W. (Ed.). *User centered system design: new perspectives on human-computer interaction.* Hillsdale-NJ, EUA: Lawrence Erlbaum, 1986. p.87-124.

57. **JOHNSON, W.L., SOLOWAY, E.** Proust: an automatic debugger for Pascal programs. In: KEARSLEY, G. (Ed.). *Artificial Intelligence And Instruction Applications and Methods.* Massachusetts: Addison Wesley,1987. p.49-67.

58. **KAFAI, Y.B.** Software by kids for kids. *Communications of the ACM*, v.39, n.1, p. 38-39, april 1996.

59. **KALAS, I.** Designing educational environments. Quicksand for developers? In: CONGRESSO INTERNACIONAL DE INFORMÁTICA E APRENDIZAGEM, 7, 1996, Rio de Janeiro. *Anais do VII Congresso Internacional de Informática e Aprendizagem*, Rio de Janeiro: [s.n.], 1996 (no prelo).

60. **KAY, A C.** Computers, networks and education. *Scientific american*, p.148-155, 1995.

61. **KEARSLEY, G.** *Artificial Intelligence And Instruction Applications and Methods.* Massachusetts: Addison Wesley, 1987A. p.1: Introduction.

62. **KEARSLEY, G.** *Artificial Intelligence And Instruction Applications and Methods.* Massachusetts: Addison Wesley, 1987B. p.3-10: Overview.

63. **KULIKOWSKI, C.A.** Artificial intelligence, modeling, and simulation. In: KULIKOWSKI, C.A., HUBER, R.M., FERRATE, G.A. (Ed.). *Artificial intelligence, expert systems, and languages in modelling and simulation.* North-Holland (Holanda): Elsevier Science, 1988. p. 5-13.

64. **LAUREL, B.K.** Interface as mimesis. In: NORMAN, D.A., DRAPER, S.W. (Ed.). *User centered system design: new perspectives on human-computer interaction.* Hillsdale-NJ, EUA: Lawrence Erlbaum, 1986. p.67-85.

65.**LAUREL, B.K., MOUNTFORD, S.J.** Introduction. In: LAUREL,B. (Ed.). *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p.xi-xvi.

66.**LAUREL, B.K.** *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p.91-93: Users and contexts: introduction.

67.**LAWLER, R.W.** Learning environments: now, then, and someday. In: LAWLER, R.W., YAZDANI, M. (Ed.). *Artificial intelligence and education*; Learning environments and tutoring systems. Norwood-NJ, EUA: Ablex, V. 1, 1987. p. 1-26.

68.**LAWLER, R.W., YAZDANI, M.** *Artificial intelligence and education*; Learning environments and tutoring systems. Norwood-NJ, EUA: Ablex, V. 1, 1987. p. ix-xiii: Introduction.

69.**LEE, M.H.** The knowledge-based factory. *Artificial intelligence in engineering*, Essex (Inglaterra), v.8, n.2, p.109-125, 1993.

70.**LINDFIELD, G.R.** The role of the Stella package in simulation and modelling. *International Journal of Mathematics, Education, Science and Technology*, v.23, n.6, p. 865-880, 1992.

71.**LUGER,G.F., STUBBLEFIELD,W.A.** *Artificial intelligence and the design of expert systems*. Redwood-CA, EUA: Benjamin/Cummings, 1989.

72.**MARQUES, J.A.V.C.** Sistema de custos com base em atividades: uma evolução das filosofias de produção e de contabilidade. *Revista de Administração de Empresas*, São Paulo (Fundação Getúlio Vargas), v. 34, n.6, p. 20-32, novembro-dezembro/1994.

73.**MAZZONE, J.S.** O Sistema "Enxuto" e a educação no Brasil. In: VALENTE, J.A. (Ed.). *Computadores e Conhecimento: Repensando a Educação*. Campinas: Gráfica Central da Unicamp,1993. p. 274-312.

74.**MAZZONE, J.S.** Educação na sociedade enxuta. *Documento interno, memo nº 33, Nied, Universidade Estadual de Campinas*, Campinas, 1995.

75.**MCDERMOTT, J.** R1 ("XCON") at age 12:lessons from an elementary school achiever. *Artificial intelligence*, v.59, n.1-2, 241-247, february 1993.

76.**MICROSOFT.** *Access V.2.00*. Microsoft, 1994.

77.**MICROSOFT.** *Visual Basic V. 4.0*. Microsoft, 1995A.

78.**MICROSOFT.** *Visual C++ V. 4.00*. Microsoft, 1995B.

79.**MOURA, R.A.** *Kanban: a simplicidade do controle de produção*. , São Paulo: IMAM, 1989.

80.**MURO, Z., YANAGIHARA, N., FUJIMOTO, H.** Super simulation shell for plants. In: THE 1991 WINTER SIMULATION CONFERENCE, 1991, Phoenix-Arizona. *Proceedings of the 1991 Winter Simulation Conference*, Phoenix-Arizona: [s.n], 1991. p.289-293.

81.**NAYLOR, T.H., BALINTFY, J.L., BURDICK, D.S., CHU, K.** *Técnicas de simulação em computadores*. Brasil: Vozes, 1971.

82.**NEUWIRTH, E.** Internet e aprendizagem. In: CONGRESSO INTERNACIONAL DE INFORMÁTICA E APRENDIZAGEM, 7, 1996, Rio de Janeiro. *Anais do VII Congresso Internacional de Informática e Aprendizagem*, Rio de Janeiro: [s.n.], 1996 (no prelo).

83.**NICOL, A.** Interfaces for learning: what do good teachers know that we don't. In: LAUREL,B. (Ed.). *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p.113-122.

84.**NIED.** *Projeto de Dinamização da Formação e da Aprendizagem nas Empresas*; documento interno do Núcleo de Informática Aplicada à Educação, Universidade Estadual de Campinas. Campinas, 1995.

85.**NORMAN, D.A.** Cognitive engineering. In: NORMAN, D.A., DRAPER, S.W. (Ed.). *User centered system design: new perspectives on human-computer interaction.* Hillsdale-NJ, EUA: Lawrence Erlbaum, 1986. p.31-62.

86.**NORMAN, D.A., DRAPER, S.W.** *User centered system design: new perspectives on human-computer interaction.* Hillsdale-NJ, EUA: Lawrence Erlbaum, 1986. p.63-65: The interface experience.

87.**NORMAN, D.A., SPOHRER, J.C.** Learner-centered education. *Communications of the ACM*, v.39, n.4, p.24-27, april 1996.

88.**NUNES, M.G.V., TAKEHARA, R.S., MENDES, M.D.C.** A network-based model for intelligent tutoring systems. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 10, 1993, Porto Alegre. *Anais do X Simpósio Brasileiro de Inteligência Artificial*, Porto Alegre:[s.n.], 1993, p.277-288.

89.**O'KEEFE, R.M.** The role of artificial intelligence in discrete-event simulation. In: WIDMAN, L.E., LOPARO, K.A., NIELSEN, N.R. (Ed.). *Artificial intelligence, simulation, and modeling.* New York: John Wiley & Sons, 1989. p. 359-380.

90.**OLIVEIRA, O.** *Design de ambientes computacionais para modelagem em um contexto educacional.* Campinas: Universidade Estadual de Campinas, 1995. (Dissertação. Mestrado em Ciência da Computação).

91.**ÖREN, T.I.** Implications of machine learning in simulation. In: ELZAS, M.S., ÖREN, T.I., ZEIGLER, B.P. (Ed.). *Modelling and simulation methodology in the artificial intelligence era.* Amsterdam: North-Holland,1986. p.41-60.

