Um modelo de desenvolvimento de sistemas para suporte a cooperação fundamentado em Design Participativo e Semiótica Organizacional

Rodrigo Bonacin

Tese de Doutorado

Instituto de Computação
Universidade Estadual de Campinas

Um modelo de desenvolvimento de sistemas para suporte a cooperação fundamentado em Design Participativo e Semiótica Organizacional

Rodrigo Bonacin
Março de 2004

Banca Examinadora:

Profª. Drª. Maria Cecília Calani Baranauskas (Orientadora)
Instituto de Computação – UNICAMP

Prof. Dr. Kecheng Liu
Department of Computer Science – University of Reading - UK

Profª. Drª. Clarisse Sieckenius de Souza
Departamento de Informática – PUC - RJ

Prof. Dr. Ricardo Ribeiro Gudwin
DCA – Faculdade de Engenharia Elétrica – UNICAMP

Profª. Drª. Ariadne Maria Brito Rizzoni Carvalho
Instituto de Computação – UNICAMP

Profª. Drª. Eliane Martins
Instituto de Computação – UNICAMP

Prof. Dr. Jacques Wainer (Suplente)
Instituto de Computação – UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Bonacin, Rodrigo

B64u Um modelo de desenvolvimento de sistemas para suporte a cooperação fundamentado em design participativo e semiótica organizacional /Rodrigo Bonacin -- Campinas, [S.P. :s.n.], 2004.

Orientadora : Maria Cecília Calani Baranauskas.

Tese (Doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1.Interação homem-máquina. 2.Semiótica e computação. 3.Grupos de trabalho – Processamento de dados. 4.Engenharia de software. I. Baranauskas, Maria Cecília Calani, 1954-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Um modelo de desenvolvimento de sistemas para suporte a cooperação fundamentado em Design Participativo e Semiótica Organizacional

Este exemplar corresponde à redação da
Tese de Rodrigo Bonacin

Campinas, 22 de março de 2004.

Profª. Drª. Maria Cecília Calani Baranauskas
(Orientadora)

Tese apresentada ao Instituto de computação, UNICAMP, como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

A minha esposa Juliana
que sempre me apoiou

Agradecimentos

Agradeço a Deus.

A minha esposa, a minha mãe e demais familiares que me apoiaram em todas as dificuldades que tive durante estes quatro anos de doutorado.

Especialmente a minha orientadora Professora Maria Cecília Calani Baranauskas, por sempre confiar em meu trabalho, por sua dedicação e profissionalismo como orientadora e pesquisadora, por suas palavras de incentivo e por suas críticas e sugestões. Aos seus demais orientados que contribuíram com sugestões e discussões em nossas reuniões nas quartas feiras.

Aos amigos que me incentivaram e contribuíram diretamente ou indiretamente para a realização deste trabalho.

Ao Grupo de estudo DAFE e ao Núcleo de Informática Aplicada à Educação NIED - UNICAMP.

Ao Professor Kecheng Liu, por suas críticas, sugestões, idéias e prontidão durante o período em que estive na *University of Reading*. Aos membros do AIS LAB (*Applied Informatics with Semiotics*) com os quais tive reuniões e discussões que foram valiosas para o desenvolvimento deste trabalho.

À Delphi-Automotive Systems pela parceria e por acreditar no projeto e em nossa capacidade. Em especial aos funcionários Ivan V. Ferreira, Márcio Villalva e Rafael Martinez Cecília.

Durante o meu doutorado tive apoio financeiro do Instituto de Computação-Unicamp, CNPq (140868/00-3), CAPES (2214/02-4) e FAPESP (2000/05460-0).

Resumo

Informação e comunicação são, no contexto atual, as bases para a inovação em organizações. As tecnologias da informação e comunicação, que viabilizam e sustentam processos de trabalho, têm um papel fundamental, portanto, nas organizações atuais. A abordagem do design de sistemas baseados nessa tecnologia é um ponto crucial para adequação do artefato tecnológico ao contexto organizacional. Isto é particularmente verdade no desenvolvimento de sistemas de suporte ao trabalho cooperativo (CSCW - Computer Supported Co-Operative Work), onde o uso do sistema é parte do dia a dia do trabalho desempenhado em uma organização. A abordagem proposta nesta tese lida com dois aspectos que têm impacto direto no design de sistemas para o contexto organizacional: (1) a participação do trabalhador no design do sistema e no desenvolvimento de um novo modelo organizacional, e (2) a representação do sistema e da organização por meio de um mesmo método. Embora o uso de técnicas participativas forneça mecanismos para capturar conceitos do contexto organizacional, resultados do uso destas técnicas não são necessariamente representados adequadamente nos métodos convencionais para a modelagem de sistemas. Para tornar possível a modelagem de aspectos que incluem diferentes interesses, origens e julgamentos de trabalhadores e designers, e facilitar a comunicação entre designers e usuários nós propusemos a integração e a adaptação de métodos de Design Participativo (DP) com métodos da Semiótica Organizacional (SO). SO é um ramo da Semiótica, que compreende toda a organização como um sistema de signos e possui o objetivo de estudar estas organizações utilizando os conceitos e métodos da Semiótica. Para promover o uso conjunto de DP e SO esta tese discute a possível integração entre DP e SO e viabiliza esta integração propondo um método de design que articula as técnicas de ambas as áreas para possibilitar o design de um contexto organizacional no qual o sistema computacional será parte integrante. Nesta tese também são propostos novos procedimentos para construir o sistema a partir dos aspectos modelados pela Semiótica Organizacional. Para avaliar e refinar a abordagem proposta foi conduzido um estudo envolvendo uma parceria entre nossa universidade e uma organização de manufatura que adota o sistema de produção

enxuta. Como resultado deste estudo de caso foi construído o Pokayoke, um sistema de CSCW que auxilia e gerencia o processo de resolução de problemas no cotidiano da fábrica. Resultados deste estudo de caso são analisados e apontam para a eficácia da abordagem proposta

Abstract

Information and communication are the basis for innovation in companies nowadays. The information and communication technologies allow and support the work process, therefore they have a fundamental role in the new organisations. The design approach for these technologies is a crucial point to the adequacy of the technological artefact in organisational contexts. This aspect is more evident when we consider the design of CSCW (Computer Supported Co-Operative Work) systems because the use of the computational artefact is part of the work practices. In this thesis we propose an approach that deals with two important aspects that impact the design of systems for the organisational context: (1) *The participation of the staff in the design of the system and in the development of a new organisational model* and (2) *the representation of both system and organisation with the same modelling method*. Although participatory techniques provide mechanisms to capture important concepts of the organisational context, some results of the use of these techniques are not well represented by traditional methods of system modelling. Therefore, to model aspects which include different interests and background, and the diversified judgements of the workers and practitioners, we proposed the integration and adaptation of the Participatory Design (PD) methods with Organisational Semiotics (OS). OS is a branch of Semiotics, which understands the whole organisation as a semiotic system and aims to study organisations using concepts and methods from Semiotics. To promote the integrated use of Participatory Design (PD) and OS techniques this thesis discusses the possibilities of connecting PD and OS concepts. A design method exploring techniques of PD and OS is proposed with the objective of constructing an organisational context integrating the computational system. We also propose new procedures to construct the system considering the human aspects modelled by the OS techniques. With the aim of evaluating the proposed approach a case study was conducted with the partnership of a manufacturing organisation which adopts the lean production paradigm. As a result of this case study Pokayoke, a CSCW system that supports and manages the problem resolution process in the organisation was constructed. Results of the case study indicate the efficacy of the proposed approach.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AS - *Análise Semântica*
AN - *Análise de Normas*
CSCW - *Computer Supported Cooperative Work*
CISP - *Cooperative Interative Storyboard Prototyping*
DP - *Design Participativo*
HOOTD - *Hierarchical Object-Oriented Task Decomposition*
ICE – *Interface Configuration Environment*
IHC - *Interface Humano Computador*
MEASUR -*Method for Eliciting, Analysing and Specifying User Requirements*
NBIC - *Norm-Based Interface Configurator*
OO - *Orientado a Objetos*
PAM - *Problem Articulation Method*
P&D - *Pesquisa e Desenvolvimento*
PHE - *Participatory Heuristic Evaluation*
RH - *Recursos Humanos*
RUP - *Rational Unified Process*
SI - *Sistema de Informação*
SO - *Semiótica Organizacional*
SPaM - *Semiotic Participatory Method*
UML - *Unified Modelling Language*
UP - *Unified Process*

Capítulo 1

Introdução

No contexto atual, informação e comunicação são os novos caminhos para inovação em organizações. Em conseqüência disto, várias organizações são levadas a investir cada vez mais no desenvolvimento e/ou na aquisição de sistemas computacionais, uma vez que eles potencialmente oferecem suporte aos funcionários destas organizações para criar, gerenciar, trocar, acessar e visualizar informações que são cruciais para a manutenção da competitividade. Entretanto, para que estes benefícios se concretizem é necessário que estes sistemas sejam desenhados e desenvolvidos de maneira adequada.

A literatura em sistemas de informação e engenharia de software tem destacado que não são raros os casos de falhas em projetos de desenvolvimento de sistemas computacionais (Liu, 2000; Booch, 1999). Com o objetivo de abordar problemas relacionados ao desenvolvimento de sistemas, pesquisadores têm proposto novos métodos e ressaltado que o design de sistemas para o contexto organizacional encontra-se muito longe de ser um problema solucionado.

Nesta tese abordamos a problemática ligada ao desenvolvimento de sistemas computacionais capazes de apoiar o trabalho em grupo nas organizações. A área de pesquisa conhecida como CSCW (*Computer Supported Co-Operative Work*) estuda a concepção, o desenvolvimento e a utilização destes sistemas. De acordo com Bannon (1993), organizações buscam alternativas para organizar e coordenar melhor as atividades de trabalho em grupo e enxergam CSCW como uma oportunidade para aprimorar a colaboração e coordenação via sistemas de informação flexíveis que estão disponíveis em qualquer lugar e tempo.

Para Grudin “muitas das falhas no desenvolvimento e na comercialização de sistemas para suporte a grupos não são decorrentes de problemas técnicos. Elas são resultados da não compreensão da demanda única que esta classe de software impõe aos designers e usuários” (Grudin, 1994, p. 92). Esta “demanda” destacada por Grudin é decorrente de que, no desenvolvimento de sistemas para serem utilizados por grupos de pessoas, devem ser considerados não só os aspectos técnicos, mas também aspectos humanos e sociais, ligados à pesquisa interdisciplinar. Bannon e Hughes (1993, p. 26) sumarizaram a problemática do design de sistemas de CSCW em três necessidades: “(1) estabelecer referências com o “mundo real” para compreender a natureza das atividades cooperativas, (2) relacionar a análise das atividades cooperativas com o design do sistema e (3) permitir que os usuários informem o processo de design”.

Para lidar com fatores humanos e sociais, pesquisadores têm proposto várias abordagens para o desenvolvimento de sistemas de CSCW. A principal motivação desta tese é o aprimoramento no desenvolvimento e uso de sistemas de CSCW por meio de um método de design que explora aspectos ainda não abordados na pesquisa e desenvolvimento de sistemas para suporte ao trabalho cooperativo.

Assumimos que o valor real de um sistema computacional não é a tecnologia por si própria, mas a capacidade do sistema de aprimorar o contexto organizacional. Conseqüentemente, o artefato computacional deveria ser construído como parte integrada de toda a organização, considerando seus aspectos humanos e sociais. Isto é particularmente verdade para o desenvolvimento de sistemas de suporte ao trabalho cooperativo, onde o uso do sistema é parte do dia a dia do trabalho realizado em uma organização.

A integração sistema-organização pode ser conduzida por diferentes abordagens. Por um lado, a organização poderia ser adaptada ao sistema computacional impondo novas práticas de trabalho, que poderiam ou não ser apropriadas ao contexto organizacional; ou, por outro lado, o sistema poderia simplesmente automatizar as práticas atuais sem se preocupar com o seu aprimoramento. Nesta tese, é proposta uma abordagem para o desenvolvimento de sistemas computacionais, em que os designers, em cooperação com os trabalhadores da organização, devem pensar no sistema de forma integrada a um novo contexto organizacional do qual o sistema computacional será parte.

A abordagem proposta lida com dois aspectos que têm impacto direto no design de sistemas para o contexto organizacional:

1. *A participação do trabalhador no design do sistema e no desenvolvimento de um novo modelo organizacional.* Os usuários (trabalhadores) possuem conhecimento sobre o contexto organizacional e a prática de trabalho no nível de detalhamento necessário para discutir e modelar um novo contexto que incluirá o sistema computacional;
2. *A representação do sistema e da organização por meio de um mesmo método.* Um método que represente as relações entre o sistema e outros aspectos da organização é necessário para definir os requisitos do sistema e construir uma visão compartilhada da organização pelos designers e pelos trabalhadores permitindo assim coordenar seus esforços.

Esta tese objetiva integrar técnicas baseadas em Design Participativo (DP) e Semiótica Organizacional (SO) em um método de design único, para lidar com aspectos ligados ao design de sistemas para organizações. Para tanto, são propostas novas técnicas e procedimentos.

O Design Participativo (DP) teve início na Escandinávia na década de 70 e emprega várias técnicas para conduzir o design "com" o usuário, em vez de "para" o usuário. O Design Participativo enfatiza a importância da democracia no ambiente de trabalho para aprimorar seus métodos, a eficiência no processo de design (considerando o conhecimento do usuário), a qualidade dos sistemas, e tende a impulsionar as atividades de formação (Müller e outros, 1997). No Design Participativo estes objetivos são atingidos por meio da interação direta dos usuários com os designers durante todo o ciclo de desenvolvimento da aplicação computacional, atribuindo ao usuário algum controle sobre as decisões de design.

Embora o uso de técnicas participativas forneça mecanismos para capturar conceitos importantes do contexto organizacional, alguns resultados do uso destas técnicas não possuem representação adequada nos métodos convencionais (de engenharia de software) para a modelagem de sistemas. Os aspectos humanos que influenciam na maneira como os sistemas deveriam ser modelados não são captados pelos métodos

convencionais, que foram inicialmente elaborados para modelar sistemas de propósito geral e possuem foco nas funções do sistema e do negócio (Liu, 2000).

Do ponto de vista filosófico, o paradigma predominante nos métodos convencionais para o design de sistemas é o objetivismo, que é frágil quando aplicado a organizações humanas. De acordo com Liu (2000), diferentes aspectos políticos, interesses econômicos e a origem cultural resultam em julgamentos diversificados em contraposição a uma conclusão comum baseada em uma "verdade objetiva".

Conseqüentemente, para tornar possível a modelagem de aspectos que incluem diferentes interesses, origens e julgamentos de trabalhadores e designers, nós estamos propondo a integração e a adaptação de métodos de DP com métodos da Semiótica Organizacional (SO). SO é um ramo da Semiótica que compreende toda organização como um sistema de signos e possui o objetivo de estudar estas organizações utilizando os conceitos e métodos da Semiótica (OSW, 1995).

Com o objetivo de modelar o contexto social, a SO adota um paradigma subjetivista/construtivista que vê a realidade como uma construção social baseada no comportamento de agentes. Nesta visão, pessoas compartilham padrões de comportamento que constituem um sistema de normas. Como as pessoas estão constantemente comunicando-se e discutindo, o mundo está sempre em mudança. Os dois axiomas básicos nos quais as metodologias de SO estão fundamentadas podem ser expressos como: "não existe conhecimento sem um conhecedor, e não existe compreensão sem ação" (Liu, 2000, p. 26).

DP complementa a SO na abordagem proposta neste trabalho como instrumento de captura dos signos que constituem o contexto organizacional. DP disponibiliza métodos que visam promover a participação do usuário no design do sistema. Esta participação é importante para explicitar como a comunicação e a interpretação dos signos afetam o contexto de trabalho.

O objetivo desta tese é propor uma abordagem para o design de sistemas computacionais no contexto organizacional, tendo como foco principal os sistemas que dão suporte ao trabalho cooperativo. Entendemos por abordagem o ato de abordar, ou seja, o ato de tratar de alguma coisa ou versar sobre um tema e assunto. Desta maneira, a palavra abordagem empregada nesta tese não está restrita a um método ou técnica, mas

constitui-se de todo o referencial teórico, prático, modelos, métodos e técnicas utilizados no ato de tratar ou versar sobre o tema ou assunto objeto.

A abordagem proposta nesta tese, baseada em DP e SO, lida com aspectos relacionados à participação e à significação no design de sistemas para organizações. A abordagem proposta foi aplicada e refinada em um estudo de caso em uma organização de manufatura real. Neste estudo de caso foi construído o sistema Pokayoke, um sistema de CSCW que dá suporte a resolução de problemas no contexto de uma organização de manufatura.