92.**PAGANO, R.** *Computer simulation as an educational tool.* Bélgica: Université Catholique de Louvain, 1992. (PhD thesis).

93.**PAPERT, S.** *Construcionism: a new opportunity for elementary science education; a* proposal to the National Science Foundation. Cambridge-Massachusetts: MIT, Media Lab., Epistemology and Learning Group, 1986.

94.**PAPERT, S.** Microworlds: transforming education. In: LAWLER, R.W., YAZDANI, M. (Ed.). *Artificial intelligence and education*; Learning environments and tutoring systems. Norwood-NJ, EUA: Ablex, V. 1, 1987. p. 79-94.

95.**PEREZ, R.S., SEIDEL, R.J.** Intelligent CAI: old wine in new bottles, or a new vintage? In: KEARSLEY, G. (Ed.). *Artificial Intelligence And Instruction Applications and Methods.* Massachusetts: Addison Wesley, 1987. p.11-45.

96.**PETER, T., WATERMAN, R.** Man waiting for motivation. In: *In search of excellence.* New York: Warner,1984. p. 55-86.

97.**PIMENTEL, M.G.C., HAGUI, S.H.** Usando a WWW como ferramenta de apoio ao ensino. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 7, 1996, Belo Horizonte. *Anais do VII Simpósio Brasileiro de Informática na Educação*, Belo Horizonte: [s.n.], 1996, p. 55-68.

98.**PLANTULLO, V.L.** Um pouco além do just-in-time: uma abordagem à teoria das restrições. *Revista de Administração de Empresas*, São Paulo (Fundação Getúlio Vargas), v. 34, n. 5, p. 32-39, setembro,outubro/1994.

99.**POLLACIA, L.F.** A survey of discrete event simulation and state-of-the-art discrete event languages. *Simulation Digest*, v.20, n.3, p. 8-25, fall 1989.

100.**PRESNO, O.** Kidlink global networking for youth 10-15 years of age In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 6, 1995, Florianópolis. *Anais do VI Simpósio Brasileiro de Informática na Educação*, Florianópolis: [s.n.], 1995. p. 443-454.

101.**PRESSMAN, R.S.** *Software engineering: a pratitioner's approach*. Singapura: McGraw-Hill, 1984.

102.**RHEINGOLD, H.** An interview with Don Norman. In: LAUREL,B. (Ed.). *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p.5-10.

103.**RICHER, M.H., CLANCEY, W.J.** Guidon-Watch: a graphic interface for viewing a knowledge-based system. In: LAWLER, R.W., YAZDANI, M. (Ed.). *Artificial intelligence and education*; Learning environments and tutoring systems. Norwood-NJ, EUA: Ablex, V. 1, 1987. p. 373-412.

104.**ROTHEMBERG, J.** The nature of modeling. In: WIDMAN, L.E., LOPARO, K.A., NIELSEN, N.R. (Ed.). *Artificial intelligence, simulation, and modeling*. New York: John Wiley & Sons, 1989. p. 75-92.

105.RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., LORENSEN, W. *Object-oriented modeling and design*. New Jersey: Prentice Hall, Englewood Cliffs, 1991.

106.SCARDAMALIA, M., BEREITER, C. Student communities for the advancement of knowledge. *Communications of the ACM*, v.39, n.4, p.36-37, april 1996.

107.SCHANK, R.C., KASS, A. A goal-based scenario for high school students. *Communications of the ACM*, v.39, n.4, p. 28-29, april 1996.

108.**SCHEIN, E.** *Organizational Culture and Leadership*. San Francisco: Jossey-Bass, 1985.

109.**SCHLUNZEN Jr., K.** *SICRE - Sistema computacional para resolução de equações de 1º grau*. Campinas: Universidade Estadual de Campinas, 1992 (Dissertação, Mestrado em Ciência da Computação).

110.**SCIENTIFIC AMERICAN.** *The computer in 21st century*. Scientific American, 1995.

111.**SELLEN, A., NICOL, A.** Building user-centered on-line help. In: LAUREL,B. (Ed.). *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p.143-160.

112.**SEMLER, R.** *Virando a própria mesa*. 47.ed. São Paulo: Best Seller, 1988.

113.**SOL, H.G.** Expert systems for modelling of decision support and information systems. In: ELZAS, M.S., ÖREN, T.I., ZEIGLER, B.P. (Ed.). *Modelling and simulation methodology in the artificial intelligence era*. Amsterdam: North-Holland,1986. p.353-364.

114.**SOLOWAY, E., PRYOR, A.** The next generation in human-computer interaction. *Communications of the ACM*, v.39, n.4, p.16-18, april 1996.

115.TELOVI, J. Por que os programas de qualidade total falham? *Revista de Administração de Empresas*, São Paulo (Fundação Getúlio Vargas), v. 34, n.6, p. 6-11, novembro,dezembro/1994.

116.THOMPSON, P.W. Mathematical microworlds and intelligent computer-assisted instruction. In: KEARSLEY, G. (Ed.). *Artificial Intelligence And Instruction Applications and Methods*. Massachusetts: Addison Wesley, 1987. p.83-109.

117.TURCSÁNYL-SZABÓ, M. Changing cultures of ICT: implications to teacher training. In: CONGRESSO INTERNACIONAL DE INFORMÁTICA E APRENDIZAGEM, 7, 1996, Rio de Janeiro. *Anais do VII Congresso Internacional de Informática e Aprendizagem*, Rio de Janeiro: [s.n.], 1996 (no prelo).

118.**VALENTE, J.A.** *Computadores e Conhecimento: repensando a educação*. Campinas: Gráfica central da Unicamp, Campinas-SP, Brasil, 1993A. p.1-23: Diferentes usos do computador na educação.

119.**VALENTE, J.A.** *Computadores e Conhecimento: repensando a educação*. Campinas: Gráfica central da Unicamp, Campinas-SP, Brasil, 1993B. p.24-44: Por quê o computador na educação.

120.VALENTE, J.A. A formação de professores: uma proposta construtivista. In: CONGRESSO INTERNACIONAL DE INFORMÁTICA E APRENDIZAGEM, 7, 1996, Rio de Janeiro. *Anais do VII Congresso Internacional de Informática e Aprendizagem*, Rio de Janeiro: [s.n.], 1996 (no prelo).

121.**VEJA.** A Revolução que Liquidou o Emprego. *Revista Veja*, Editora Abril , p. 88-95, 19 out. 1994.

122. **VELTERNEY, L., ARENT, M., LIEBERMAN, H.** Two disciplines in search of an interface. In: LAUREL,B. (Ed.). *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p.45-55.

123.**VELTERNEY, L., BOOKER, S.** Designing the whole-product user interface. In: LAUREL,B. (Ed.). *The art of human-computer interface design*. Massachusetts: Addison-Wesley, 1990. p.57-63.

124.**VOLLMAN, T.E. BERRY, W.L., WHYBARK, D.C., IRWIN, D.J.** *Manufacturing planning and control systems*. 2.ed. Homewood-Illinois: Richard D.Irwin, 1988.

125.**WALLACH, B.** Development strategies for ICAI on small computers. In: KEARSLEY, G. (Ed.). *Artificial Intelligence And Instruction Applications and Methods*. Massachusetts: Addison Wesley,1987. p.305-322.

126.**WANG, W., BELL, R.** A knowledge based system structure for the modelling of flexible manufacturing systems. *Transactions of the society for computer simulation*, v.9, n.3, p.159-173, 1992.

127.**WENGER, E.** *Artificial intelligence and tutoring systems*; computational and cognitive approaches to the communication of knowledge. Los Altos-CA, EUA: Morgan Kaufmann, 1987.

128.**WIDMAN, L.E., LOPARO, K.A.** Artificial intelligence, simulation, and modeling: a critical survey. In: WIDMAN, L.E., LOPARO, K.A., NIELSEN, N.R. (Ed.). *Artificial intelligence, simulation, and modeling*. New York: John Wiley & Sons, 1989. p. 1-44.

129.**WLOKA, D.W.** Modelling and simulation of complex manufacturing cells. In: TZAFESTAS, S., EISINBERG, A., CAROTENUTO, L. (Ed.). *System modelling and simulation*. North-Holland (Holanda): Elsevier Science, 1989. p. 281-285.

130.**WOMACK, J.P., JONES, D.T., ROOS, D., FERRO, J.R.** *A máquina que mudou o mundo.* 3. ed. Rio de Janeiro: Editora Campus, 1992.

131. **WOOD, JR., T., URDAN, F.T.** Gerenciamento da qualidade total: uma revisão crítica. *Revista de Administração de Empresas*, São Paulo (Fundação Getúlio Vargas), v. 34, n. 6, p. 46-59, novembro-dezembro/1994.

132.**WOOLF, B.P.** Intelligent multimedia tutoring systems. *Communications of the ACM*, v.39, n.4, p.30-31, april 1996.

133.**WOOLF, B.P., BLEGEN, D., JANSEN, J.H., VERLOOP, A.** Teaching a complex industrial process. In: LAWLER, R.W., YAZDANI, M. (Ed.). *Artificial intelligence and education*; Learning environments and tutoring systems. Norwood-NJ, EUA: Ablex, V. 1, 1987. p. 413-427.

134.**YAZDANI, M.** Intelligent tutoring systems: an overview. In: LAWLER, R.W., YAZDANI, M. (Ed.). *Artificial intelligence and education*; Learning environments and tutoring systems. Norwood-NJ, EUA: Ablex, V. 1, 1987. p. 183-202.

135. **3M.** *Gerenciamento da qualidade total - o processo de melhoria.* 3. ed. Sumaré: 3M-SP, Brasil, 1988A.

136.**3M.** *Qualidade: um novo conceito. In Gerenciamento da qualidade total - o processo de melhoria.* 3 ed. Sumaré: 3M-SP, Brasil, 1988B.

Apêndice 1

O primeiro experimento: *check-list*, instruções para as duplas e figuras utilizadas

Experimento de 14/11/95 - GMB- Harrison

Check-list

1) Serão 3 duplas, duas de funcionarios que serão usuarios tipicos e uma formada por especialistas nas tecnicas de produção (da propria GMB) Para cada dupla havera instruções escritas O experimento sera realizado no ambiente de trabalho, num local não sujeito a interrupção Sera utilizado algum metodo de gravação, preferencialmente video, pelo menos audio

2) Indicar que o teste é do produto e não das pessoas que o farão Exibir uma idéia geral da observação (definir a interface do JONAS)

3) Indicar que eles podem desistir a qualquer momento

4) Apresentar o que esta na sala equipamentos para gravação e computador simulado

5) Pedir para que se pense alto, pois todo pensamento é importante

6) Explicar que durante o experimento não havera ajuda e que todas as dúvidas serão sanadas ao final

7) Explicar o que e o JONAS Explicar os passos que devem ser seguidos (separadamente por dupla) Entregar instruções escritas

8) Perguntar se não ha mais questões Iniciar o exercicio

9) Concluir as observações
   explicar os objetivos da observação
   responder qualquer questão
   discutir qualquer coisa interessante que foi percebida, pedindo explicações
   perguntar qual é a impressão geral

Dupla com questões padronizadas

Tarefa 1 (20min)

Definir como serão chamadas as opções explica e dica a partir da tela do enxuto

Desenhe na folha de papel com a tela do Enxuto (simplificada)

Tarefas 2,3,4 (15min/cada)

Objetivos

Um objetivo e entender o que aconteceu entre uma simulação e outra (com resultados finais apresentados nas folhas entregues) Coloque esta explicação na folha de modelos

O outro objetivo é alterar o último modelo buscando melhorias no resultado Indique as alterações no modelo, indicando onde e como mudar

Indique, na folha, se houverem informações que precisariam ser passadas mas não foram (resultados das simulações)

Ao final, escrevam o que vocês acharam desta experimentação e do futuro sistema, do que gostaram e do que não gostaram

Conseguindo informações

Indique com a borracha se vocês desejam explicação ou dica para conseguir os dois objetivos acima Uma vez pedido uma destas informações, vocês receberão uma resposta Desejando mais dados, indique em um dos botões o que vocês desejam Continuem apertando os botões e recebendo as respostas até conseguirem cumprir os objetivos acima Note que antes de apertar um botão, vocês devem verificar se conseguem atingir os objetivos sem acessá-lo Só use os botões se realmente for necessário Use-os o mínimo necessário Os usuários não precisam, e devem evitar, que o JONAS chegue ao raciocínio final. Devem perceber a linha de raciocínio e prever resultado dele antes de ser apresentado pelo computador

Importante enquanto numa explicação ou dica vocês não podem escrever na folha de modelos Antes vocês devem indicar que estão satisfeitos para que o JONAS se retire

Explicações sobre os botões

- Explicação explica o porquê dos resultados diferentes obtidos nas duas simulações Baseia-se nas alterações feitas entre elas Só indica caminhos de raciocínio, não apresenta a resposta
- Dica indica pontos que podem ser melhorados no último modelo, buscando resultados superiores na simulação Também só indica caminhos de raciocínio, não apresenta a resposta
- Continue raciocínio aproxima mais o raciocínio anterior (de dica ou explicação) da resposta que se busca Apresenta mais detalhes do raciocínio iniciado
- Mais informações explica o último raciocínio O raciocínio continuará do ponto anterior, após estas informações
- Outra dica ou explicação se vocês não gostarem ou não concordarem com a dica ou explicação que estão recebendo, o sistema buscará identificar outra possível dica ou explicação, abandonando o primeiro raciocínio
- Satisfeito vocês não desejam mais ajuda do JONAS, querem utilizar-se da folha de modelos

Dupla com questões livres

Tarefa 1 (20min)

Definir como serão chamadas as opções explica e dica a partir da tela do enxuto

Desenhe na folha de papel com a tela do Enxuto (simplificada)

Tarefas 2,3,4 (15min/cada)

Objetivos

Um objetivo é entender o que aconteceu entre uma simulação e outra (com resultados finais apresentados nas folhas entregues) Coloque esta explicação na folha de modelos

O outro objetivo é alterar o último modelo buscando melhorias no resultado Indique as alterações no modelo, indicando onde e como mudar

Indique, na folha, se houverem informações que precisariam ser passadas mas não foram (resultados das simulações)

Ao final, escrevam o que vocês acharam desta experimentação e do futuro sistema, do que gostaram e do que não gostaram

Conseguindo informações

Peçam para a dupla que simula o sistema uma Dica ou uma Explicação Recebendo esta, continuem fazendo questões até conseguirem obter as respostas aos objetivos acima Os usuarios não precisam, e devem evitar, que o JONAS chegue ao raciocinio final Devem perceber a linha de raciocinio e prever resultado dele antes de ser apresentado pelo computador Faça o minimo de questões que for necessário

Evitem questões longas Prefiram questões simples e gerais Para fazer uma questão, escreva-a numa folha de papel e passe para a dupla que simula o sistema Para facilitar a identificação das perguntas e das respostas obtidas, numere-as em ordem crescente

Dupla JONAS simulado

Tarefa 1 (7min/modelo)