1.1 Questões da Pesquisa

A principal questão da pesquisa que esta tese aborda envolve:

- Por quê e como a Semiótica Organizacional e o Design Participativo poderiam juntos contribuir para o design de sistemas para suporte ao trabalho cooperativo no contexto organizacional?

A partir desta questão principal, podemos derivar outras questões relacionadas:

- Que aspectos trabalhados pelo DP e SO são relevantes para a construção de sistemas cooperativos?
- Por que é possível integrar conceitos de DP e SO?
- Como promover o design cooperativo através da participação e significação?
- Como desenvolver um método de design baseado em DP e SO?
- Como possibilitar a participação dos usuários na modelagem proposta pela SO?
- Como derivar modelos para implementar o sistema e fazer manutenção a partir dos resultados do uso de DP e SO ?

Para responder a estas questões, esta tese envolve vários desafios que resultam na condução de trabalhos em diversas áreas do conhecimento como: Design Participativo,

Semiótica Organizacional, Interação Humano-Computador, CSCW, Sistemas de Informações e Engenharia de Software. O trabalho envolveu:

- Investigar a viabilidade de relacionar conceitos do DP e SO. De que maneira SO poderia contribuir para a DP e de que maneira a participação do usuário poderia contribuir para a modelagem semiótica?
- Desenvolver uma abordagem para construção de um novo contexto organizacional por meio do uso de DP e SO. Esta abordagem envolve áreas relacionadas ao desenvolvimento de sistemas no contexto organizacional e sistemas para o suporte ao trabalho cooperativo;
- Desenvolver um método que englobe as técnicas de DP e SO e especifique as fases e passos que direcionam o design do sistema. É necessário modelar a organização e o sistema computacional como um todo. Este método serve também para embasamento de pesquisas futuras que tenham como foco incluir técnicas de SO em processos de desenvolvimento de software;
- Especificar uma técnica de DP que permita o design cooperativo de modelos da SO. Esta técnica envolve a pesquisa em sistemas de informação, uma vez que balancearia tanto aspectos orientados a produto quanto a processo (Floyd, 1987; Müller et al 1998; Mahemoff e Johnston, 1998). Projetos futuros poderiam ser conduzidos para incluir esta técnica em processos de engenharia de software;
- Desenvolver novos procedimentos para derivar diagramas que especificam sistemas computacionais a partir dos modelos da SO. O estudo de como cada aspecto trabalhado pela SO pode ser refletido na construção de sistemas computacionais é objeto de pesquisa da área de SO.

1.2 Paradigma da Pesquisa

Esta tese adota o paradigma *subjetivista*, que é o paradigma base para a Semiótica Organizacional (Liu, 2000). Atualmente, o paradigma predominante no desenvolvimento de software tem raízes no objetivismo, que pressupõe a existência de um mundo independente do observador e uma realidade objetiva composta por uma estrutura de

entidades pré-existentes. As Ciências Naturais representam tipicamente o paradigma objetivista, visto que o conhecimento científico é estabelecido principalmente por meio da generalização indutiva de experiências sensoriais (Filipe, 2000). O objetivismo, que tem sido bem sucedido quando aplicado às Ciências Naturais apresenta algumas fragilidades quando aplicado a organizações humanas.

Lyytinen (citado em Liu (2000, p.23)) descreve quatro princípios de deficiência do paradigma objetivista em lidar com organizações humanas: (1) princípio da correspondência, para o qual correlações fixas e imutáveis não são adequadas para tratar ambigüidades semânticas, (2) princípio da objetividade, onde observações supostamente independentes de observadores não são adequadas para descrever o contexto humano considerando as modalidades, propósitos e intenções dos observadores, (3) princípio do terceiro excluído, para o qual a exclusão do meio termo em que 'toda assertiva é verdadeira ou falsa' torna difícil de aplicar este paradigma a situações do dia a dia onde não é possível estabelecer o conceito de uma verdade única, e (4) princípio da neutralidade lingüística, onde a linguagem simplesmente significa a representação utilizada para descrever o mundo, não englobando também a maneira como a linguagem modifica o mundo.

De acordo com Liu (2000) diferentes interesses políticos e econômicos, e diferenças culturais conduzem a julgamentos diversificados em vez de uma conclusão comum baseada em uma verdade absoluta. Na abordagem proposta pelo Design Participativo, os trabalhadores e designers possuem interesses e conhecimentos distintos; é essencial que modelos de design considerem julgamentos diversificados. Além disso, sistemas de CSCW dão suporte a várias atividades que incluem pessoas que têm diferentes interesses, experiências e conhecimentos; conseqüentemente modelos de CSCW deveriam representar estes conceitos diversificados com o propósito de dar suporte às atividades.

No paradigma subjetivista a realidade compartilhada é o ponto inicial para discussão de diferentes conceitos. Este paradigma enfatiza a habilidade dos indivíduos, sua liberdade de escolher suas ações e sua responsabilidade moral sobre as escolhas. Os comportamentos afetivos e cognitivos são considerados e a realidade é construída a partir da experiência. A Tabela 1.1 sumariza e compara posições típicas dos paradigmas

subjetivista e objetivista em alguns conceitos-chave, relacionados ao trabalho em sistemas de informações.

Tabela 1.1 - Visões objetivista e subjetivista de alguns conceitos-chave no desenvolvimento de sistemas computacionais (Liu,2000, p. 25)

Conceito	**Perspectiva Objetivista**	**Perspectiva Subjetivista**
Realidade	A mesma para todos, composta de entidades, suas propriedades e relacionamentos	Criada subjetivamente e socialmente, com diferenças sutis entre os grupos de agentes
Dado	Um meio de representar a realidade	Um meio de indicar intenções e coordenar ações
Verdade	Uma correspondência correta entre as entidades reais	Um consenso atingido (temporariamente) como uma base para uma ação coordenada
Significado	Um relacionamento entre um signo e alguma entidade real	Um relacionamento entre um signo e algum padrão de ações estabelecido como uma norma de um grupo
Sistemas de Informações	Um tipo de "sistema de encanamento" por onde passa o fluxo de dados	Um sistema semiótico, principalmente informal mas suplementado por mensagens formais
Papel do analista	Especificar as estruturas e funções de dados reais que são necessários para os usuários	Ajudar os usuários a articularem seus problemas, descobrirem suas necessidades de informações e desenvolverem uma solução sistêmica

1.3 Método da Pesquisa

O método da pesquisa aplicado neste trabalho incluiu um estudo sistemático de Design Participativo, Semiótica Organizacional e sistemas de CSCW em contextos reais. Este estudo levou a uma abordagem de design analisada em um estudo de caso em uma organização real. Este estudo de caso resultou em modelos e protótipos do sistema Pokayoke que foram submetidos a avaliações participativas.

A Figura 1.1 ilustra elementos da pesquisa desenvolvida. Nela representamos o método de design proposto, que envolve aspectos provenientes do estudo das seguintes áreas: Design Participativo, Semiótica Organizacional, CSCW e IHC e Engenharia de Software. Este método, por sua vez, foi utilizado durante o desenvolvimento do sistema Pokayoke. O sistema e o método de design utilizado foram analisados várias vezes durante o estudo de caso em busca do refinamento da abordagem proposta nesta tese.

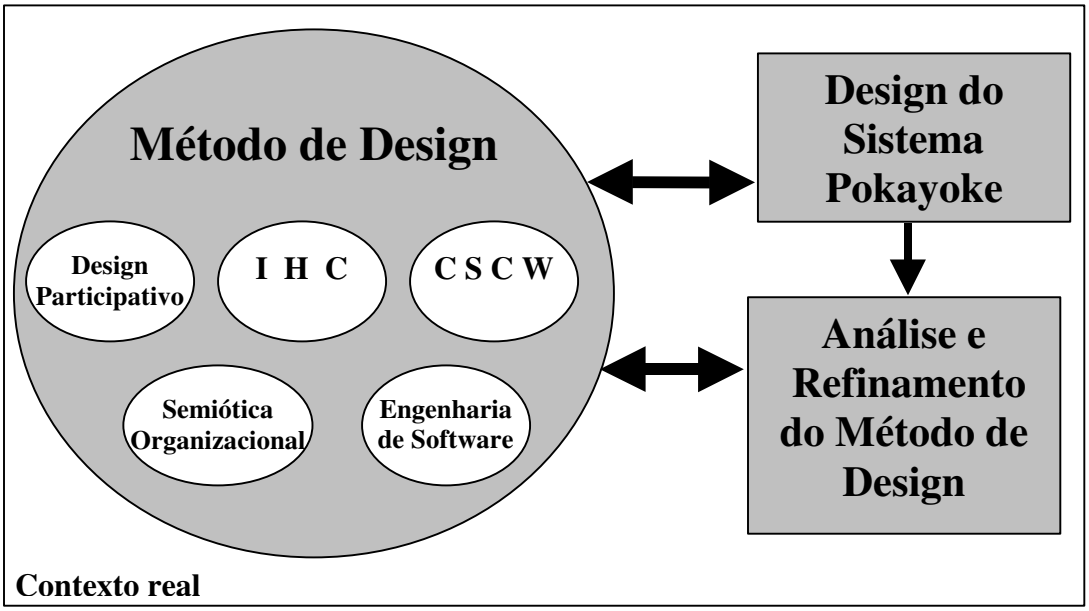


Figura 1.1 - Elementos da Pesquisa

1.4 Histórico do Projeto na Organização em questão

Este trabalho foi parte de um projeto de cooperação entre o Núcleo de Informática Aplicada à Educação (NIED – Unicamp) e a Delphi Automotive Systems - Jaguariúna (Nied, 1999; Nied, 2002), que teve apoio da FAPESP. A Delphi Automotive Systems - Jaguariúna é uma organização que produz radiadores e equipamentos de ar condicionado para carros da GM e de várias outras montadoras. Seu sistema de produção tem bases nas idéias da "produção enxuta".

Estas novas indústrias, baseadas na produção enxuta, privilegiam programas de educação continuada a seus funcionários e projetos que possibilitem a aprendizagem e o entendimento do processo global de produção do ambiente onde trabalham. Estes

funcionários devem *aprender enquanto trabalham*, para estarem aptos a evoluir junto à organização. O primeiro projeto desenvolvido na parceria Universidade-Empresa visava dinamizar a formação e a aprendizagem na fábrica, a partir de ambientes de aprendizagem baseados em jogos, simulação e modelagem gerados a partir de levantamento das necessidades conceituais de formação de funcionários da linha de produção em relação a conceitos de controle estatístico de processos e conceitos de "produção enxuta". Além de resultados dentro da própria organização fabril, o projeto institucional de parceria apresentou diversos resultados de pesquisa (Baranauskas e outros, 1997; Baranauskas e outros, 1999; Baranauskas e outros, 2000; Baranauskas e outros, 2001; Baranauskas e outros, 2002; Borges e Baranauskas, 1998; Borges e Baranauskas, 2000; Borges e Baranauskas, 2002a; Borges e Baranauskas, 2002b) e de formação acadêmica (Bonacin, 2002; Borges, 1997; Schuluzen, 2000; Borges, 2000; Chebabi, 2002; Silva, 2001; Souza, 2003).

Em continuidade ao projeto referenciado (Nied, 1999) foi desenvolvido um segundo projeto de cooperação (Nied, 2002) envolvendo o design de ambientes de aprendizagem onde o foco da atividade do aprendiz era a resolução de problemas de forma colaborativa/cooperativa e contextualizada na sua realidade de trabalho, por intermédio de sistemas computacionais.

Aprendizagem deve ser entendida em um contexto mais amplo do que a aprendizagem que se refere a atividades de treinamento com o uso do computador. A aprendizagem auxiliada por sistemas computacionais referida neste projeto não se limita ao objetivo intencional de formação, mas é contextualizada no dia-a-dia de trabalho (mediado por sistemas computacionais) dos funcionários. Dentro do escopo desse projeto foi desenvolvido o ambiente Andon (Silva e Baranauskas, 2000a; Silva e Baranauskas, 2000b; Silva e outros, 2001; Silva, 2001), um ambiente de suporte à discussão e *Brainstorming* eletrônico para o tratamento de casos-problema existentes no cotidiano da fábrica.

Seguindo a filosofia da "produção enxuta", onde os funcionários são considerados elementos inteligentes dentro da fábrica, o sistema Andon possibilitava que a solução de problemas de rotina fossem realizados por pessoas que estão próximas aos problemas. Tal sistema visava estimular a colaboração entre os trabalhadores e a dinâmica

multifuncional do trabalho em grupo (Womack, 1990), possibilitando o aprimoramento contínuo das práticas de trabalho. Adicionalmente, o sistema Andon, assim como o novo sistema proposto, possibilitou o desenvolvimento de atividades formativas durante o design do sistema, pela utilização de técnicas de DP.

O sistema derivado do estudo de caso descrito nesta tese, o sistema Pokayoke, tem como ponto de partida as discussões e avaliações do sistema Andon. Este novo sistema visa integrar o processo de resolução de problemas aos procedimentos da fábrica, de acordo com a idéia de construir o sistema como parte de um novo contexto organizacional.

Rotineiramente as organizações enfrentam inúmeros problemas que exigem discussão e colaboração entre indivíduos de um grupo para resolvê-los. O sistema Pokayoke é baseado em um procedimento chamado "Cinco Passos", cujo objetivo é definir um método disciplinado para lidar com problemas de rotina da produção. As quatro principais dificuldades enfatizadas pelos trabalhadores em relação ao procedimento de "Cinco Passos" em papel eram: (1) "*O procedimento de cinco passos dificulta a participação dos funcionários devido a problemas de comunicação*", (2) "*Para utilizar algumas ferramentas em papel são necessárias reuniões longas*", (3) "*É necessário um treinamento para possibilitar que os trabalhadores utilizem as ferramentas de acordo com os procedimentos da organização.*" e (4) "*Cada pessoa armazena seus formulários de cinco passos dificultando reaproveitar as soluções*".

1.4.1 Envolvimento dos funcionários no Projeto

Após uma reunião inicial com a gerência da fábrica, foi estabelecido que o departamento de Recursos Humanos (RH) seria o responsável por viabilizar o envolvimento do trabalhador no processo de design do novo sistema. Foi estabelecido um responsável no RH, que por sua vez envolveu trabalhadores das três "plantas" da fábrica para estabelecer um procedimento comum para "Cinco Passos", visto que cada "planta" tinha o seu. O responsável teve um papel importante até que fosse definido um contato direto com os participantes.

No decorrer do processo participaram 14 funcionários da fábrica, de diferentes funções e posições hierárquicas. Estes funcionários participaram de maneira direta nas

atividades participativas e demonstraram interesse direto no design e utilização do sistema. É importante salientar o apoio da gerência que, de maneira geral, olhava as atividades como parte do trabalho do funcionário.

Quatro funcionários em diferentes etapas do desenvolvimento assumiram a responsabilidade de serem o principal contato com a universidade; eles eram responsáveis por viabilizar encontros nos quais as técnicas participativas eram aplicadas. Os funcionários também participaram da implantação do sistema de forma ativa, uma vez que foram eles próprios que treinaram os outros e estabeleceram um grupo de usuários iniciais.

1.4.2 Design Participativo e Projetos de Pesquisa e Desenvolvimento

Conforme mencionado anteriormente, o estudo que conduzimos para avaliar a abordagem proposta é parte de um projeto de parceria universidade-empresa. Quando desejamos aplicar novas metodologias a projetos de Pesquisa e Desenvolvimento (P&D) alguns aspectos relevantes como as diferenças culturais em relação ao ambiente comercial e as próprias características de um projeto de P&D devem ser considerados. A utilização do DP em conjunto com conceitos presentes nas novas metodologias para desenvolvimento de software surge como uma abordagem alternativa para a diminuição das dificuldades presentes em projetos de P&D (Bonacin e Silva, 2004).

Um projeto de pesquisa e desenvolvimento inclui tanto objetivos de uma organização (ou organizações) quanto objetivos científicos (acadêmicos). Cabe ao pesquisador em conjunto com a organização estabelecer parâmetros que garantam contribuições científicas por um lado, e econômicas para a organização, por outro lado.

Antes de uma organização investir em um projeto de P&D ela deve ser capaz de avaliar os riscos associados a estes projetos e também ser capaz de acompanhar o seu andamento. Porém em muitos casos, estas são tarefas muito complexas (em relação a projetos de desenvolvimento de software comercial), dadas as várias características de um projeto de pesquisa presentes em um projeto de P&D. Em projetos de P&D existem riscos extras além daqueles presentes em sistemas comerciais; projetos de P&D na área de Ciência da Computação podem incluir, por exemplo, a elaboração de uma nova

técnica ou a utilização de técnicas computacionais ainda inexploradas para a resolução de determinado tipo de problema. Isto dificulta a avaliação de risco por parte da organização e a confiança dela na palavra do pesquisador pode tornar-se o principal mecanismo de análise de risco e acompanhamento do projeto.