Para cada um dos modelos apresentados discuta possíveis

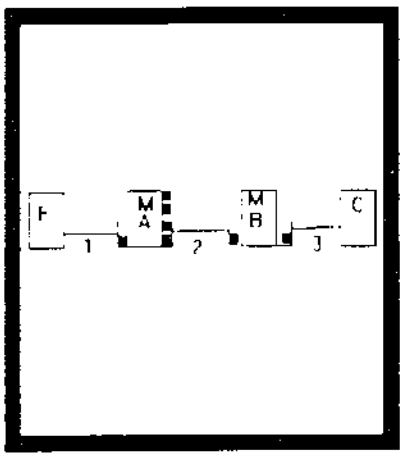
- causas para que a modificação entre os dois modelos tenha resultado na diferença nos resultados das simulações (explicação)
- possíveis melhorias no último modelo que levem a resultados melhores na simulação (dica)

Tarefa 2 (15min/modelo)

Apresente, quando requisitados pelo outro grupo, respostas as questões por eles formuladas. As respostas devem sempre ser curtas, com uma frase. Não apresente respostas diretas, indicando por exemplo que em um ponto da fábrica deve ser feita uma certa alteração. As respostas devem ser formuladas de tal modo que levem os usuários a perceber o raciocínio que vocês usaram e a chegar, eles próprios, as suas respostas.

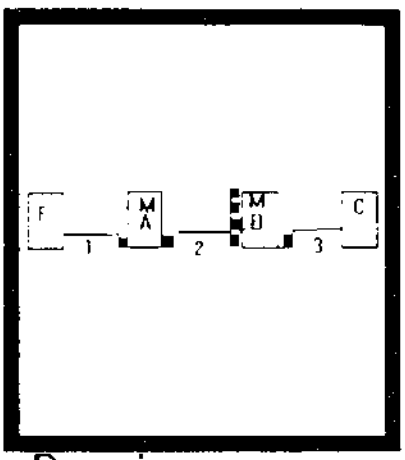
Quando o usuário pedir explicação ou dica, apresente a primeira parte do raciocínio. A partir daí, só respondam ao que for perguntado anteriormente.

Não sabendo ou não querendo responder a pergunta, podem responder que não há respostas.



ANTES

Lucro: 11
Número de Pedidos: 9
Cliente: Indiferente
Nível do depósito: 15

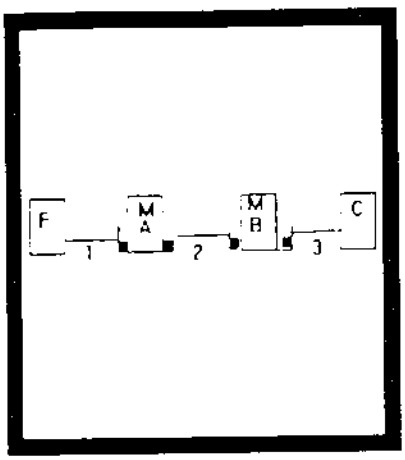


Depois

Lucro: 10
Número de Pedidos: 8
Cliente: Indiferente
Nível do depósito: 14

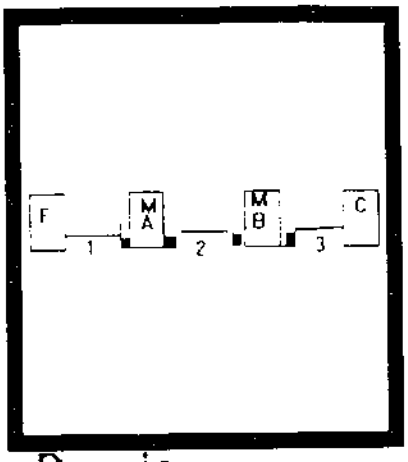
1

Alterou: aumentou a capacidade do transporte 2.



ANTES

Lucro: 11
Número de Pedidos: 9
Cliente: Insatisfeito
Nível do depósito: 1

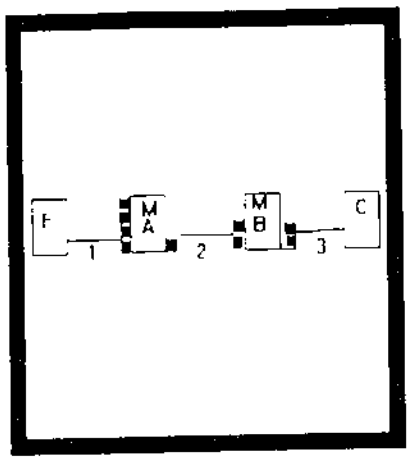


Depois

Lucro: -2 (prejuízo)
Número de Pedidos: 12
Cliente: Satisfeito
Nível do depósito: 22

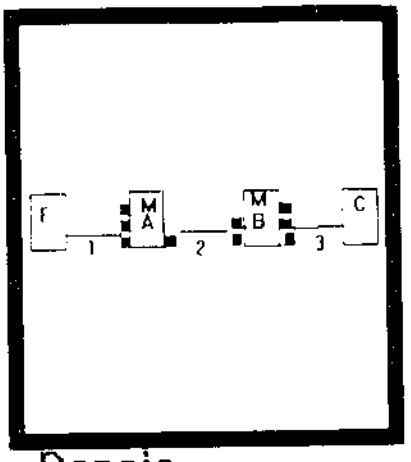
2

Alterou: aumentou estoque segurança do depósito



ANTES

Lucro: 1
Número de Pedidos: 9
Cliente: Satisfeito
Nível do depósito: 21

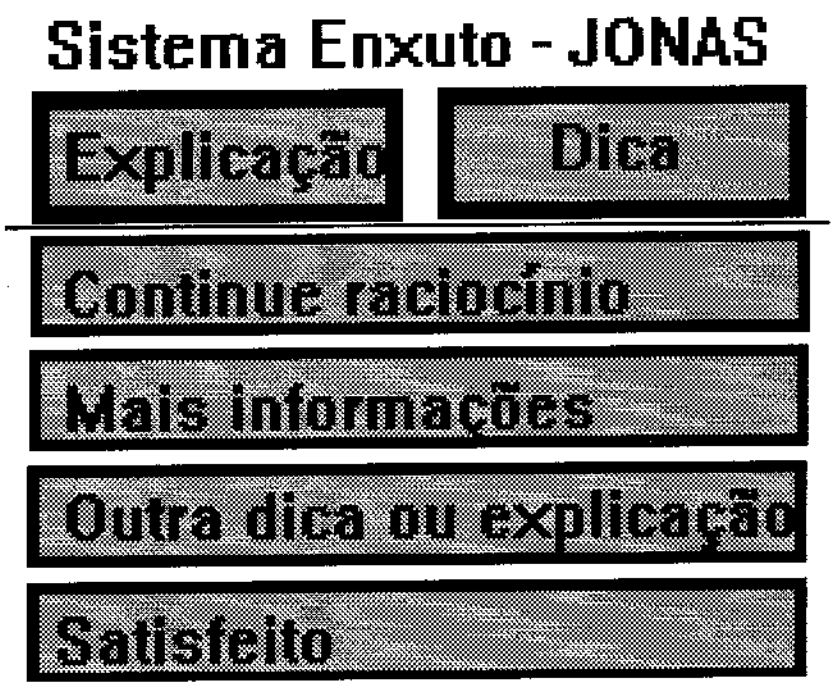


Depois

Lucro: 1
Número de Pedidos: 5
Cliente: Indiferente
Nível do depósito: 22

3

Alterou: Aumentou o preço final do produto


Sistema Enxuto - JONAS
Explicação
Dica
Continue raciocínio
Mais informações
Outra dica ou explicação
Satisfeito

Apêndice 2

O segundo experimento: *check-list*, instruções para a dupla e figuras utilizadas

Experimento de 15/07/96 - GMB- Harrison

Check-list

1) Sera 1 dupla de funcionarios que serão usuarios tipicos de produção (da própria GMB) Para a dupla haverá instruções escritas O experimento sera realizado no ambiente de trabalho, num local não sujeito a interrupção Tudo será gravado em video

2) Indicar que o teste é do produto e não das pessoas que o farão Exibir uma idéia geral da observação (testar a interface do Jonas)

3) Indicar que eles podem desistir a qualquer momento

4) Apresentar o que esta na sala. equipamentos para gravação e computador

5) Pedir para que se pense alto, pois todo pensamento é importante

6) Explicar que durante o experimento não havera ajuda e que todas as dúvidas serão sanadas ao final

7) Explicar o que e o Enxuto e o Jonas Apresentar de forma generica os dois sistemas no computador Explicar os passos que devem ser seguidos Entregar instruções escritas

8) Perguntar se não há mais questões Iniciar o exercicio

9) Concluir as observações
   explicar os objetivos da observação
   responder qualquer questão
   discutir qualquer coisa interessante que foi percebida, pedindo explicações
   perguntar qual é a impressão geral

Instruções para a dupla

Tarefas
Objetivos
1 (20 minutos) Um objetivo e entender o que aconteceu entre uma simulação e outra (com os dados apresentados nas folhas entregues) Coloque esta explicação na folha "explicando mudanças"
2 (20 minutos) O outro objetivo e alterar o último modelo buscando melhorias no resultado Indique as alterações na folha "buscando melhorias", indicando onde e como mudar
3 (5 minutos) Indique, na folha, se houveram informações que precisariam ser passadas mas não foram (resultados das simulações)
4 (5 minutos) Ao final, escrevam o que vocês acharam desta experimentação e do futuro sistema, do que gostaram e do que não gostaram

Conseguindo informações

Selecione no Jonas se vocês desejam analisar os resultados da mudança no modelo ou buscar melhorias no estado atual da fábrica Uma vez selecionada uma destas informações, vocês receberão uma resposta Desejando mais dados, indique em um dos botões o que vocês desejam. Continuem apertando os botões e recebendo as respostas até conseguirem cumprir os objetivos acima Vocês não precisam, e devem evitar, que o Jonas chegue ao raciocínio final Devem perceber a linha de raciocínio e prever resultado dele sozinhos

Explicações sobre os botões
Primeira tela

- Mudei o modelo quais foram as consequências ? explica o porquê dos resultados diferentes obtidos nas duas simulações Baseia-se nas alterações feitas entre elas Só indica caminhos de raciocínio, não apresenta a resposta
- Estado atual da fábrica indica pontos que podem ser melhorados no último modelo, buscando resultados de maior qualidade na simulação Também só indica caminhos de raciocínio, não apresenta a resposta.
- O que é ? explica a palavra apresentada.
- Tchau sai do sistema Jonas (não usar no teste)

Opção Mudei o modelo

- Explique a alteração selecionada botão que aparece após você selecionar uma alteração Apresenta explicações para esta alteração
- Continue continua apresentando informações do raciocínio iniciado

---

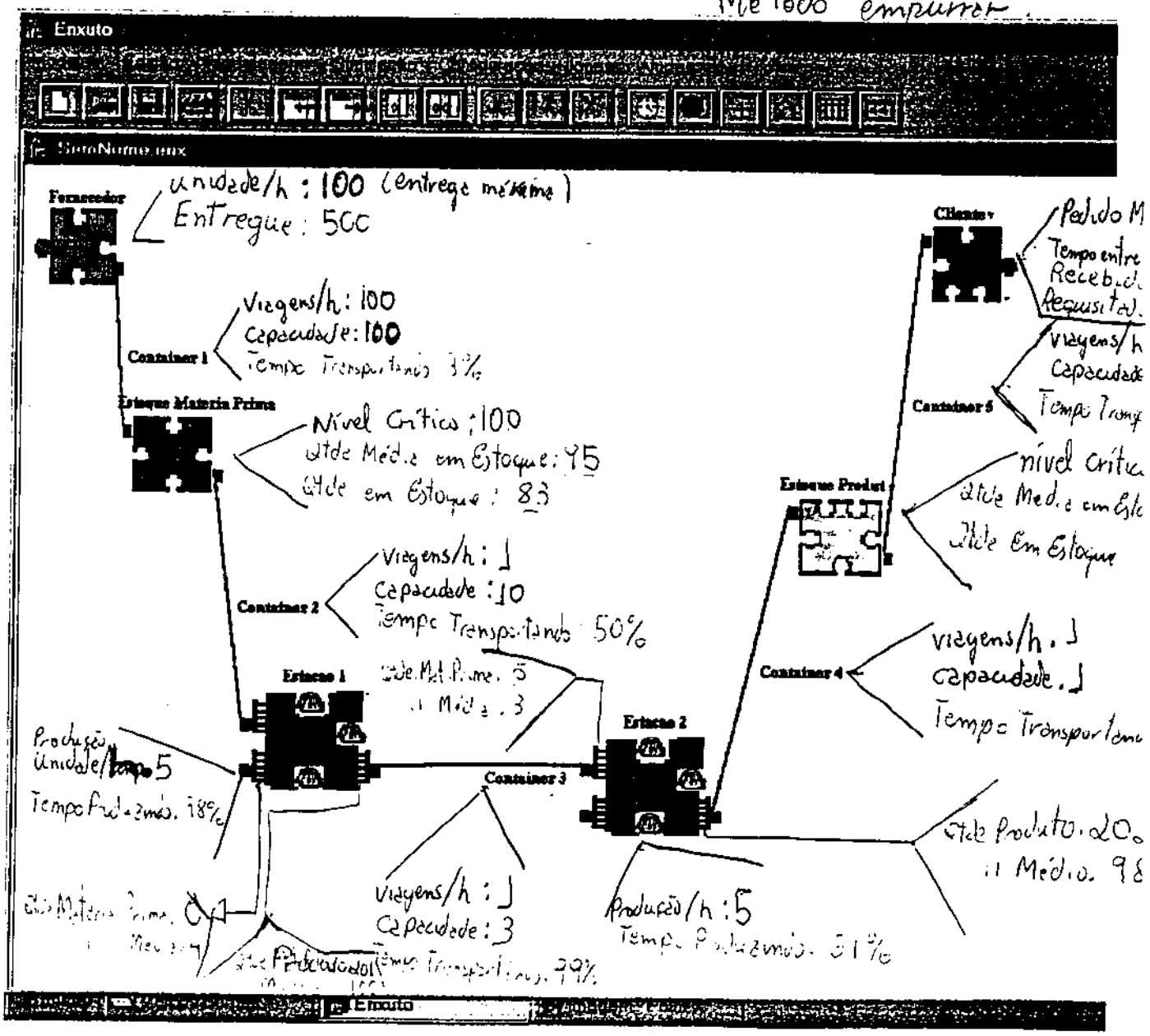
Opção Estado atual da fábrica

- Explique a alteração selecionada botão que aparece após você selecionar um resultado final Apresenta melhorias que podem ser identificadas a partir deste resultado