O aprimoramento da comunicação entre o(s) pesquisador(es) e a organização é fundamental para que seja desenvolvido um sistema que realmente seja válido e útil para a organização. Com este aprimoramento a empresa poderá planejar, de forma um pouco mais precisa, seus investimentos no projeto, prevendo riscos e acompanhando o seu andamento.

Para possibilitar a interação entre a organização e o(s) pesquisador(es) podem ser utilizadas técnicas de DP. Estas técnicas podem trazer indivíduos da organização como elementos ativos no andamento do projeto de P&D. Com esta interação é possível obter parâmetros para um melhor acompanhamento do projeto, avaliação dos riscos envolvidos e adequação do projeto às necessidades da organização. Por outro lado, por meio desta participação o pesquisador terá mecanismos melhores para o entendimento do problema existente na organização e seus objetivos quanto ao projeto.

Sob o ponto de vista de projetos de P&D a utilização do Design Participativo poderia trazer vários benefícios, alguns deles seriam:

- Possibilidade de acompanhamento e avaliação do projeto por parte da empresa;
- Maior compreensão do domínio objeto pelos pesquisadores, uma vez que os problemas são compartilhados entre os participantes que estão diretamente envolvidos. Por meio desta interação os pesquisadores podem colocar em prática suas propostas teóricas;
- Maior possibilidade de satisfazer objetivos da organização, uma vez que ela participa diretamente no desenvolvimento do sistema;
- Possibilidade de aprendizagem mútua e aprimoramento das práticas de trabalho;
- Maior eficiência e qualidade.

1.5 Estrutura da Tese

Esta tese está organizada em oito capítulos. O capítulo 2 apresenta a motivação para uma abordagem ao desenvolvimento de sistemas de CSCW alinhada ao tratamento de questões organizacionais. É destacada a importância dos aspectos sociais para o desenvolvimento de sistemas de CSCW e a natureza interdisciplinar da pesquisa desta área. São apresentadas várias abordagens para lidar com aspectos humanos em CSCW, assim como pesquisa para facilitar a manutenção e adaptação destes sistemas. Neste capítulo também é situada a abordagem adotada nesta tese em relação às demais propostas.

O capítulo 3 apresenta o referencial teórico das duas principais áreas de pesquisa relacionadas a esta tese: o Design Participativo e a Semiótica Organizacional. Primeiramente são apresentados os conceitos básicos do DP, seus princípios, aspectos políticos e teóricos, e as técnicas de DP utilizadas. Em seguida é apresentada a SO destacando-se seus conceitos básicos e as suas técnicas. É apresentada também a descrição e exemplos das duas técnicas de SO empregadas (a Análise Semântica e a Análise de Normas) e a notação utilizada em seus modelos.

O capítulo 4 envolve a discussão sobre as possibilidades de se relacionar conceitos do DP e da SO. Discute-se também como a Análise Semântica e de Normas podem contribuir para o Design Participativo e como a participação do usuário poderia contribuir para a modelagem semiótica. É apresentada também uma abordagem para a construção de um novo contexto organizacional a partir da utilização de DP e SO, para modelar uma organização futura.

No capítulo 5 é proposto o SPaM (Semiotic Participatory Method), um método que integra DP e SO de modo a representar aspectos sociais e construir sistemas computacionais de acordo com necessidades humanas. É apresentada a estrutura básica do SPaM, suas principais fases e os passos que formam cada uma delas. Neste capítulo também é descrita a Conferência Semiótica, uma técnica de DP proposta nesta tese para trabalhar, com os usuários, conceitos do contexto organizacional utilizando os modelos da SO como ferramenta de apoio.

O capítulo 6 propõe abordagens para implementar sistemas computacionais informados pelos modelos da SO. Estas abordagens são descritas em três camadas: dados,

negócio (ou aplicação) e interface. Para a camada de dados é proposto o uso de uma simplificação do procedimento apresentado em (Liu, 2000). Para a camada de negócio é proposto um novo procedimento para construir diagramas UML (Unified Modelling Language) a partir dos conceitos modelados durante a análise semântica. Para a camada de interface é proposta uma arquitetura baseada na análise semântica e de normas que possibilita modificações no sistema de maneira mais simples e econômica, facilitando desta maneira a sua manutenção.

O capítulo 7 apresenta a descrição e análise do contexto do estudo de caso que foi conduzido com o objetivo de avaliar e refinar o SPaM em uma organização. Primeiramente é apresentado o paradigma de produção adotado pela organização fabril na qual conduzimos o desenvolvimento do sistema. No capítulo 7 também detalhamos o estudo de caso apresentando como o SPaM e demais procedimentos propostos foram utilizados na prática. É apresentado o sistema resultante deste estudo de caso, o Pokayoke, um sistema que dá suporte a resolução de problemas no contexto de uma organização de manufatura. É apresentada a aplicação do SPaM no desenvolvimento do sistema Pokayoke, detalhando seus ciclos de prototipação, a aplicação das técnicas de DP, como estas técnicas contribuíram para a Análise Semântica e de Normas, o uso da Conferência Semiótica durante o estudo de caso, e aspectos do projeto do sistema Pokayoke enfatizando sua arquitetura e a construção da interface. Também são discutidos aspectos sobre a avaliação e o processo de significação, durante o desenvolvimento do sistema, e são analisados dados sobre os 10 primeiros meses do uso em larga escala do sistema na organização.

Ao final, o capítulo 8 apresenta a conclusão desta tese, suas contribuições e os principais trabalhos futuros identificados.

Capítulo 2

CSCW e o Contexto Organizacional

A expansão das redes de computadores tais como a Internet e a Intranet tornou possível a interação entre várias pessoas em diferentes lugares facilitando o desenvolvimento de trabalhos em conjunto. Até então, o enfoque à utilização de sistemas computacionais era tipicamente no trabalho individual. O advento das redes despertou a necessidade de se entender a interação de grupo de pessoas tendo o computador como meio de suporte dessa interação.

Em muitos casos tarefas que poderiam ser efetuadas de forma distribuída por um grupo de pessoas são realizadas individualmente e possuem custos elevados. Utilizando sistemas de CSCW, estes custos tendem a diminuir e viabilizar a realização destas tarefas à distância e ao mesmo tempo. Uma das principais causas desta redução refere-se ao impacto que estes sistemas têm sob os principais aspectos físicos que dificultariam a realização destas tarefas, como as variáveis tempo, espaço e distância. Este capítulo apresenta a motivação para uma abordagem ao desenvolvimento de sistemas de CSCW alinhada ao tratamento de questões organizacionais.

2.1 A Importância do Contexto Organizacional para CSCW

Vários pesquisadores na comunidade de CSCW enfatizam a importância de compreender e descrever o contexto social (ou organizacional), explicitando-a em várias

definições de CSCW. A figura 2.1 exibe algumas definições de CSCW que apontam para a questão: "Como deveríamos compreender e modelar o contexto social?".

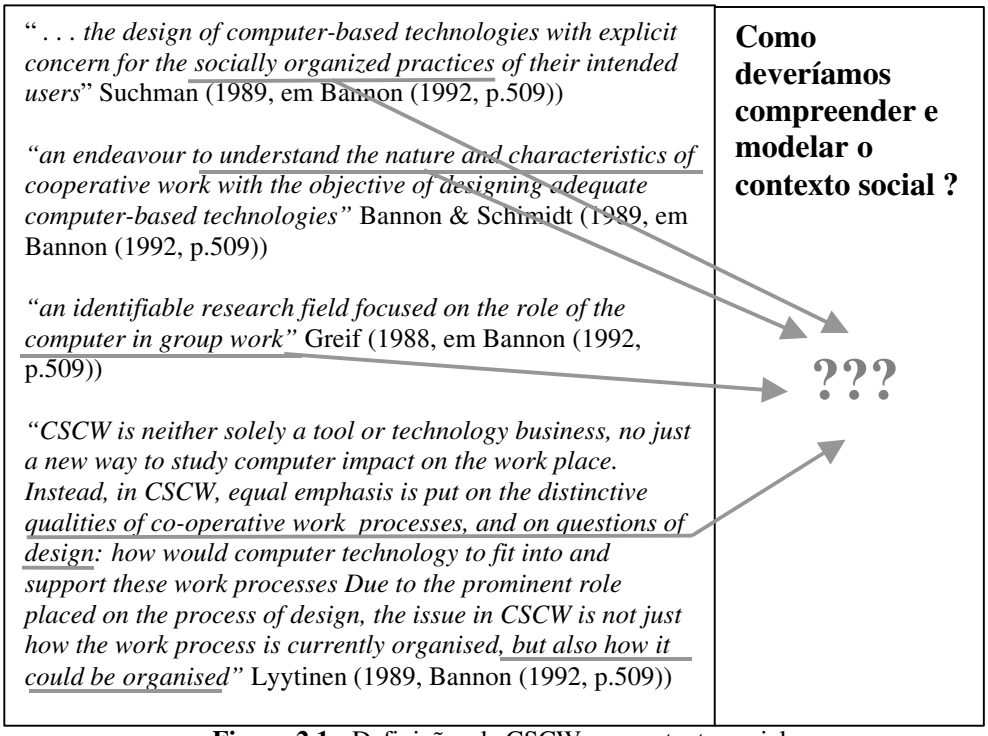


Figura 2.1 - Definições de CSCW e o contexto social

A natureza multi e interdisciplinar da pesquisa envolvendo a criação de sistemas computacionais para o uso por grupos de pessoas é inegável. Para responder a questões como as sugeridas na figura 2.1, CSCW tem sido uma área de pesquisa aberta para uma grande variedade de disciplinas incluindo psicologia cognitiva e social, sociologia do trabalho e antropologia, entre outras.

Conforme destacado por Grudin (1994), não é a organização que deve ser adaptada a um sistema de *groupware*; mas, o sistema é que deve ser adaptado às práticas organizacionais. CSCW, que pode ser visto como um tipo particular de *groupware*, deveria portanto ser integrado ao sistema de informação da organização, respeitando as normas sociais instituídas no local de trabalho.

Nesta tese, é proposto que um novo contexto organizacional seja discutido e construído junto com o design do sistema, mantendo assim o sistema alinhado com as normas organizacionais (formais e informais) que ditam as práticas de trabalho. Por motivos práticos não é proposto (nesta tese) que a organização seja inteiramente

reestruturada toda vez que um novo sistema é construído, mas somente as partes que os usuários considerem possíveis de serem aprimoradas pelo advento da nova tecnologia.

Grudin (1994) destaca oito desafios para o design de sistemas de *groupware*:

1. *A disparidade entre quem faz o trabalho e quem tem o benefício*. Uma aplicação de *groupware* não provê necessariamente o mesmo benefício para todos os membros do grupo. É esperado que se obtenha um benefício coletivo, porém, algumas pessoas devem ajustar-se ao novo sistema mais do que outras;
2. *Massa crítica e o problema do dilema do prisioneiro*. Uma aplicação de *groupware* é útil somente se uma alta percentagem de membros do grupo for utilizá-la. Se todo o grupo agir em função de seus interesses pessoais, o resultado final será pior não só para o grupo mas também para cada indivíduo;
3. *Fatores sociais, políticos e motivacionais.* Pode existir resistência, se o sistema interferir na delicada e complexa dinâmica social dos grupos de trabalho;
4. *Tratamento de exceções nos grupos de trabalho.* Um processo de trabalho pode ser descrito de duas maneiras: a maneira como as coisas deveriam ser ou a maneira como elas realmente são. Entretanto um sistema de *groupware* talvez não seja capaz de acomodar um grande número de tratamento de exceções e improvisações que caracterizam o trabalho em grupo;
5. *O design para características não freqüentemente utilizadas.* As características do sistema que não são freqüentemente utilizadas não devem obstruir e devem estar integradas com as características mais utilizadas e ao mesmo tempo estar acessíveis ao usuário;
6. *A dificuldade subestimada para avaliar groupware.* A análise, design e avaliação são muito mais árduos para aplicações multi-usuários, em função da dificuldade para capturar o dinamismo social, motivacional, econômico e político dos grupos;

7. *A quebra da tomada de decisão intuitiva.* O uso da intuição para tomar decisões de design em aplicações de *groupware* pode resultar em problemas;
8. *O gerenciamento da aceitação.* A introdução de um sistema de *groupware* no ambiente de trabalho pode ser muito mais complexa se comparada a de um sistema para uso individual. Em determinadas aplicações, se um usuário não a aceitar pode inviabilizar o uso do sistema para todo o grupo de usuários.

Várias abordagens foram desenvolvidas nas áreas de CSCW e HCI para lidar com problemas como os levantados por Grudin (1994). Na próxima seção são apresentados alguns trabalhos que destacam a importância de diferentes disciplinas e abordagens no design de sistemas de CSCW, lidando com a complexidade de entender e modelar o contexto social.

2.2 Abordagens de CSCW para entender e modelar o contexto social

Sob o enfoque da influência que fatores humanos exercem no design, podemos citar trabalhos como o de Finholt e Teasley (1998), que apresenta como as ciências sociais podem contribuir no desenvolvimento e na implantação de sistemas de CSCW. Em seu trabalho é destacada a importância não só de ciências que historicamente são referências para o desenvolvimento de sistemas cooperativos, como exemplo Etnografia e Lingüística (Honeycutt, 1998), mas também de outras disciplinas como a Psicologia. Estas ciências oferecem princípios bem fundamentados sobre o comportamento humano em grupo, como também métodos para identificar e generalizar características do ambiente, que podem ser aplicadas ao contexto organizacional por pesquisadores em CSCW.

Considerando estes aspectos multidisciplinares, Berg (1998) analisa novas abordagens para o projeto em tecnologias como CSCW, sendo estas abordagens fundamentadas em antropologia social e filosofia social. Seu trabalho discute como são

tratados os aspectos ligados ao relacionamento humano-máquina durante o trabalho cooperativo. Já Bouges e Scrivener (1998), apresentam limitações nos projetos de sistema de CSCW quando estes se destinam a usuários de diversas culturas, principalmente quando estes sistemas possuem usuários situados em diferentes partes do mundo (Schlichter e outros, 1998; Doerr, 1997). Neste contexto é destacada a necessidade de se utilizar técnicas de HCI, como por exemplo as que têm o objetivo de lidar com usuários de diferentes culturas ou técnicas para aprimorar a capacidade "adaptativa" (*tailoring*) da interface do sistema (Wulf e outros, 1999; Kahler e outros, 2000; Ronald e outros, 1995).

Analisando o contexto social, Trepess e Stockman (1999) discutem e propõem soluções para o tratamento de erros humanos em sistemas de CSCW, uma vez que parâmetros sociais influenciam na análise da causa, detecção, conseqüência e recuperação de erros humanos. Já Heaton (1998) apresenta fortes argumentos de como os aspectos culturais podem influenciar no desenvolvimento e na utilização de sistemas de CSCW. Em seu trabalho foram realizadas experiências entre grupos, um da Dinamarca e outro do Japão, obtendo resultados diferentes para cada um destes grupos. Além destes podemos destacar outros aspectos importantes para o design de sistemas de CSCW, associados ao relacionamento humano, tais como negociação e resolução de conflitos (Adelson , 1999; Monplaisir, 1999). Já Hayes e Walsham (2000) apresentaram uma discussão sobre quais influências um sistema de CSCW exerce em tarefas de uma organização.

Büscher e outros (2001), propuseram uma abordagem para o design de sistemas cooperativos chamada de "bricolage". Esta abordagem propõe envolver os usuários, especialistas em DP e etnógrafos em um ciclo contínuo de design e revisão das práticas de trabalho. Em seu trabalho é proposto o desenvolvimento em paralelo de sete aspectos interconectados durante o design do sistema: (1) a análise etnográfica e conceitual da organização social do trabalho, (2) a participação dos especialistas como co-designers, (3) o suporte técnico total, (4) o suporte técnico disponível/factível, (5) o design imediato (disponível em um espaço curto de tempo), (6) o design contínuo, e (7) lidar com o relacionamento *incerto* entre a tecnologia e as práticas de trabalho.

A abordagem adotada nesta tese visa explorar duas áreas de pesquisa (DP e SO) conjuntamente em um método especialmente criado para integrá-las, trazendo benefícios para o design de sistemas de CSCW. Estas áreas, em conjunto, têm o potencial de lidar

com vários problemas ligados ao design e ao uso de sistemas cooperativos, principalmente no que diz respeito aos aspectos relacionados à participação e significação.

2.3 A Pesquisa de CSCW para Manutenção de Sistemas

Uma das principais motivações para a pesquisa de alternativas para a manutenção/modificação de sistemas de CSCW durante seu uso é a natureza dinâmica dos grupos sociais. Nestes grupos os padrões de comportamento usualmente se modificam no decorrer do tempo e novos membros são continuamente introduzidos. Os grupos de agentes estão constantemente interagindo em contextos sociais, sendo afetados e afetando-os.