Continue continua apresentando informações do raciocínio iniciado

- regulado na modelagem
- resultado da simulação

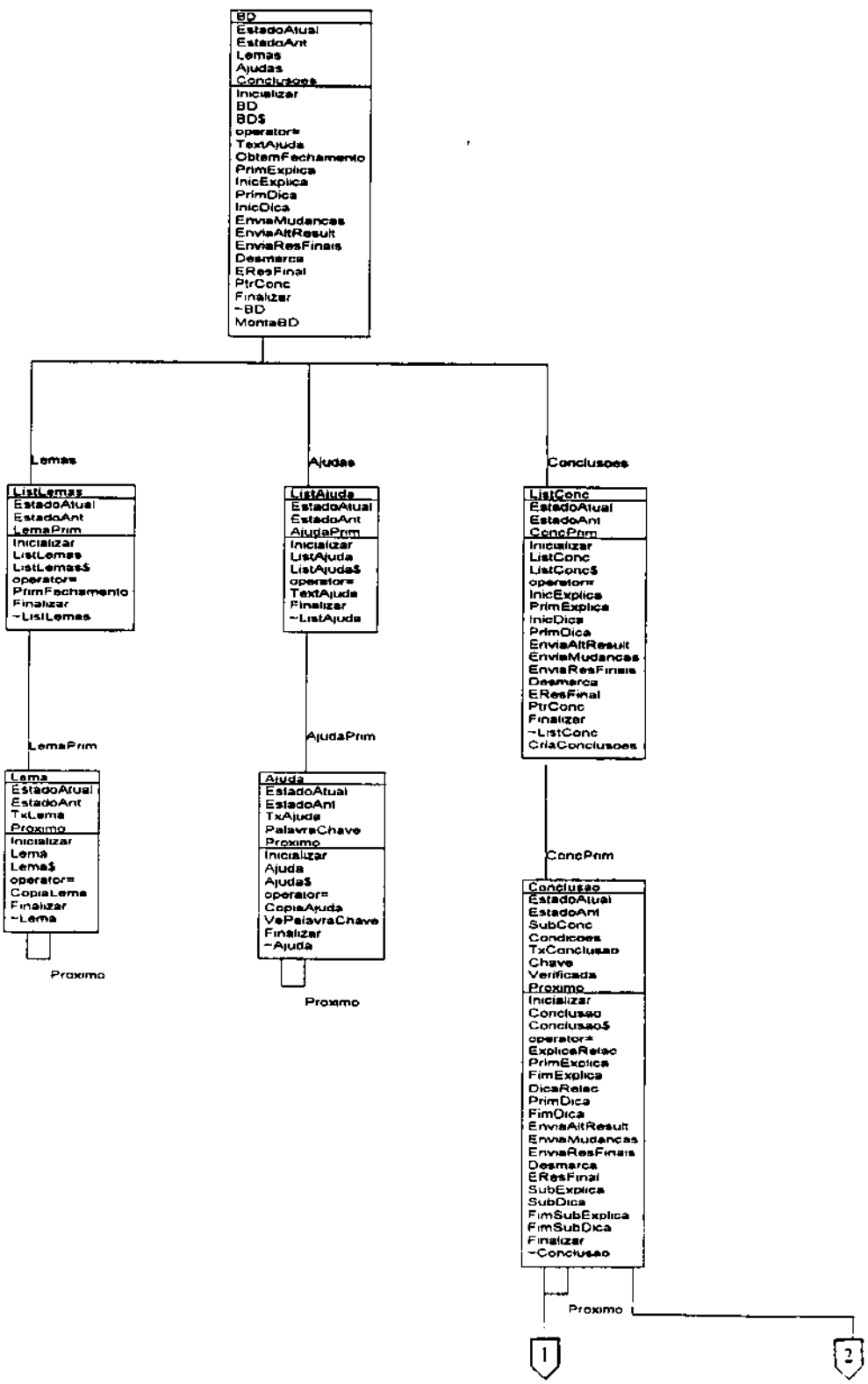
Simulado 100h.
Método empurrar.

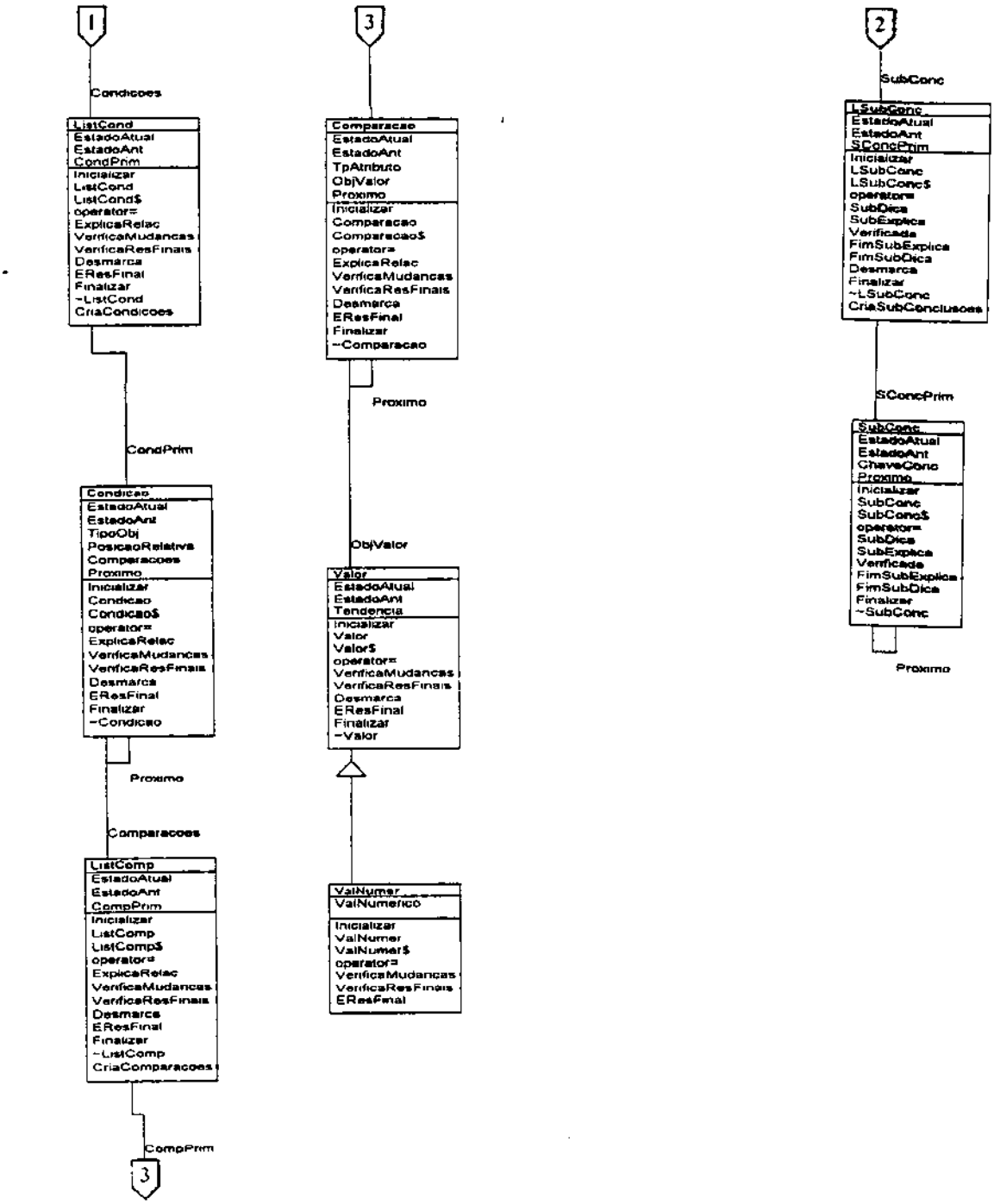


Apêndice 3

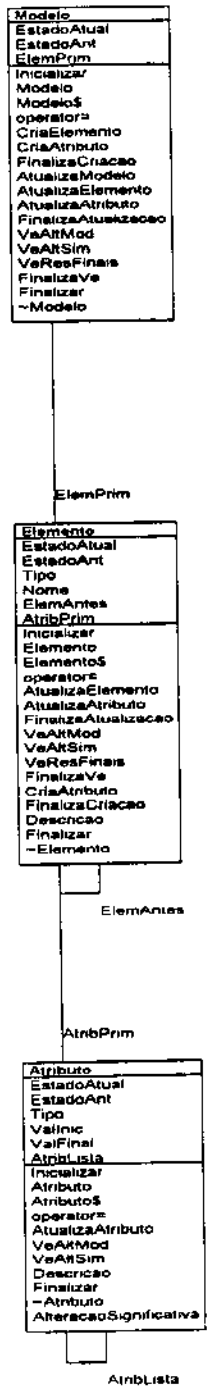
Modelos de Objetos (metodologia OMT)