Os pesquisadores de CSCW têm explorado alternativas para manter o sistema alinhado com os requisitos dos usuários mesmo após o fim do desenvolvimento do sistema. Tais soluções incluem: o design de sistemas de CSCW flexíveis (Dourish, 1996), o design para o uso não antecipado (Robinson, 1993) e o design de sistemas de CSCW *tailorable* (Kahler e outros, 2000).

"Tailoring" pode ser compreendida como a atividade de modificar uma aplicação computacional durante o contexto de seu uso (Kahler e outros, 2000, pg. 1). O "tailoring" de sistemas de *groupware* levanta várias questões (Teege, 1999, p. 2) tais como:

- Qual processo em grupo e cultura de trabalho permite encorajar o "tailoring" pelos participantes? O "tailoring" somente é possível se for parte integrada do processo de trabalho;
- Como fazer "tailoring" de maneira colaborativa? Nos termos do trabalho colaborativo é necessário que as etapas de "tailoring" que afetem vários participantes sejam realizadas ou aceitas por todas as pessoas afetadas;
- Seriam necessários diferentes tipos de mecanismos de "tailoring" em diferentes áreas de groupware?
- Como podemos transferir o desenvolvimento do domínio de IHC para a implementação do suporte a "tailoring"?

- Quais são os atributos comuns e quais são as diferenças entre o uso de interfaces em geral e o uso de interfaces para 'tailoring" em *groupware*?
- Como podemos incluir os direitos e restrições no suporte a "tailoring"? Quando o "tailoring" pode afetar um grupo de trabalho inteiro, é necessário definir restrições para as ações de "tailoring"? Quem define e configura estas restrições?
- Que arquiteturas são necessárias para permitir "tailoring" durante o uso do sistema? Como os mecanismos de "tailoring" podem ser integrados ao sistema?
- Como podemos levar em consideração os requisitos diversos e dinâmicos durante o estágio inicial do processo de desenvolvimento? Podemos determinar os pontos e o grau de habilidade de "tailoring" nas aplicações futuras?

Pesquisadores usualmente pressupõem que o usuário deveria ser responsável pela tarefa de "tailoring" (Teege, 1999; Morch and Mehandjiev, 2000). Morch (1995) identificou três níveis de "tailoring": (1) *personalização,* no qual o usuário final pode definir os parâmetros para várias opções de configuração; (2) *integração,* no qual usuários finais podem adicionar novas funcionalidades para uma aplicação através da integração de componentes de software predefinidos; e, (3) *extensão,* no qual os usuários finais aprimoram aspectos de implementação adicionando novo código.

Embora existam alguns casos bem sucedidos de aplicações para o "tailoring", a modificação do sistema pelos usuários finais com o propósito de adaptá-lo a mudanças sociais complexas, como as promovidas por alterações das normas organizacionais, ainda está muito longe de ser um problema resolvido. Desta forma, para resultados imediatos e práticos como proposto em técnicas de Engenharia de Software, nesta tese é proposta uma arquitetura para promover modificações de maneira mais simples e econômica por um especialista em interface em conjunto com os usuários finais.

De acordo com a maioria dos pesquisadores de CSCW, nesta tese é argumentado que os usuários finais estão mais próximos do domínio e suas experiências deveriam ser consideradas na definição das modificações do sistema. Além disso, o conhecimento a

respeito da tecnologia e a experiência em modelagem de um especialista em interface é importante para modificações complexas do sistema. Nesta abordagem, é suposto que o especialista em interface trabalhe em conjunto com os usuários para realizar alterações no sistema de CSCW. Utilizando Design Participativo é possível incluir o usuário no processo respeitando seu poder sobre as decisões.

2.4 Síntese e Considerações Finais do Capítulo

A expansão das redes de computadores tornou possível a interação entre várias pessoas em diferentes lugares facilitando o desenvolvimento de trabalhos em conjunto. Entretanto a construção de sistemas que dão suporte ao trabalho cooperativo continua sendo um grande desafio em função de fatores humanos e sociais presentes no dia-a-dia do trabalho em uma organização. Várias abordagens foram desenvolvidas nas áreas de CSCW e IHC para lidar com estes fatores.

A abordagem proposta nesta tese, assim como várias outras abordagens citadas, inclui aspectos multidisciplinares. Entretanto, esta abordagem engloba em um mesmo método tanto aspectos ligados à promoção da participação do usuário no design do sistema, quanto a representação de seu processo de significação assistido por modelos semióticos do contexto organizacional, tópico ainda não explorado por outros métodos desenvolvidos para o design de sistemas de CSCW.

Uma das principais motivações para a pesquisa de alternativas para a manutenção/modificação de sistemas de CSCW durante seu uso é a natureza dinâmica dos grupos sociais. Nestes grupos os padrões de comportamento usualmente se modificam no decorrer do tempo e novos membros são continuamente introduzidos no processo. Outra característica ligada à abordagem descrita no decorrer desta tese é a possibilidade de manter os sistemas alinhados com o contexto de trabalho mesmo após o seu desenvolvimento, através de alterações na interface por modificações na especificação das normas sociais por usuários e especialistas em conjunto.

O próximo capítulo apresenta o referencial teórico da abordagem proposta, fundamentado no Design Participativo e na Semiótica Organizacional.

Capítulo 3

Referencial Teórico

Conforme mencionado anteriormente, a abordagem proposta lida com dois aspectos importantes que têm impacto direto no design de sistemas para o contexto organizacional: (1) *a participação do trabalhador no design do sistema e no desenvolvimento de um novo modelo organizacional* e (2) *a representação do sistema e da organização através de um método de design.* Para tanto, propomos uma abordagem que integra conceitos e técnicas de duas áreas de pesquisa principais: o Design Participativo e a Semiótica Organizacional. Neste capítulo, é apresentado o referencial teórico utilizado que serve de base para a formulação da abordagem proposta.

3.1 Design Participativo

O Design Participativo (DP) foi desenvolvido inicialmente na Escandinávia e provê uma série de técnicas para conduzir o design *com* os usuários e não *para* o usuário. A abordagem proposta no DP enfatiza a importância da democracia no ambiente de trabalho para aprimorar os métodos de trabalho, a eficiência no processo de design (através da experiência e comentários dos usuários), aprimoramento da qualidade do sistema, e a condução das atividades formativas. Em DP estes objetivos são atingidos através da interação direta dos usuários com designers durante todo o ciclo de desenvolvimento, e pelo controle do usuário sobre as decisões de design.

As atividades conduzidas durante o DP visam o desenvolvimento de uma visão compartilhada da tecnologia e da organização, explorando novas estruturas

organizacionais, requisitos dos sistemas e protótipos de novos sistemas. Conseqüentemente a tecnologia é compreendida como uma ferramenta para aprimorar a organização e os trabalhadores são compreendidos como as pessoas que realmente conhecem o contexto e os objetivos da organização.

A colaboração dos usuários durante todo o processo de design provê as informações necessárias para os designers. Além disso as atividades de DP aprimoram a qualidade do sistema resultante por compreender melhor o trabalho desenvolvido pelo usuário e combinar diferentes conhecimentos dos participantes durante o processo de design (Braa, 1996).

Em adição aos aspectos políticos e teóricos relativos à participação, pesquisadores da área de DP desenvolveram técnicas para promover a cooperação produtiva entre trabalhadores e designers, e.g. *future workshops, wall charting, started conference* (ver seção 3.1.2), entre outras. Müller e outros (1997) apresentam um sumário de 61 técnicas de DP e propõem um espaço taxonômico de atividades participativas a serem conduzidas durante o ciclo de design.

De acordo com Kensing e Blomberg (1998), algumas contribuições possíveis de DP para a pesquisa em CSCW incluem aspectos como:

- Pesquisadores de CSCW poderiam ficar mais atentos à importância de compreender que sistemas sempre existem em uma organização, em particular em contextos políticos e que estes contextos mostram o que é possível do ponto de vista organizacional e técnico;
- Os pesquisadores de CSCW poderiam apreciar os benefícios da participação ativa do trabalhador durante o processo de design e adotar técnicas de Design Participativo para promover esta participação;
- Sistemas de CSCW poderiam ser desenvolvidos e adaptados para organizações em particular utilizando técnicas de Design Participativo;
- Através de DP pesquisadores de CSCW poderiam aprender estratégias para conectar estudos do trabalho e o design de sistemas;

Em DP os designers estão imersos no local de trabalho. Compartilhando o mesmo contexto social com o usuário, eles (usuários+designers) podem desenvolver uma melhor

compreensão do contexto social, o que é essencial no design de sistemas de CSCW. Esta participação direta pode também potencialmente aprimorar a aceitação do sistema pelos usuários.

Novas técnicas de DP foram propostas para possibilitar a cooperação entre os usuários e designers em diferentes contextos. Entretanto, alguns fatores devem ser observados antes de aplicar DP. O primeiro fator a ser observado é a concordância por parte da empresa com os aspectos políticos ligados ao Design Participativo. Entre estes aspectos estão os relacionados à introdução de sistemas computacionais e a distribuição do poder no ambiente de trabalho.

Outro aspecto amplamente discutido nas pesquisas sobre Design Participativo é a negligência dos trabalhadores quanto à introdução de sistemas computacionais, visto que eles são os mais afetados pela introdução desta tecnologia. O interesse direto do trabalhador na introdução de novas tecnologias e seu impacto (positivo ou negativo) no ambiente de trabalho é requisito básico para a aplicação do Design Participativo. Como o Design Participativo pressupõe construir sistemas diretamente com os usuários no intuito de aprimorar o desenvolvimento de suas tarefas, é fundamental que estes estejam dispostos a investir o esforço necessário e ir além das suas atribuições habituais para obter estas melhorias em seu ambiente de trabalho.

Clement e Van den Besselar (1993) revisaram dez projetos que utilizaram Design Participativo e destacaram cinco requisitos básicos (três já apresentados anteriormente por Kensing (1983) e outros dois novos) para a sua aplicação:

1. Acesso à informação relevante;
2. Possibilidade de tomar uma posição independente frente aos problemas;
3. Participação na tomada de decisões;
4. Métodos de Design Participativo apropriados;
5. Espaço para técnicas alternativas e/ou trocas (rearranjos) organizacionais;

Muitas das técnicas de Design Participativo têm suas raízes na etnografia e Simonsen e Kensing (1997) apresentam quais são as condições básicas para utilizá-las em um projeto de design de software:

1. Os designers e os usuários devem ter uma posição positiva frente aos recursos necessários e estes recursos devem estar disponíveis;
2. Os usuários devem estar seguros com relação ao propósito da abordagem adotada, ou seja, por exemplo dar suporte a força de trabalho existente e não reduzir suas funções;
3. Os designers devem ter habilidades para conduzir e tratar situações como as em que existe conflito;
4. Os designers e os usuários devem conseguir identificar domínios em que existe demanda para aplicação da etnografia.

3.1.1 Princípios e Aspectos Políticos e Teóricos

O Design Participativo levanta questões sobre democracia, poder, e controle no ambiente de trabalho. De acordo com Ehn (1993) a democracia ideal é uma maravilhosa invenção humana: todo humano deveria ter direito a participar igualmente em decisões relativas a sua vida. Porém na prática esta liberdade sempre foi limitada; particularmente a democracia no trabalho é limitada pelas restrições impostas pela economia de mercado e pelo poder de capital.

Na Escandinávia, os sindicatos serviram de veículos para a democracia industrial por elevar o interesse coletivo dos trabalhadores. Alguns dos primeiros projetos de DP enfatizaram a estratégia de incluir os sindicatos no processo de design. Kensing e Blomberg (1998) destacam alguns resultados dos primeiros projetos, como o DEMOS (Ehn e Sanberg, 1979). De acordo com Kensing and Blomberg (1998) os resultados destes esforços incluem: um aumento do poder de barganha devido a sindicalistas melhor informados, acordos fortificados, e garantias de leis nacionais, pelas quais eles reivindicavam seus direitos e informações a respeito dos planos de gerência para novas tecnologias. Estes projetos também iniciaram uma comunidade internacional de pesquisadores com enfoque na interface entre a tecnologia e o local de trabalho.

Müller e outros (1997) consideraram três motivações convergentes para a abordagem de DP; a primeira é a democracia e as outras duas são:

- *Eficiência, Perícia e Qualidade.* Elas são aprimoradas pela participação direta do usuário durante o design, uma vez que nenhuma pessoa ou disciplina tem todo o conhecimento que é necessário para o design;
- *Confiança e Aceitação Interna.* A participação em atividades de DP pode aprimorar a aceitação do sistema;

Alguns dos projetos recentes de DP estão concentrados na promoção de uma participação racional dos trabalhadores no processo de design. Pesquisadores de DP estão constantemente propondo novas alternativas para promover a participação de trabalhadores em países com diferentes contextos sócio-econômicos e também ao novo contexto da Escandinávia e Europa no qual houve a diminuição do poder de barganha dos sindicatos. As condições necessárias para promover a participação dos trabalhadores (Clement e Besselar, 1993), a estratégia para promover esta participação, a definição de quem participa no design (freqüentemente a participação do gerente está restrita), e o grau de envolvimento do trabalhador nas tarefas de design e decisões, são exemplos de aspectos freqüentemente abordados atualmente pelos pesquisadores de DP.

Gartner e Wagner (1996) propuseram dividir a participação em três arenas:

A. *Modelando o Trabalho – Modelando o Sistema:* Nesta arena sistemas específicos são modelados e novas formas organizacionais são criadas. O que pode ser um compromisso geral para melhorar o local de trabalho e aprimorar as condições de trabalho deve ser concretizado. A tarefa de reengenharia do trabalho e sistemas técnicos conduz a agenda nesta arena, a qual abrange desde práticas, procedimentos, comunicação e cooperação, até aspectos de dependência e automação;

B. *Modelando Estruturas Organizacionais para Ação.* Freqüentemente um padrão de conflito não resolvido e repetitivo na arena A ocasiona uma ação na arena B; ou serve como uma oportunidade para tomar tal ação. A Arena B é o local no qual as "interrupções" ou violações de acordos são diagnosticadas e padrões até então estáveis da organização são questionados e redesenhados. A agenda então é redefinir condições gerais para o desenvolvimento de sistemas na organização; em termos de informações e

direitos de consulta, em função do processo, aspectos que precisam ser abordados, regras e direitos de diferentes agentes no processo, etc;

C. *Modelando as relações do contexto industrial.* Esta é a arena na qual as estruturas de leis e políticas que definem as relações entre vários padrões industriais e o conjunto de normas (de todos os aspectos relacionados ao trabalho) é negociado.

Neste trabalho aplicamos nossa abordagem para promover o DP em uma unidade de uma multinacional de capital Norte Americano no Brasil, que adota o paradigma de produção enxuta (ver seção 7.1.1). Este contexto reflete como aplicar o DP em um local com uma realidade sócio-econômica diferenciada em relação aos países Escandinavos e também no mundo globalizado. Aspectos citados nos últimos parágrafos são analisados durante a descrição do estudo de caso.

3.1.2 Técnicas de Design Participativo

Como citado anteriormente, pesquisadores na área de DP também desenvolveram técnicas que exploram diferentes abordagens para promover uma cooperação produtiva entre trabalhadores e designers. Estas técnicas têm o objetivo de prover para os designers e trabalhadores uma maneira de conectar as práticas de trabalho atuais e futuras com as possíveis novas tecnologias (Kensing e Blomberg, 1998).