Estrutura que representa a base de conhecimento

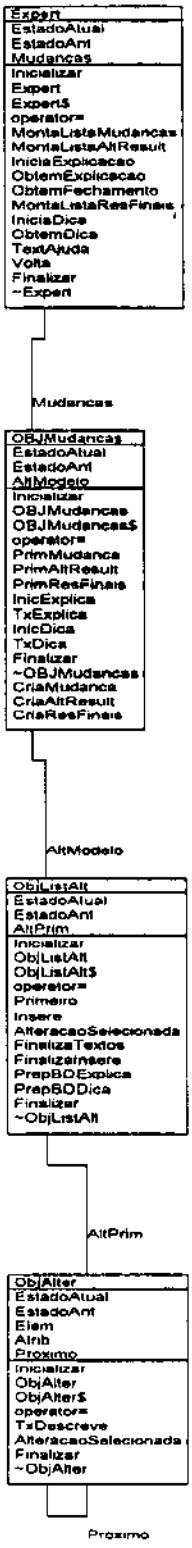



1
Condicoes
ListCond
EstadoAtual
EstadoAnt
CondPrim
Inicializar
ListCond
ListCond$
operator=
ExplicaRelac
VerificaMudancas
VerificaResFinais
Desmarca
EResFinal
Finalizar
~ListCond
CriaCondicoes
CondPrim
Condicao
EstadoAtual
EstadoAnt
TipoObj
PosicaoRelativa
Comparacoes
Proximo
Inicializar
Condicao
Condicao$
operator=
ExplicaRelac
VerificaMudancas
VerificaResFinais
Desmarca
EResFinal
Finalizar
~Condicao
Proximo
Comparacoes
ListComp
EstadoAtual
EstadoAnt
CompPrim
Inicializar
ListComp
ListComp$
operator=
ExplicaRelac
VerificaMudancas
VerificaResFinais
Desmarca
EResFinal
Finalizar
~ListComp
CriaComparacoes
CompPrim
3
3
Comparacao
EstadoAtual
EstadoAnt
TpAtributo
ObjValor
Proximo
Inicializar
Comparacao
Comparacao$
operator=
ExplicaRelac
VerificaMudancas
VerificaResFinais
Desmarca
EResFinal
Finalizar
~Comparacao
Proximo
ObjValor
Valor
EstadoAtual
EstadoAnt
Tendencia
Inicializar
Valor
Valor$
operator=
VerificaMudancas
VerificaResFinais
Desmarca
EResFinal
Finalizar
~Valor
ValNumer
ValNumerico
Inicializar
ValNumer
ValNumer$
operator=
VerificaMudancas
VerificaResFinais
EResFinal
2
SubConc
LSubConc
EstadoAtual
EstadoAnt
SConcPrim
Inicializar
LSubConc
LSubConc$
operator=
SubDica
SubExplica
Verificada
FimSubExplica
FimSubDica
Desmarca
Finalizar
~LSubConc
CriaSubConclusoes
SConcPrim
SubConc
EstadoAtual
EstadoAnt
ChaveConc
Proximo
Inicializar
SubConc
SubConc$
operator=
SubDica
SubExplica
Verificada
FimSubExplica
FimSubDica
Finalizar
~SubConc
Proximo