Uma técnica de DP não é uma seqüência rígida de passos bem compreendidos que produz um resultado garantido, mas um suporte ou infraestrutura para um complexo processo em grupo. Abaixo são apresentados resumos das técnicas de DP que foram utilizadas no decorrer desta tese (os resumos foram retirados de Müller e outros, 1997):

- *Search Conference ou Starting Conference.* Participantes de várias organizações relacionadas, em múltiplos níveis de gerência e poder, se reúnem para analisar os relacionamentos de trabalho atuais, oportunidades futuras e como planejar o futuro a partir da situação atual. Participantes de diferentes níveis de poder são parcialmente protegidos de riscos para explorar suas idéias ou perspectivas de suas organizações (Palshaugen, 1986);

- *Ethnographic Practices.* Práticas etnográficas tiveram grande influência em DP. Etnografia requer um treinamento extensivo tanto em práticas específicas, quanto nas perspectivas e disciplinas que delineiam estas práticas (Rose e outros, 1995). Através delas podemos observar e analisar como as pessoas trabalham em seu dia-a-dia, trazendo para o desenvolvimento de sistemas aspectos relevantes sobre o contexto organizacional, que podem influenciar na concepção e posteriormente no uso do sistema na organização. Algumas destas práticas incluem a imersão do designer no contexto organizacional e o registro sistemático (em papel e via gravações) de como o usuário realiza suas tarefas em seu contexto de trabalho;
- *HOOTD(Hierarchical Object-Oriented Task Decomposition).* Participantes decompõem uma descrição de uma tarefa em objetos e ações tomadas sobre eles, e agrupam estes objetos em janelas de interface;
- *Icon Design Game.* Um participante (o desenhista) desenha esboços de ícones enquanto outros tentam descobrir que conceito o desenhista está tentando expressar. Os desenhos são os primeiros rascunhos para o desenvolvimento dos ícones na interface do sistema. Ele pode ser jogado de maneira cooperativa (com um único time) ou competitiva (com múltiplos times) (Müller e outros, 1994);
- *Artifact Walkthrough.* Usuários utilizam artefatos dos seus ambientes para reconstruir e revisar um exemplo específico de seus processos de trabalho (Wixon e Raven, 1994);
- *Prototyping.* A prototipação é utilizada de várias maneiras nas atividades participativas. Exemplo de métodos são: *Storyboard Prototyping* e *CISP(Cooperative Interative Storyboard Prototyping).* No *Storyboard Prototyping* usuários e designers avaliam e usam um protótipo que existe somente em papel (uma série de imagens). Este tipo de protótipo é freqüentemente mais rápido e barato para construir do que aqueles criados com as linguagens tradicionais de programação e, conseqüentemente, as iterações de design e a avaliação são mais rápidas. Algumas versões deste

método com o CISP incluem os usuários no desenvolvimento do protótipo e não somente na avaliação de uma versão dele (Andriole, 1992);

- *Participatory Heuristic Evaluation (PHE).* Inspetores usam um conjunto de heurísticas (algumas orientadas a produto, e outras orientadas a processo) para identificar problemas potenciais no design, no protótipo ou no sistema, em termos de usabilidade e aprimoramentos para o trabalho dos usuários finais (Müller e outros, 1998; Müller e outros, 1995);

Estas técnicas foram escolhidas com base na sua adequação a diferentes fases do ciclo de vida do desenvolvimento do sistema, a facilidade de aplicá-las no contexto de trabalho e o seu potencial de estimular a reflexão e colaboração. De acordo com a natureza flexível das técnicas de DP algumas delas foram devidamente adaptadas para facilitar seu uso no contexto do estudo de caso e sua articulação com princípios da SO; entretanto as características principais foram preservadas. Em adição a estas técnicas também foi proposta a Conferência Semiótica (Bonacin e Baranauskas, 2003a), uma nova técnica descrita no capítulo 5.

3.2 Semiótica Organizacional

De forma complementar às técnicas de Design Participativo para modelar o contexto organizacional é necessária uma metodologia formal para modelagem de sistemas, que envolva a compreensão do contexto social da organização. Para tanto é proposta a utilização da Semiótica Organizacional, que pode ser definida como o estudo de organizações utilizando conceitos e métodos da Semiótica (Osw,1995).

Semiótica como "ciência dos signos" ou teoria sobre produção de sentido e interpretação, tem sido objeto de estudo de pesquisadores em Lingüística, Estudos de Mídia, Ciências Educacionais, Antropologia, Filosofia da Linguagem entre outros. A Semiótica como um todo ou em parte tem influenciado muitos destes estudos. Computação é outra área em que a Semiótica tem mostrado ser de grande relevância (Andersen, 1991), especialmente nas áreas de interfaces humano-computador (Oliveira e Baranauskas, 1999; Souza, 1993), semiótica computacional (Gudwin e Gomide, 1997),

entre outros trabalhos como por exemplo Benyon (1994), Cunningham (1992), Fiske (1990), Gonzalez (1997), Klein (1996), Liu e Dix (1997) e Liu e Gao (1997).

Semiótica Organizacional é um dos ramos da Semiótica que entende uma organização como um sistema de signos e objetiva estudá-las utilizando conceitos e técnicas baseados na Semiótica desenvolvida por Peirce (Peirce 1931-1958) e Morris (Morris, 1938) entre outros. SO entende que todo comportamento organizado é afetado pela comunicação e interpretação dos signos pelas pessoas, de maneira individual ou em grupos.

A noção de signo é essencial em Semiótica. Na teoria Peirceana (Peirce 1931-1958, cf 2.228) "Um signo, ou *representâmen*, é aquilo que, sob certo aspecto ou modo, representa algo para alguém. Dirige-se a alguém, isto é, cria, na mente dessa pessoa, um signo equivalente, ou talvez um signo mais desenvolvido. Ao signo assim criado denomino *interpretante* do primeiro signo. O signo representa alguma coisa, seu *objeto*. Representa esse objeto não em todos os seus aspectos, mas com referência a um tipo de idéia que eu, por vezes denominei *fundamento* do *representâmen. . .*"

Qualquer marca, movimento físico, símbolo, sinal, etc. usado para indicar e "transportar" pensamentos, informações e comandos constituem signos (Sebeok, 1994, p. xi). Uma foto é um signo na medida em que ela "está para" os elementos nela representados, para alguém que a interpreta. Se, na interpretação de alguém, a palavra "amarelo" está para a cor amarela, a pronúncia da palavra "cavalo" está para o animal cavalo, fumaça está para fogo, o desenho de uma impressora na tela de um computador está para imprimir, então a palavra "amarelo", a pronúncia de "cavalo", a fumaça e o desenho da impressora na tela são todos exemplos de signos. A figura 3.1 apresenta um exemplo do signo como uma relação triádica.

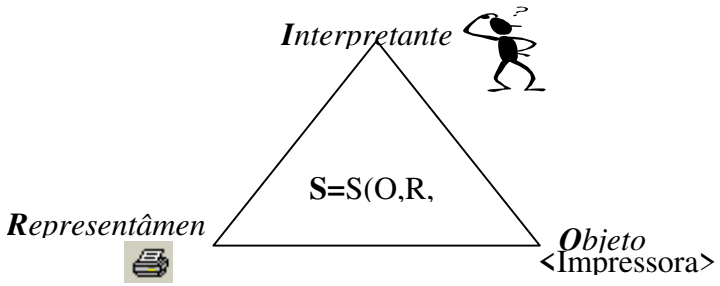


Figura 3.1 - O signo de Peirce como uma relação triádica, exemplificado (de Rocha e Bananuskas, 2000, p. 148)

Semiose é outro conceito central da Semiótica, que a escola Peirceana define como "a ação do signo". É o processo pelo qual o signo tem um efeito cognitivo sobre seu interpretante. "Por semiose entendo uma ação, uma influência que seja ou coenvolva uma cooperação de *três* sujeitos, como por exemplo um signo, o seu objeto e o interpretante, tal influência tri-relativa não sendo jamais passível de resolução em uma ação entre duplas" (Peirce, 1931-1958, cf. 5.484).

Conforme destacado por Eco (1976, p.58) na "teoria Peirceana", para estabelecer o significado de um significante é necessário nomear o primeiro significante por meio de um outro significante, que a seu turno conta com outro significante que pode ser interpretado por outro significante, e assim sucessivamente, constituindo assim um processo de Semiose Ilimitada que, para Eco, é a única garantia de um sistema semiótico capaz de explicar-se a si próprio, em seus próprios termos.

Com o propósito de explicar a fundamentação teórica dos métodos da Semiótica Organizacional, Liu (2000) destaca 4 características que descrevem a semiose:

1. Ela é universal. É aplicada para todo tipo de propriedade de processamento de signos;
2. É um processo capaz de identificar alguma coisa presente de acordo com critérios específicos ou normas;
3. O processo de representação pode ser recursivo: um *representâmen* pode ser visto como um objeto em outro processamento de signo, assim como um interpretante ou um objeto pode ser um *representâmen*;
4. É dependente de interpretação subjetiva. A semiose está fortemente relacionada ao intérprete que pode ser um indivíduo, ou um grupo social que possui certo conhecimento e obedece a certas normas. A semiose é sempre subjetiva na interpretação, dependendo do ponto de vista do intérprete e o conhecimento e habilidade que ele possui.

O conceito de "significação" pode ser entendido como o efeito que o *interpretante* final produz sobre o intérprete; ela é o resultado interpretativo que o intérprete atinge se o signo for considerado. A figura 3.2 ilustra os conceitos básicos envolvidos em uma semiose como um processo de significação.

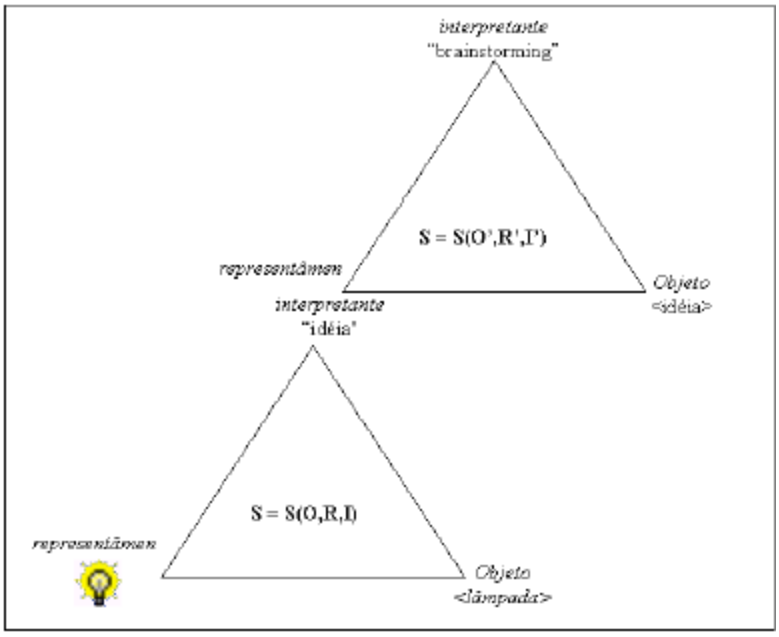


Figura 3.2 - Semiose como um processo de significação

De acordo com Eco (1976), enquanto o processo comunicativo é uma passagem de sinal (que não significa necessariamente ´um signo´) de uma fonte, através de um transmissor, ao longo de um canal, até um destinatário (ou ponto de destinação), o processo de significação não se limita a funcionar como um simples estímulo, mas desperta uma resposta interpretativa por parte do destinatário. Comunicação portanto nesta visão, só existe através de um sistema de significação que une entidades presentes a entidades ausentes, com base em regras subjacentes, algo materialmente presente à percepção do destinatário que está para qualquer outra coisa.

Podemos entender significação não só em termos denotativos (signo está para o quê) mas também em termos conotativos (associações culturais dos signos) (Barthes, 1957). A conotação envolve aspectos emocionais, de interpretação subjetiva, valores sócio-culturais e posições ideológicas (Chandler, 2001). De acordo com Fiske (1990, p. 92) a conotação é altamente arbitrária, específica para uma cultura, mas freqüentemente tem dimensão icônica.

Como a "doutrina dos signos", Semiótica engloba várias disciplinas, facilitando nosso entendimento sobre como as pessoas utilizam os signos para todos os tipos de propósitos. Semiótica Organizacional considera o trabalho interno da organização, incluindo seus sistemas de informações e suas interações com o ambiente, objetivando

encontrar novas maneiras de analisar, descrever e explicar a estrutura e o comportamento das organizações. Os estudos em SO não estão restritos a informação expressa de maneira escrita ou gráfica, mas levam em consideração os aspectos do comportamento humano durante o trabalho em uma organização. Por isso a SO é mais do que uma ferramenta para desenvolver sistemas, ela pode ser utilizada para compreender e aprimorar a organização (Liu e outros, 1997).

A figura 3.3 exibe o *framework* semiótico, proposto por Stamper (1973) para analisar o uso dos signos em seis camadas, do mundo físico ao mundo social, onde os efeitos do uso dos signos são percebidos. Estas camadas (Figura 3.3) são analisadas com técnicas específicas como a análise de informações, modelo de morfologia, análise semântica e análise de normas.

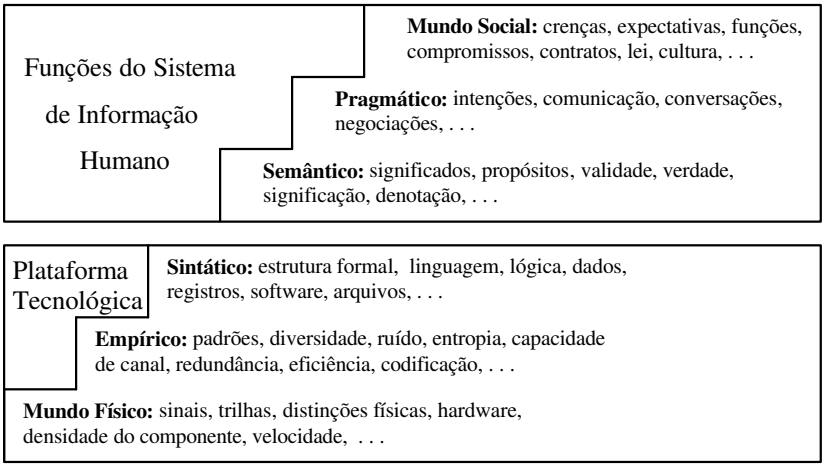


Figura 3.3 - O Framework Semiótico (de Stamper, 1973)

Uma breve descrição de cada nível é apresentada a seguir:

1. *Mundo Físico.* Neste nível os signos são estudados em sua forma física. Suas propriedades como tamanho, formato e mídia, o hardware utilizado para enviá-los, guardá-los e os processamentos de sinais são analisados;

2. *Empírico.* As propriedades estatísticas dos signos são estudadas neste nível. Os signos são vistos como seqüências de sinais não considerando seus significados. De acordo com Liu (2000) algumas questões a serem estudadas neste nível são codificações, medidas de entropia, transmissões de sinais óticos, capacidade do canal, etc;

3. *Sintático*. Neste nível estruturas complexas da linguagem são analisadas, não considerando seus significados. As regras utilizadas para compor signos complexos são descritas neste nível;
4. *Semântico.* O relacionamento entre um signo e o que ele refere são estudados neste nível. A partir do conceito de "significado comportamental" (Stamper, 1973), os significados são construídos, constantemente testados e reparados através do uso dos signos, agindo como a ligação operacional entre os signos e a prática;
5. *Pragmático.* No nível pragmático os propósitos do uso dos signos são analisados. Este é um ramo da semiótica focado no relacionamento entre os signos e o comportamento dos agentes. A comunicação e a relação com a informação pragmática são expressas neste nível;
6. *Nível Social.* As relações entre o uso dos signos e o contexto social são analisadas neste nível. Conversas devem seguir convenções sociais ao mesmo tempo em que elas podem alterar o contexto social. Atos de comunicações podem invocar, violar ou alterar normas sociais;

Para ilustrar estes níveis podemos analisar um processo de comunicação simples como uma conversa telefônica (exemplo extraído de Liu (2000)):

- no *nível físico*, os telefones devem estar conectados por uma linha;
- no *nível empírico*, sinais de voz devem ser convertidos em sinais eletrônicos (ou óticos) e transmitidos entre os dois telefones;
- no *nível sintático*, as duas pessoas que estão conversando devem seguir as mesmas regras gramaticais, isto é falar a mesma linguagem;
- no *nível semântico*, as palavras, termos técnicos ou não técnicos e os objetos citados durante a conversação devem ser compreendidos pelas duas pessoas. As sentenças e o conteúdo devem fazer sentido para ambos;
- no *nível pragmático*, existe a intenção da pessoa que está falando. Por exemplo, a frase "O preço está muito alto", poderia indicar a intenção de obter um desconto;

- no *nível social*, compromissos e obrigações sociais são freqüentemente criados ou descartados como resultado da conversação. Por exemplo, na frase: “Eu dou 10% de desconto se você comprar mais um computador” existe uma obrigação social de dar um desconto caso o comprador adquira mais computadores;

Na abordagem da SO, uma organização e o seu sistema de informação são compreendidos como um sistema social no qual os comportamentos das pessoas estão organizados por um sistema de normas. A figura 3.4 exibe esta visão através de um paradigma chamado “the organizational onion” (Stamper, 1992). Por este paradigma a organização pode ser estruturada em três níveis: o nível mais interno é o sistema de informação técnico; o segundo nível, que contém o sistema de informação técnico, é o sistema de informação formal, onde as normas formais são definidas; e o nível mais externo é o sistema de informação informal onde os significados são negociados e estabelecidos.

De acordo com Stamper (1996) mesmo que um sistema computacional fosse capaz de realizar funções técnicas com perfeição não garantiria que ele tivesse algum valor para o negócio. Desta maneira, sistemas computacionais somente têm valor para o negócio quando eles englobam soluções para problemas pertencentes ao domínio social. Por isso a modelagem semiótica deve explorar elementos dos níveis informal, formal e técnico para construir sistemas computacionais que fazem parte do nível técnico da organização mas possuem relações diretas com os níveis informal e formal.