Estrutura que representa modelo e resultados da simulação



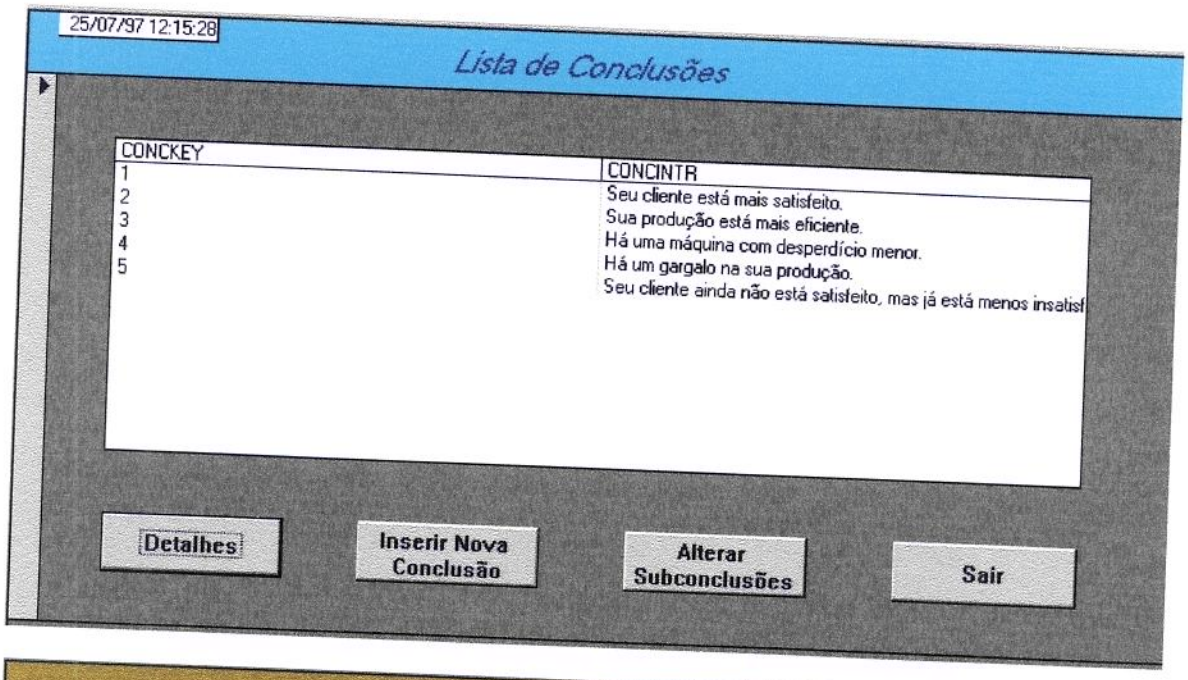
Estrutura da máquina de inferência

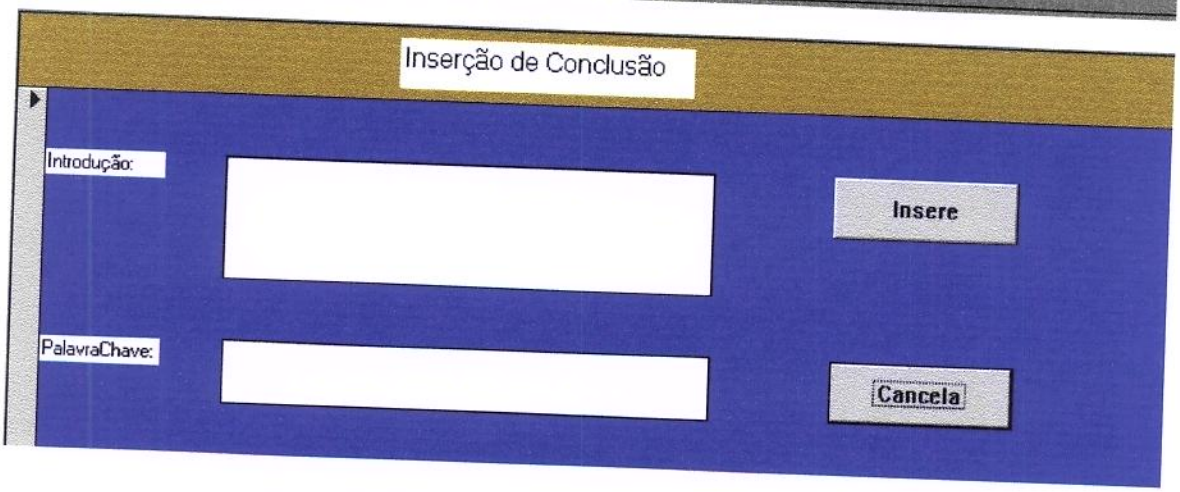


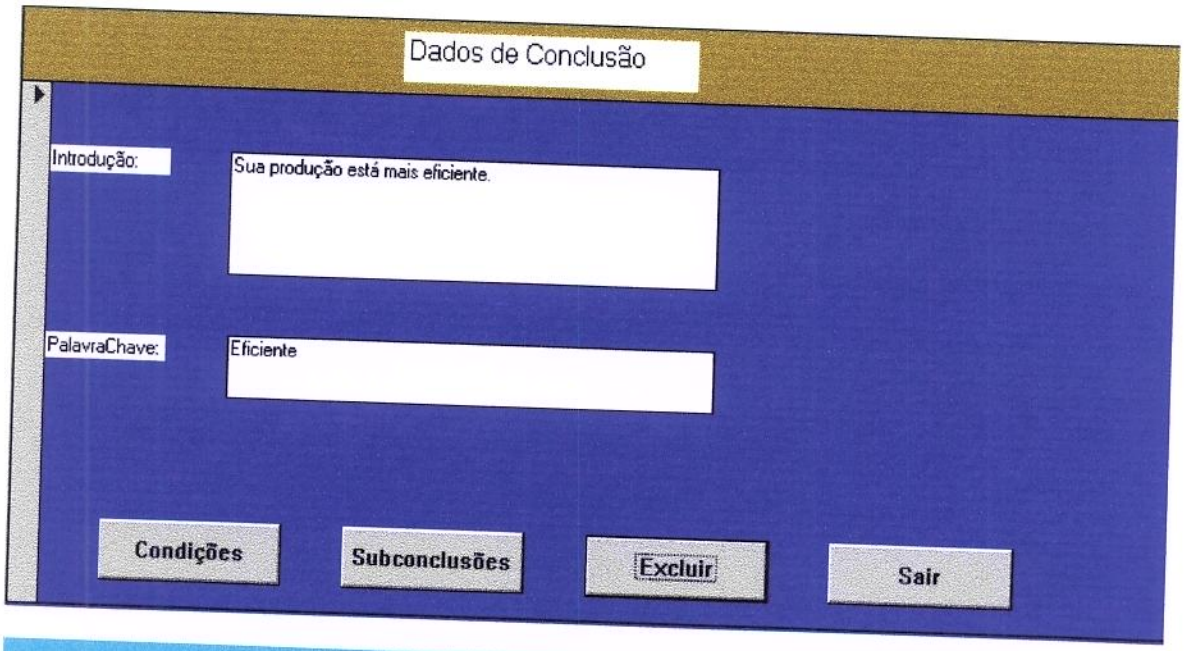
Apêndice 4

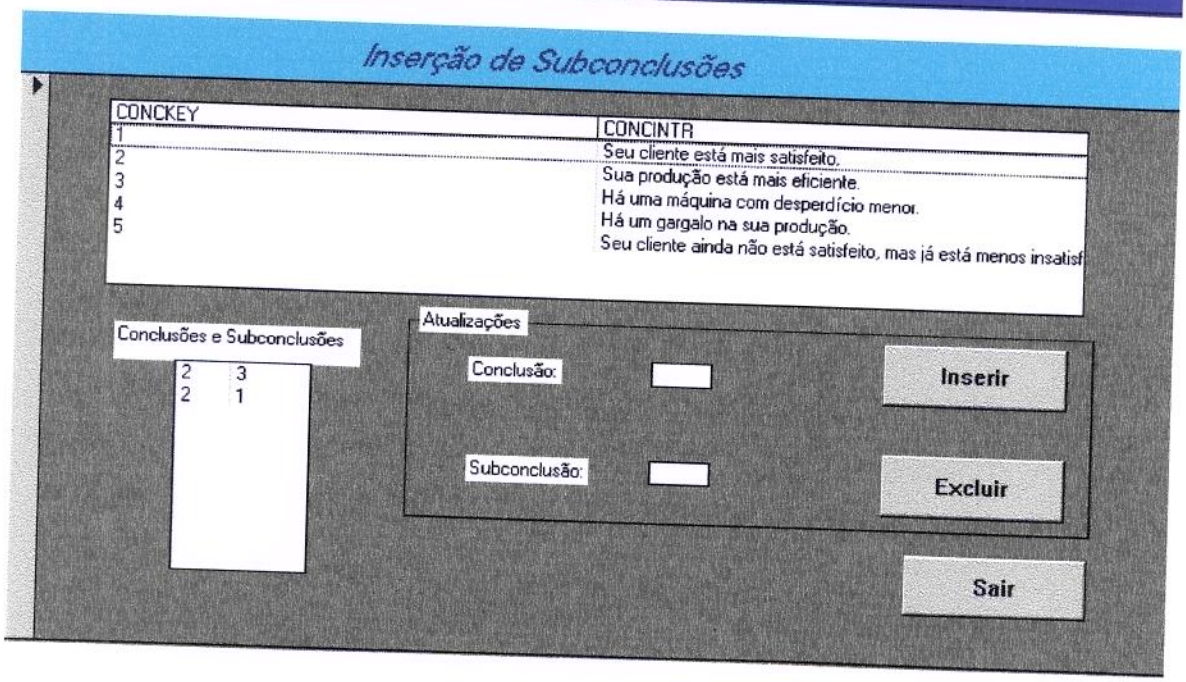
Subsistema Administrativo

Exemplos de formulários do subsistema administrativo






Dados de Conclusão
Introdução:
Sua produção está mais eficiente.
PalavraChave:
Eficiente
Condições
Subconclusões
Excluir
Sair


Inserção de Subconclusões
CONCKEY
CONCINTR
1
Seu cliente está mais satisfeito.
2
Sua produção está mais eficiente.
3
Há uma máquina com desperdício menor.
4
Há um gargalo na sua produção.
5
Seu cliente ainda não está satisfeito, mas já está menos insatisf
Conclusões e Subconclusões
2 3
2 1
Atualizações
Conclusão:
Subconclusão:
Inserir
Excluir
Sair

Exemplos de relatórios do subsistema administrativo

25/07/97 12:55:01

Sistema Jonas - Subsistema Administrativo

Lista dos Textos de Ajuda Cadastrados

Palavra Chave	Explicação
Cliente	Entidade que compra nossos produtos. Nossa razão de existir

---

25/07/97 13:00:28

Sistema Jonas - Subsistema Administrativo

Lista dos Lemas de Qualidade Cadastrados

Lemas
Melhorar a qualidade é dever de todos