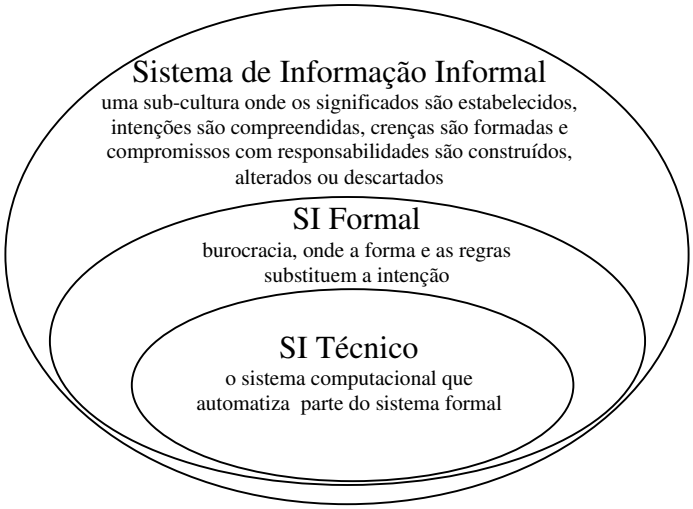


Figure 3.4 - A "organisational onion" (de Liu, 2000)

3.2.1 Técnicas da Semiótica Organizacional

Do ponto de vista filosófico, conforme mencionada anteriormente, a posição que delineia a SO é o *subjetivismo radical* (Liu, 2000) e as técnicas da SO estão fundamentadas em dois axiomas básicos: não existe conhecimento sem um conhecedor, e não existe compreensão sem ação. Este paradigma vê a realidade como uma construção social baseada no comportamento de agentes. Nesta visão, pessoas compartilham padrões de comportamento que constituem um sistema de normas. Como as pessoas estão constantemente se comunicando e discutindo, o mundo está sempre em mudança. Os métodos da SO buscam capturar estes padrões de comportamento através da análise dos signos utilizados no ambiente social. Nesta tese investigamos o uso com DP de uma das abordagens, entre várias outras propostas pelos pesquisadores da SO, que está baseada no MEASUR (Method for Eliciting, Analysing and Specifying User Requirements).

O MEASUR surgiu de um projeto de pesquisa iniciado por Stamper (1973) com o objetivo de investigar e construir um conjunto de técnicas que possam ser utilizadas por pesquisadores e usuários para compreender, desenvolver, gerenciar e utilizar sistemas de informações. O conceito que delineia o MEASUR é que organizações, por elas mesmas, são sistemas de informações e uma norma social é a unidade apropriada para a especificação. (Stamper, 1993).

De acordo com Stamper (1993), nós podemos incluir as normas necessárias para a resolução dos problemas em fases incrementais de detalhamento, como segue:

1. *Articulação do Problema.* Identifica um tópico significante e articula os problemas em um nível de detalhamento suficiente. Esta fase incorpora as seguintes técnicas:
    - *Análise semiótica* que examina a organização como um sistema de signos;
    - *Análise colateral* que lida com a inovação e a sua infraestrutura;
    - *Morfologia funcional* que olha para as normas de comunicação e controle, e como elas estão distribuídas pelas interfaces informais/formais e homem/máquina;
    - *Concepção da Avaliação* que identifica as diferentes partes interessadas (*Stakeholders*) e analisa suas avaliações sobre a situação;
    - *Gerenciamento de Disputa* ajuda a resolver conflitos entre as diferentes visões do mundo reveladas por outras análises;
2. *Análise Semântica* visa clarificar os significados da linguagem utilizada para definir o problema e sua solução em termos operacionais onde todos os estados significativos dos acontecimentos são tornados explícitos;
3. *Especificação de Normas* que aponta as autoridades e responsabilidades, e modela as normas que governam todas as atividades relevantes;
4. *Definição de Comunicação e Controle*, são aplicadas as técnicas de análise das fases 1, 2 e 3 para problemas de comunicação e controle;
5. *Definição de Projeto e Gerenciamento* para examinar escalas, custos, benefícios potenciais, etc; para implementação de uma solução.

Nesta tese, as técnicas utilizadas durante as fases do MEASUR denominadas, análise semântica e especificação de normas, são exploradas em conjunto com técnicas de DP em um método especialmente criado para integrá-las. A Análise Semântica (AS), como a técnica aplicada na fase de mesmo nome, ajuda a levantar e representar os requisitos do sistema de uma maneira formal e precisa. A Análise de Normas (AN) como

técnica aplicada durante a fase de especificação de normas, ajuda a analisar como as normas sociais, culturais e organizacionais governam a ação dos agentes no domínio de negócio.

3.2.2 Análise Semântica

Na Análise Semântica (AS) o analista no papel de facilitador, especifica as funções requeridas em um diagrama de ontologia, que representa uma visão dos agentes responsáveis no domínio de negócio em foco e seus comportamentos e ações chamadas de *affordances*. O significado das palavras utilizadas no modelo semântico para representar o negócio é visto como o relacionamento entre os signos e as ações apropriadas. A Análise Semântica foca os agentes e seus padrões de comportamento para descrever a organização.

Os conceitos da AS adotados neste trabalho são baseados em Liu (2000). A seguir é apresentada uma descrição resumida de cada um destes conceitos:

- *O Mundo* é socialmente construído através das ações dos agentes, tendo como base o que é oferecido pelo mundo físico para ele próprio;
- *Affordance* é um conceito introduzido por Gibson (1968) que pode ser utilizado para expressar invariantes de repertórios de comportamento de um organismo possíveis pela estrutura do organismo combinada com a de seu ambiente. Na análise semântica, *affordances* (Liu, 2000) são construções sociais válidas em um certo contexto social. Neste ponto temos que distinguir os diferentes conceitos de *affordance*: o original de Gibson (1973), o conceito adotado na comunidade de IHC e o conceito utilizado no MEASUR. O conceito de *Affordance* foi introduzido originalmente por Gibson (1968) e pode ser utilizado para expressar invariantes de repertórios de comportamento de um organismo possíveis pela estrutura do organismo combinada com a de seu ambiente. "*Eu quero dizer, ela (palavra affordance) é alguma coisa que se refere para ambos o ambiente e o animal de uma maneira que nenhuma palavra existente o faz. Ela implica em*

complementaridade do animal e o ambiente." (1968, p. 127). *Affordances* não são somente propriedades físicas, eles são relativos à postura e ao comportamento do animal. Para Norman (1999, 2002) no mundo do design, o termo affordance como adotado está a uma vida de distância de seu significado original. Segundo Norman (1999, p.38) ele próprio deveria ter utilizado o termo "perceived *affordance*" (ou *affordance* percebida) porque no design de interfaces nós estamos muito mais interessados sobre o quê o usuário percebe de determinado objeto. Em interfaces gráficas os designers têm algum controle sobre os "perceived *affordances*", eles devem garantir, por exemplo, que o usuário perceba o significado associado ao ato de pressionar em uma determina região da tela (Norman, 1990). Na Semiótica Organizacional, mais particularmente no MEASUR a noção que Gibson introduziu foi estendida por Stamper para incluir invariantes que são percebidas no mundo social. Estas invariantes, no mundo social, são chamadas de normas sociais por Stamper (1996, p. 374): "*Gibson deu maior enfoque à percepção do mundo físico; entretanto, a noção de affordance pode ser generalizada para incluir invariantes que nós percebemos no mundo social. Se alguém tem uma patente de direitos autorais então deveríamos ser hábeis em supor um comportamento invariante das pessoas em direção a respeitar o seu trabalho. Um "copo" tem vários invariantes sociais que são válidos, por exemplo, para permitirmos beber algo de uma maneira aceitável em uma companhia refinada e ele pode ter também a invariante de posse, assim como a patente pode . . .*". Desta maneira, na Análise Semântica, *affordances* (Liu, 2000) são construções sociais válidas em um certo contexto social. Neste sentido, Stamper (2000) argumenta que a realidade que conhecemos não foi construída de maneira individual, mas criada pelo desenvolvimento das culturas durante séculos ou milênios. Um copo é um artefato humano e sua utilização é possível não somente por causa de seus aspectos físicos, mas também porque ele tem affordances sociais (uma criança aprende que o copo não é para jogar em outras pessoas, mas para tomar água). Mesmo o mundo físico conhecido através das

ciências naturais neste sentido é predominantemente uma construção social e a ciência é a atividade social de construí-lo.

- *Agente* é um tipo especial de *affordance*, que pode ser definido como alguma coisa que tem um comportamento responsável. Um agente pode ser um indivíduo, um grupo cultural, uma comunidade, sociedade, etc. (uma pessoa, um departamento, uma organização, etc. podem ser agentes);
- *Determinante* é uma invariante de quantidade e qualidade de agentes e *affordances*, pelos quais podemos diferenciar uma instância de outra;
- *Ontologia*: Agentes devem criar significados comuns para os signos; por exemplo para a comunicação humana é necessário um vocabulário comum, ou seja, ter uma compreensão comum dos termos utilizados, este vocabulário sígnico é chamado de ontologia.
- *Dependência Ontológica* é formada quando uma *affordance* só e possível se outros *affordances* existirem. Se o *affordance* "A" é ontologicamente dependente do *affordance* "B" significa que "A" somente será possível quando "B" também for possível. Por exemplo: para que uma pessoa tropece, primeiro ela deve andar; para duas pessoas se divorciarem elas devem estar casadas; assim existe dependência ontológica entre tropeçar e andar, e divórcio e casamento;
- *Especialização:* agentes e *affordances* podem ser colocados em estruturas genérico-específicas de acordo com as propriedades que eles compartilham ou não;
- *Parte-todo,* um agente ou *affordance* pode ser parte de outro agente ou *affordance.* A parte possui todas as dependências ontológicas do todo;
- *Papel (Role-Name).* Um agente pode desempenhar um papel específico em função de um *affordance* que ele possui;
- *Responsabilidade. Q*ualquer reconhecimento de *affordances* e seus períodos de existência são vinculados aos agentes. Ações e trocas de estados são

sempre decididas por um agente responsável de acordo com suas autoridades;

Os conceitos da AS podem ser representados por meio de diagramas de ontologia. A figura 3.5 exibe esta representação gráfica, onde agentes são representados por círculos, *affordances* por retângulos, dependências ontológicas por linhas da esquerda para a direita, papel (*role-name*) por parênteses e parte-todo por linhas com pontos pretos. Esta notação é baseada na NORMA, que é uma linguagem de representação de conhecimento desenvolvida como parte das técnicas do MEASUR (Stamper, 1992; Stamper, 1993; Stamper, R. K. e outros, 1988).

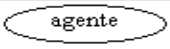
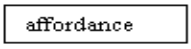
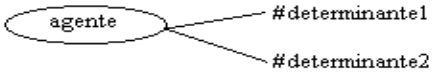
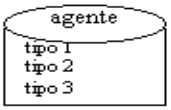
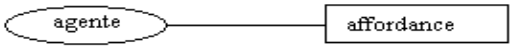
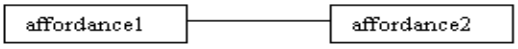
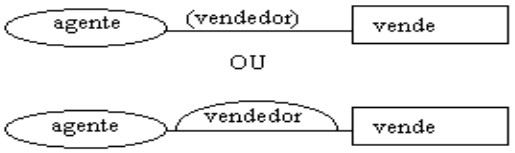
REPRESENTAÇÃO	CONCEITOS
	Agente
	Affordance
	Determinantes
	Especialização (pode ser utilizada em agentes, affordances ou papéis)
	Dependência ontológica entre um affordance e um agente (o affordance só pode existir enquanto o agante existir)
	Dependência ontológica ente dois affordances (o affordance2 só por existir enquanto o affordance1 existir)
	Relação parte-todo, o agente2 é parte do agente1 (pode existir parte todo entre agentes, affordances e papéis)
	Papel (o agente que vende é chamado de vendedor)

Figure 3.5 - Representação Gráfica de alguns conceitos da Análise Semântica

A análise semântica pode ser realizada em quatro fases principais conforme exibe a figura 3.6:

1. *Definição do problema.* Nesta fase é feita uma descrição do problema e ele é estudado. De maneira complementar à definição oficial do problema, podem

ser realizadas investigações através do estudo de documentos relevantes e entrevistas com usuários potenciais;

2. *Geração de affordances candidatos.* Os designers analisam o material da primeira fase para identificar *affordances*, possíveis agentes e outros tipos de relacionamentos. Uma lista com todas as palavras consideradas relevantes é construída durante esta fase, algumas palavras são deixadas de fora, mas isto não significa que elas não possam ser incluídas no modelo posteriormente;
3. *Agrupamento de Candidatos.* Primeiramente, os designers agrupam os candidatos da lista construída durante a última fase em categorias de acordo com os conceitos básicos da análise semântica (como agentes, *affordances*, determinantes, etc). O próximo passo é construir uma tabela com duas colunas: (1) as palavras candidatas a *affordances*, e (2) as classificações e alguma informação adicional explicando suas funções. Alguns candidatos novos podem ser incluídos na lista da fase 2. Em seguida pequenos diagramas ontológicos são construídos como fragmentos de um diagrama maior; estes fragmentos têm o propósito de exibir os efeitos dos agrupamentos nas unidades semânticas;
4. *Diagramação Ontológica.* Os fragmentos então são conectados para formar um único diagrama de ontologia, de acordo com as dependências existentes entre eles. Alguns erros cometidos durante a última fase são corrigidos nesta fase. O diagrama de ontologia deve ter um agente raiz único, a "sociedade" (se a sociedade não existir nada irá existir). Também é recomendado que cada agente deva ter no máximo dois antecedentes nas relações de dependências ontológicas (Stamper, 1979). Finalmente, o diagrama de ontologia deve ser lido várias vezes por analistas e verificado pelos usuários.

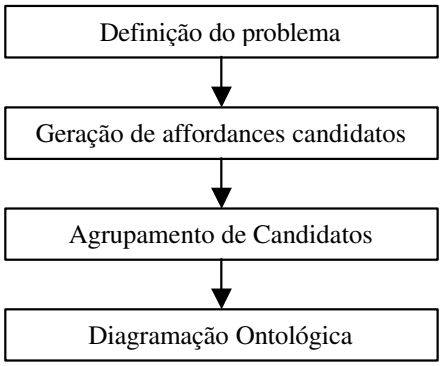


Figura 3.6 - As principais fases da Analise Semântica (de Liu, 2000)

A figura 3.7 é um exemplo de um diagrama de ontologia. Nele:

> "A *sociedade* é o agente raiz neste modelo, e tem dois *affordances*: *pessoa* e *coisa*, onde *pessoa* é um agente e *coisa* é um *affordance* normal. Ambos são ontologicamente dependentes de *sociedade*. Se a sociedade não existir, então os conceitos de *pessoa* e *coisa* ficam não definidos. Conseqüentemente, é a *sociedade* que tem os *affordances* para reconhecer *pessoa* e *coisa* e seus outros dependentes. Se a *sociedade* for vista como um agente modificado com a extensão dos *affordances pessoa* e *coisa*, é possível falar sobre "*possuir"*. De maneira mais direta, uma *pessoa* pode ter uma *coisa*. Os agentes são colocados em círculos, como *sociedade* e *pessoa*. Os *papéis* são colocados em semicírculos, como *dono, vendedor, comprador.* Ações e outros tipos de *affordances* são colocados em caixas retangulares. Note que a ação *vende* é ontologicamente dependente do *Role-name dono* e do *affordance possui*, e a ação *compra* está construída sobre *pessoa* e *possui*. Isto sugere que vender somente é possível para o dono que possui alguma coisa, enquanto comprar é para qualquer pessoa. Vender e comprar referem-se ao *affordance possui, o* que significa que quando pessoas estão negociando, estamos lidando com a "propriedade" e não com o artefato." (Liu, 2000, p. 69-70)

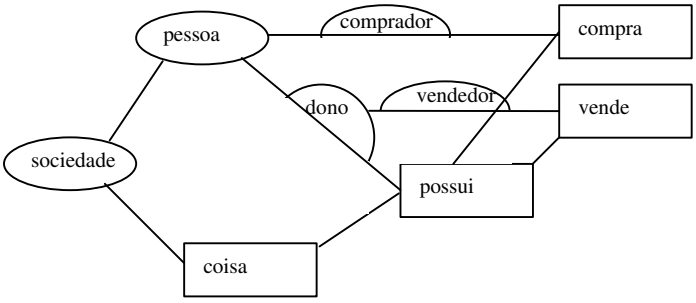


Figura 3.7 - Uma ilustração da NORMA na forma Gráfica (de Liu, 2000)

3.2.3 Análise de Normas

Em adição à análise semântica (que modela os aspectos estáticos do comportamento dos agentes) também é necessário modelar os aspectos dinâmicos. Através da Análise de Normas (AN) é possível representar o comportamento dos agentes nos níveis pragmático e social do framework Semiótico (figura 3.3).

No nível pragmático a AN descreve o relacionamento entre o uso intencional dos signos para comunicação entre agentes e o comportamento resultante dos agentes responsáveis no contexto social. No nível social as normas expressam crenças, expectativas, compromissos, contratos, leis, culturas e também o negócio. Normas correspondem no nível social à idéia de *affordance* no nível individual (Liu, 2000). As normas são resultantes de ações (*affordances*) dos agentes em uma sociedade e ao mesmo tempo controlam e ordenam as ações dos agentes em um contexto social. Elas também podem ser vistas como *affordances* coletivos de agentes complexos (por exemplo: organizações, governo, departamentos, grupos culturais, . . .) no nível social.

Normas podem ser definidas nos níveis técnico, formal e informal da organização. De acordo com Stamper e outros (2000), normas podem ser representadas por todos os tipos de signos, sejam eles documentos, comunicação oral ou comportamento, com o intuito de preservá-las, difundi-las e segui-las. Entretanto ninguém pode colocar as mãos em uma norma da forma como se pega um documento e se carrega através da

organização. Uma norma é mais parecida com um "campo de força" que faz os membros de uma comunidade terem a tendência de comportar-se ou pensar de uma certa maneira.

A análise de normas usualmente é efetuada com base nos resultados da análise semântica. O modelo semântico delimita a área de atuação da organização e identifica os padrões básicos de comportamento (*affordances*) dos agentes, enquanto normas especificam condições e restrições do comportamento (Stamper e outros, 2000)

As normas descrevem como um agente pode julgar uma situação e como ele toma decisões para ações, incluindo aspectos ligados a autoridade e delegação de responsabilidade que podem ser descritas através da lógica Deôntica (Areces et al., 2001; Goré, 1992). O seguinte formato pode ser utilizado para a especificação de normas de comportamento (Liu e Dix, 1997)

<Norma>::= **whenever** <condição> **if** <estado> **then <**agente**> is** <D> **to do** <ação>

onde <Norma> é o nome da norma. O campo <condição> especifica uma certa condição que a norma é aplicada em um determinado estado <estado>. O campo <agente> especifica qual agente tem a obrigação, permissão ou proibição, de acordo com operador deôntico <D>, de realizar uma determinada ação <ação>.

As normas nem sempre são seguidas pelos agentes, mas servem como um parâmetro para suas decisões, por exemplo: uma norma especifica que os agentes são obrigados a pagar taxas e é possível que um agente não tenha dinheiro suficiente e neste caso provavelmente ele não irá pagá-las; mas, usualmente existe um custo quando agentes não obedecem a normas que são obrigatórias.

As normas são usualmente especificadas e relacionadas aos *affordances* do diagrama de ontologia. A figura 3.8 é parte de um digrama de ontologia; este diagrama ilustra que o *affordance empregar* tem *organização* e *pessoa* como antecedentes. Isso significa que *empregar* existe somente durante a existência de *organização* e *pessoa*. Mas este diagrama de ontologia não modela as condições e restrições de comportamento sobre o *affordance empregar*; para tanto, conforme ilustra a figura 3.8, normas são associadas ao *affordance empregar*. Um exemplo de uma norma possível é: sempre que um

empregado faltar ao trabalho, se a falta não for justificada então o *empregador* tem a permissão de descontar a falta no salário.

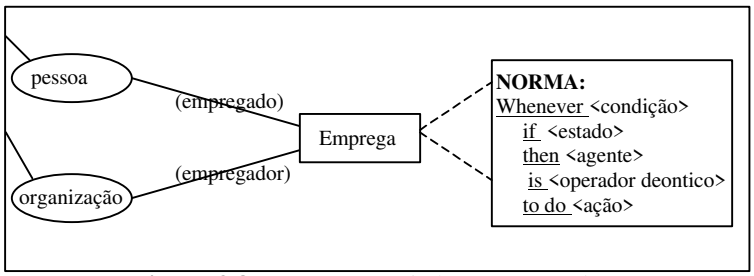


Figura 3.8 - Norma associada a um *affordance*

A análise de normas pode ser realizada em quatro passos. O primeiro passo é a "análise de responsabilidade", resultando em descrições detalhadas da responsabilidade, autoridade e as ações correspondentes. O segundo passo é a *proto-norm analysis* onde as circunstâncias que uma ação "pode, deve ou não deve" ser realizada pelos agentes são especificadas. O terceiro passo é a *trigger analysis* onde as relações entre o tempo e as ações são modeladas para especificar um cronograma para estas ações. O quarto passo é a "especificação detalhada de normas" onde as normas são inteiramente especificadas em linguagem natural e formal.

3.2.3 Análise de Normas

Em adição à análise semântica (que modela os aspectos estáticos do comportamento dos agentes) também é necessário modelar os aspectos dinâmicos. Através da Análise de Normas (AN) é possível representar o comportamento dos agentes nos níveis pragmático e social do framework Semiótico (figura 3.3).

No nível pragmático a AN descreve o relacionamento entre o uso intencional dos signos para comunicação entre agentes e o comportamento resultante dos agentes responsáveis no contexto social. No nível social as normas expressam crenças, expectativas, compromissos, contratos, leis, culturas e também o negócio. Normas correspondem no nível social à idéia de *affordance* no nível individual (Liu, 2000). As

normas são resultantes de ações (*affordances*) dos agentes em uma sociedade e ao mesmo tempo controlam e ordenam as ações dos agentes em um contexto social. Elas também podem ser vistas como *affordances* coletivos de agentes complexos (por exemplo: organizações, governo, departamentos, grupos culturais, . . .) no nível social.

Normas podem ser definidas nos níveis técnico, formal e informal da organização. De acordo com Stamper e outros (2000), normas podem ser representadas por todos os tipos de signos, sejam eles documentos, comunicação oral ou comportamento, com o intuito de preservá-las, difundi-las e segui-las. Entretanto ninguém pode colocar as mãos em uma norma da forma como se pega um documento e se carrega através da organização. Uma norma é mais parecida com um "campo de força" que faz os membros de uma comunidade terem a tendência de comportar-se ou pensar de uma certa maneira.

A análise de normas usualmente é efetuada com base nos resultados da análise semântica. O modelo semântico delimita a área de atuação da organização e identifica os padrões básicos de comportamento (*affordances*) dos agentes, enquanto normas especificam condições e restrições do comportamento (Stamper e outros, 2000)

As normas descrevem como um agente pode julgar uma situação e como ele toma decisões para ações, incluindo aspectos ligados a autoridade e delegação de responsabilidade que podem ser descritas através da lógica Deôntica (Areces et al., 2001; Goré, 1992). O seguinte formato pode ser utilizado para a especificação de normas de comportamento (Liu e Dix, 1997)

<Norma>::= **whenever** <condição> **if** <estado> **then <**agente**> is** <D> **to do** <ação>

onde <Norma> é o nome da norma. O campo <condição> especifica uma certa condição que a norma é aplicada em um determinado estado <estado>. O campo <agente> especifica qual agente tem a obrigação, permissão ou proibição, de acordo com operador deôntico <D>, de realizar uma determinada ação <ação>.

As normas nem sempre são seguidas pelos agentes, mas servem como um parâmetro para suas decisões, por exemplo: uma norma especifica que os agentes são obrigados a pagar taxas e é possível que um agente não tenha dinheiro suficiente e neste

caso provavelmente ele não irá pagá-las; mas, usualmente existe um custo quando agentes não obedecem a normas que são obrigatórias.

As normas são usualmente especificadas e relacionadas aos *affordances* do diagrama de ontologia. A figura 3.8 é parte de um digrama de ontologia; este diagrama ilustra que o *affordance empregar* tem *organização* e *pessoa* como antecedentes. Isso significa que *empregar* existe somente durante a existência de *organização* e *pessoa*. Mas este diagrama de ontologia não modela as condições e restrições de comportamento sobre o *affordance empregar*; para tanto, conforme ilustra a figura 3.8, normas são associadas ao *affordance empregar*. Um exemplo de uma norma possível é: sempre que um *empregado* faltar ao trabalho, se a falta não for justificada então o *empregador* tem a permissão de descontar a falta no salário.

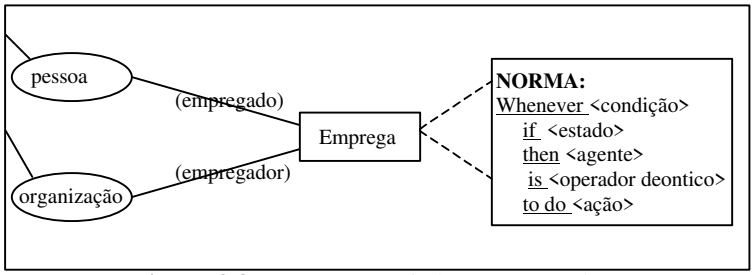


Figura 3.8 - Norma associada a um *affordance*

A análise de normas pode ser realizada em quatro passos. O primeiro passo é a "análise de responsabilidade", resultando em descrições detalhadas da responsabilidade, autoridade e as ações correspondentes. O segundo passo é a *proto-norm analysis* onde as circunstâncias que uma ação "pode, deve ou não deve" ser realizada pelos agentes são especificadas. O terceiro passo é a *trigger analysis* onde as relações entre o tempo e as ações são modeladas para especificar um cronograma para estas ações. O quarto passo é a "especificação detalhada de normas" onde as normas são inteiramente especificadas em linguagem natural e formal.

3.3 Síntese e Considerações Finais do Capítulo

Neste capítulo foram apresentadas as duas áreas que servem de referencial teórico para a solução proposta nesta tese. O Design Participativo (DP) objetiva conduzir o

design *com* o usuário e não exatamente para o usuário. A abordagem proposta no DP enfatiza a importância da democracia no ambiente de trabalho para aprimorar os métodos de trabalho, a eficiência no processo de design (através da experiência e comentários dos usuários), aprimoramento da qualidade do sistema e a condução das atividades formativas. Em DP estes objetivos podem ser atingidos através da interação direta dos usuários com designers durante todo o ciclo de desenvolvimento e pelo controle do usuário sobre as decisões de design.

O DP tem como principais motivações promover a democracia no ambiente de trabalho, a eficiência, perícia e qualidade no design de sistema, e a confiança e aceitação interna do sistema. Pesquisadores na área de DP também desenvolveram técnicas que exploram diferentes abordagens para promover uma cooperação produtiva entre trabalhadores e designers. Estas técnicas têm o objetivo de prover para os designers e trabalhadores uma maneira de conectar as práticas de trabalho atuais e futuras com as possíveis novas tecnologias. Neste capítulo foram apresentadas descrições resumidas das técnicas utilizadas no decorrer desta tese (Müller e outros, 1997): *Search Conference or Starting Conference, Ethnographic Practices, HOOTD(Hierarchical Object-Oriented Task Decomposition), Icon Design Game, Artifact Walkthrough, Prototyping,* e *Participatory Heuristic Evaluation.*

De forma complementar às técnicas de Design Participativo, para modelar o contexto organizacional é necessária uma metodologia formal para modelagem de sistemas que envolvam a compreensão do contexto social da organização. Para tanto, é proposta a utilização da Semiótica Organizacional que pode ser definida como o estudo de organizações utilizando conceitos e métodos da Semiótica (Osw,1995).

A SO é um dos ramos da Semiótica particularmente relacionada a negócio e organizações. O estudo das organizações está baseado na observação de que pessoas criam o comportamento organizado através da comunicação de signos individualmente ou em grupos (Liu, 2000). Um conjunto de técnicas propostas inicialmente por Stamper (1973) permite estudar o uso dos signos nas organizações e seus efeitos sociais. Neste capítulo também foram apresentadas as duas principais técnicas de SO utilizadas nesta tese: (1) a Análise Semântica, que define a área de atuação de uma organização pela identificação das invariantes de comportamentos dos agentes, (2) a Análise de Normas,

que descreve o nível pragmático e social da organização, como estes agentes podem julgar e agir de forma organizada. O próximo capítulo descreve como a SO pode contribuir para o DP e vice-versa e como ambos podem ser utilizados na construção de um novo contexto organizacional.

Capítulo 4

Conectando Conceitos de Design Participativo e Semiótica Organizacional

Este capítulo discute a possibilidade de relacionar conceitos do Design Participativo e da Semiótica Organizacional, como a Análise Semântica e de Normas podem contribuir para o Design Participativo e como a participação do usuário poderia contribuir para a modelagem semiótica. É apresentada uma abordagem para a construção de um novo contexto organizacional baseada na aprendizagem mútua promovida pelo uso de DP e SO. Ao final são apontados aspectos relevantes à construção de um modelo organizacional futuro.

4.1 Porque é possível relacionar conceitos de Design Participativo e da Semiótica Organizacional

O Design Participativo não está condicionado a nenhum paradigma de modelagem de sistemas ou de linguagem de programação; o seu objetivo é trazer o usuário para dentro do processo de design de sistemas. Algumas técnicas de DP, como por exemplo a *HOOTD (Hierarchical Object-Oriented Task Decomposition),* utilizam modelos específicos para auxiliar a representação dos conceitos e aprimorar a dinâmica da

interação entre designers e trabalhadores. Assim como a HOOTD faz uso destes modelos, propomos que técnicas de SO sejam utilizadas para modelar conceitos capturados pelas técnicas de DP e desenvolvemos uma nova técnica de DP que utiliza modelos da SO para a representação dos conceitos elicitados e aprimorar a dinâmica de interação do grupo.

Por outro lado, técnicas da SO, como a Análise Semântica e a Análise de Normas não restringem a participação do usuário; muito pelo contrário, na descrição dos passos da Análise Semântica por Liu (2000, pp. 75) é enfatizada a importância dos usuários e de documentos para ajudar o analista a compreender melhor as palavras e expressões presentes no problema em foco. Entretanto, a Semiótica Organizacional não provê um conjunto de técnicas para que os usuários participem da modelagem. Em nossa abordagem utilizamos o DP para trazer os usuários para dentro do processo de design do sistema e participar das decisões de design.

Nas duas próximas seções é apresentada uma visão na qual DP e SO mostram-se como técnicas complementares. Na próxima seção são apresentados alguns aspectos nos quais a SO pode contribuir para o desenvolvimento de software utilizando DP. Na seção seguinte é apresentado como DP pode contribuir para o desenvolvimento de software utilizando SO, partindo do conceito de signo.

4.2 Como a Semiótica Organizacional Pode Contribuir para o Design Participativo

Primeiramente temos que analisar os aspectos teóricos e filosóficos que servem como base para DP e SO. Um princípio básico do Design Participativo é a democracia no ambiente de trabalho; já na Análise Semântica e de Normas o subjetivismo é um dos princípios básicos. Analisando-os é possível extrair perguntas tais como: “Por quê a postura subjetivista estaria mais alinhada à democracia no ambiente de trabalho do que a postura objetivista?” “Como uma postura subjetivista poderia colaborar para a democracia no ambiente de trabalho?”

Durante o design de um sistema várias decisões devem ser tomadas; no caso específico de DP elas freqüentemente envolvem discussões por grupos de pessoas. “Qual seria a melhor decisão na interpretação objetivista: a que o grupo quer ou a que é

realmente melhor?" Na interpretação objetivista existe uma realidade independente do observador; portanto, a "melhor decisão" é independente da vontade do grupo e deve ser definida por quem (ou parte do grupo) supostamente seria mais hábil em identificar esta realidade. Por outro lado, a postura subjetivista interpreta a realidade como uma construção social do grupo. Desta maneira, a "melhor decisão" é construída pela interação entre os elementos do grupo; neste caso, se prevalecer o ambiente democrático a melhor decisão será vista como o consenso da maioria do grupo.

A seguir são apresentadas situações hipotéticas que ilustram casos freqüentes no design de sistemas utilizando DP, nos quais é difícil estabelecer-se uma verdade absoluta. Estas situações refletem decisões sobre aspectos do sistema desde o nível social até o nível físico do *framework* semiótico:

- *Nível Social*: A implantação de uma determinada tecnologia na produção de uma fábrica pode resultar na demissão de vários funcionários. Talvez, olhando somente para a gerência, seja fácil extrair uma solução única e objetiva, como por exemplo: implantar o sistema imediatamente para aumentar a competitividade da empresa. No entanto, quando levamos a discussão a um contexto mais amplo e democrático, outros fatores são levantados, que vão desde aspectos humanitários como por exemplo "Como ficam as famílias daqueles que forem demitidos?", até aspectos de interesse imediato da própria empresa: "Não seria melhor e mais barato não implantá-lo do que quebrar os acordos de estabilidade? Não poderíamos fazer uma implantação gradual e qualificar os funcionários a desenvolver outras funções, dados os custos de demissão?". Em um contexto como este é difícil extrair e modelar uma solução objetiva; seria mais razoável analisar uma solução subjetiva que é construída pela interação entre os agentes no decorrer da discussão;
- *Nível Pragmático:* Quando construímos um sistema devemos ter claros seus propósitos e ao mesmo tempo estes devem estar visíveis para ao usuário. Por exemplo, um sistema de sugestões pode pedir a identificação dos usuários para um plano de gratificações; qual seria a melhor maneira de passar as reais intenções ao usuário, de maneira que ele se sinta seguro sobre elas?

Talvez para alguns basta citar o número de um certo procedimento para que isto fique claro; entretanto para outros talvez seja necessário um aviso no sistema acompanhado de uma longa explicação. A solução para que as intenções por traz do software fiquem claras poderia ser construída durante as discussões e pode não ser possível estabelecer uma solução ideal baseada em uma verdade absoluta e independente de interpretação;

- *Nível Semântico:* Qual é o melhor elemento gráfico para representar na interface do usuário o conceito de *Brainstorming*? Alguém individualmente pode sugerir que determinado ícone uma representação para o conceito; entretanto, o que faz sentido para alguns pode não fazer sentido para outros, isto é visível no próprio conceito de signo. Em DP técnicas como o *Icon design game* são utilizadas para extrair uma decisão democrática sobre os ícones que serão representados no sistema. Neste contexto os ícones da interface são criados e analisados, e a representação "ideal" é construída pela interação dos agentes durante a aplicação da técnica;
- *Nível Sintático:* Que protocolo de rede e linguagem computacional utilizar? Isto pode parecer uma decisão somente para os desenvolvedores e pode ser simplesmente avaliada por parâmetros objetivos. Entretanto, o DP inclui mais pessoas no processo; desta maneira, parâmetros até então considerados objetivos como custos tornam-se passíveis de discussões. A escolha de uma linguagem pode influenciar no custo de desenvolvimento em função da habilidade prévia dos programadores na linguagem, necessidade de treinamento e a dificuldades de manutenção em longo prazo. Além disso, em algumas empresas, sistemas mais baratos podem significar disponibilidade de recursos para outras áreas. Mensurar os custos e benefícios de cada linguagem envolve discussões e nem sempre é possível estabelecer uma verdade absoluta sobre qual é a melhor opção. Desta maneira a solução é construída pela interação entre agentes no processo de discussão;
- *Nível Empírico:* Qual capacidade de rede devemos almejar? Quanto maior a capacidade normalmente maior é o custo; quem realmente pode opinar sobre

o tempo de resposta aceitável para uma determinada tarefa são os usuários. Nem sempre a melhor solução tecnológica é a melhor solução para a organização; em DP o custo da tecnologia empregada é discutido por usuários e designers com diferentes concepções sobre o assunto;

- *Nível Físico:* Qual hardware empregar? Em DP os designers não deveriam estar sozinhos nesta decisão; as necessidades e custos devem ser discutidos com usuários e desenvolvedores; desta maneira fica difícil estabelecer uma verdade absoluta e independente de interpretação sobre qual é a melhor opção. A solução adotada é construída pela discussão entre os agentes e a realidade sobre a melhor solução é construída sobre esta interação.

Outro aspecto a ser analisado sobre o ponto de vista da democracia no design de sistemas é que o valor do DP, em termos democráticos, fica reduzido caso apliquemos técnicas, obtenhamos relatórios e modelos, mas tivermos os designers apropriando-se do uso da palavra. Por exemplo, imagine que em um determinado contexto usuários desejam "resposta rápida do sistema"; os designers, através da apropriação do significado da palavra, podem usar o termo "resposta rápida" de várias maneiras diferentes, inclusive para pedir um hardware novo que melhore em alguns mili-segundos o tempo de resposta, o que pode ser insignificante em termos de benefícios aos usuários. Para evitar este tipo de situação entendemos que os signos são criados e compreendidos em um contexto social (no caso a organização). Portanto, durante a modelagem é fundamental utilizar conceitos que foram definidos e são compreendidos no contexto social, para manter os modelos conectados com a realidade deste contexto.

Os signos têm significados diferentes em diferentes culturas de acordo com a construção social. Por exemplo, a palavra "*football*" não tem o mesmo significado nos Estados Unidos e na Inglaterra, referindo-se a construções sociais diferentes. Da mesma maneira a palavra "casamento" tem diferentes significados em diferentes culturas. Uma alternativa é distinguir estes conceitos atribuindo a eles diferentes nomes em um sistema de classificação, mas esta abordagem tem no mínimo dois problemas:

1. É possível atribuir nomes que talvez não façam sentido no contexto social que é modelado. Por exemplo: podemos atribuir a expressão "Casamento

Brasileiro" para nos referirmos à palavra "Casamento" no contexto do Brasil, mas, como invenção do designer, pode significar qualquer coisa ferindo o princípio da democracia;

2. Algumas palavras têm tantos significados diferentes em diferentes contextos sociais que não é factível atribuir nomes diferentes para cada uma delas. Nós não podemos atribuir uma nova palavra para cada conceito que não encaixe em nosso sistema de classificação, pois o modelo ficaria incompreensível.

As palavras podem ser explicadas pelos padrões de comportamento dos grupos sociais. Na análise semântica nós não estamos tentando classificar as coisas, mas descrever o contexto social através de padrões de comportamento, que possuem normas (modeladas na análise de normas) dependentes do contexto em que estão associadas. Esta relação estreita entre o mundo social e os agentes promove uma explicação plausível de como os signos podem ser interpretados pelos agentes no contexto social.

Outro princípio básico do DP é interação e comunicação entre os participantes do design; entretanto para que isto ocorra é fundamental uma linguagem comum entre os designers e usuários. Esta tese apresenta como os modelos da Semiótica Organizacional podem aprimorar a comunicação entre os designers e usuários em um processo participativo de construção de um novo contexto organizacional. A suposição que a Semiótica Organizacional pode aprimorar a comunicação entre designers e usuários, está fundamentada no princípio de que os signos representados nos modelos são definidos em função de *affordances* e normas construídos pelas interações entre agentes no contexto que está sendo modelado. Além disso os modelos da SO são independentes da tecnologia empregada; eles podem ser utilizados para revisão de métodos de trabalho ou simplesmente como um artefato para discussão independente da tecnologia empregada para modelar o sistema.

4.3 Como o Design Participativo Pode Contribuir para a Semiótica Organizacional

A análise semântica e de normas pressupõe que os *affordances* e normas sejam elicitados do contexto social, e o DP inclui os agentes que fazem parte deste contexto no processo de modelagem. Se os designers constroem sozinhos modelos, estes representam a sua interpretação dos signos presentes do contexto social e não a interpretação dos agentes que fazem parte dele.

Nesta tese é defendida uma posição na qual não basta olhar para o contexto organizacional para construímos uma boa interpretação dele, é necessário estar imerso nele e incluir os agentes que fazem parte dele na discussão. Por melhor que sejam as técnicas de modelagem empregadas no design do sistema, na abordagem convencional (onde em geral, o usuário final tem papel restrito nas tomada de decisões) os modelos são freqüentemente construídos com base no conhecimento que os designers têm sobre o domínio. Eles, como observadores externos, não vivem o dia a dia no domínio, não exercem funções no contexto de trabalho e não possuem habilidades para desenvolver as tarefas; portanto, seu conhecimento sobre o contexto é sempre limitado por sua percepção parcial do domínio.

A figura 4.1 ilustra um caso típico do que pode ocorrer na abordagem convencional: em uma determinada organização, trabalhadores interagem em um contexto organizacional e cada um deles possui sua própria interpretação, desejos e opiniões sobre o sistema (representados por balões na figura 4.1). Por sua vez os designers também (parte direita superior da figura 4.1) possuem suas próprias interpretações, desejos e opiniões sobre o sistema. Neste modelo nada garante que o sistema esteja de acordo com as necessidades dos trabalhadores e da organização como um todo.

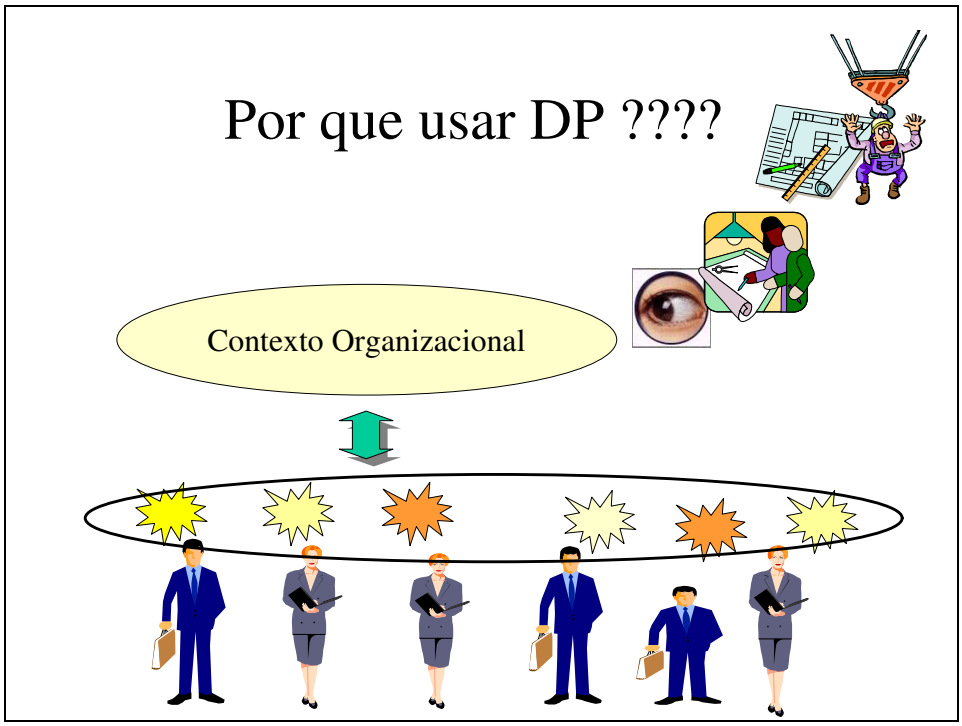


Figura 4.1 - Interpretações do Contexto Organizacional

A figura 4.2 apresenta como o DP pode contribuir. Nesta abordagem os designers estão imersos no contexto organizacional. A construção de um novo contexto organizacional é possibilitada pela discussão entre os designers e usuários; assim, os signos empregados nos modelos são resultado da interpretação de todas as pessoas do grupo. Como resultado final espera-se que os modelos e o novo contexto organizacional (que contém o sistema) esteja de acordo com a significação do grupo.

As análises semântica e de normas envolvem a articulação de vários conceitos e o desafio está em como conduzir as atividades de Design Participativo e como transformar o que foi discutido para o modelo do sistema. Conforme apresentado anteriormente, a Análise Semântica tem quatro fases: (1) Definição do problema, (2) Geração de *affordances* candidatos, (3) Agrupamento de Candidatos e (4) Diagramação Ontológica. Estas fases não prevêem como os usuários estariam participando diretamente na modelagem; entretanto técnicas de DP podem contribuir de diferentes maneiras para cada uma delas. A análise de normas também não prevê a participação direta do usuário; assim como na análise semântica, estas técnicas também podem contribuir de diferentes maneiras.

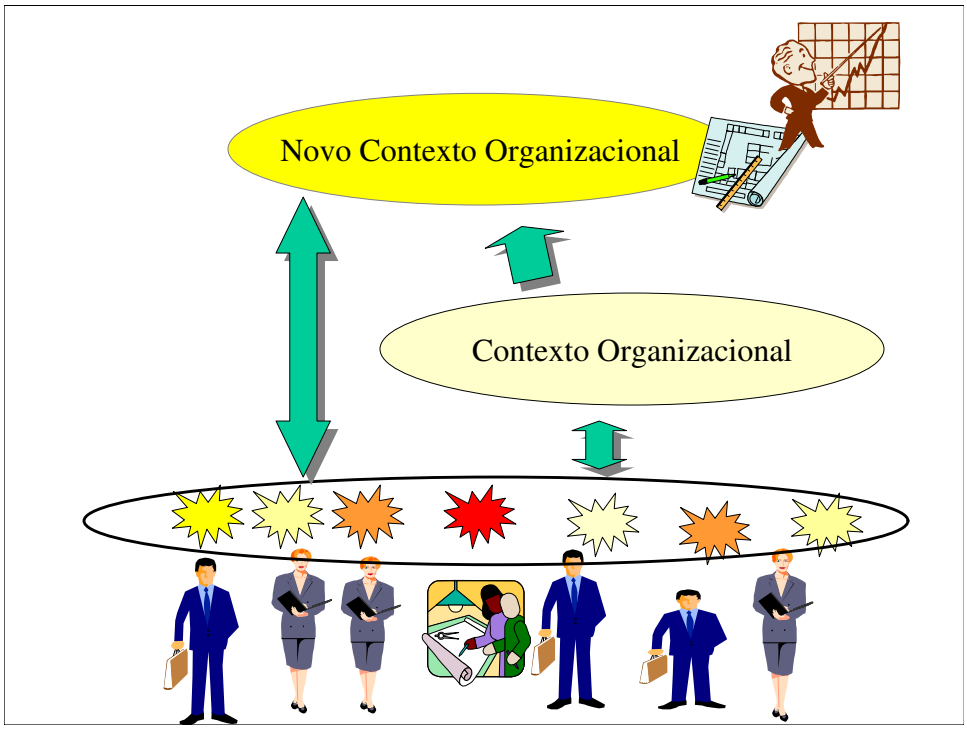


Figura 4.2 - Construindo um novo contexto social

Na próxima seção a construção do novo contexto organizacional é apresentada como um processo de aprendizagem mútua que envolve designers e usuários. Também são apresentados aspectos sobre a modelagem "do futuro" sobre o ponto de vista da SO. Estes aspectos servem de base para a construção do método apresentado no capítulo 5.

4.4 Construindo um Novo Contexto Organizacional

Atualmente informação e comunicação são os novos caminhos para a inovação em organizações. Ao mesmo tempo, as organizações devem lidar com novos desafios, uma vez que estas tecnologias alteram as práticas de trabalho e aumentam a necessidade de treinamento.

A efetividade de um artefato tecnológico no contexto organizacional é dependente da capacidade dos trabalhadores de percebê-lo como um veículo para o aprimoramento de suas práticas de trabalho e estabelecimento de novas práticas de trabalho. Atividades para o aprimoramento desta capacidade, incluindo treinamento, não são atribuições usuais dos designers. Entretanto, a articulação entre os designers e os funcionários de

uma organização pode promover a aprendizagem durante o processo de design. Nesta tese o conceito de aprendizagem é entendido não só como um resultado mas também como um processo.

Um sistema de CSCW não é compreendido como um sistema para treinamento ou um software que simplesmente automatiza as práticas de trabalho atuais, mas como um sistema que é parte de um contexto organizacional no qual a aprendizagem é necessária para estabelecer novas práticas de trabalho. Para tanto destacamos a necessidade do conhecimento compartilhado da organização, incluindo os aspectos formais e informais, e um método para representar as relações do sistema que está sendo construído com o contexto organizacional.

A abordagem proposta integra DP e SO em um trabalho coordenado entre os designers, trabalhadores e o departamento de Recursos Humanos (RH) (Bonacin e outros, 2003a). Esta abordagem visa promover a participação direta dos funcionários da organização no desenvolvimento do sistema durante as atividades de DP utilizando os métodos e formalismos da SO. Ela exibe a transição entre o que foi aprendido a partir das práticas de trabalho (pelas técnicas de DP) para a modelagem de um contexto organizacional desejado (utilizando as técnicas e formalismos da SO).

Ferramentas computacionais que permitam a integração dos times no trabalho cooperativo e promovam a aprendizagem construtivista podem compor um ótimo cenário de aprendizagem nas organizações. Entretanto, a tecnologia não é hábil para promover a aprendizagem por ela mesma. Nesta tese é explorada a necessidade de fundamentar a idéia de aprendizagem nas organizações em conceitos que conectam aspectos teóricos e práticos. O interesse é articular as condições sobre as quais as situações de aprendizagem possam ocorrer, não somente durante o uso do sistema, mas principalmente durante o processo de design do sistema para o contexto de trabalho. Alguns dos aspectos discutidos são:

- *A aceitação do sistema pelo usuário*. Conforme aponta Grudin (1994), o sucesso de um sistema que suporta atividades em grupo não é garantida pela tecnologia por ela própria, mas é diretamente influenciada pela aceitação no grupo de trabalho. Como promover esta aceitação?

- *Condições para o design cooperativo*. São necessários programas organizacionais que motivem a participação dos usuários em atividades em grupo e nas discussões a respeito das práticas de trabalho. Como promover a cooperação através de um processo de design?
- *Oportunidades oferecidas pelos sistemas em atividades de treinamento futuras*. Deve ser encontrada uma maneira de utilizar o sistema como uma ferramenta para aprimorar atividades de treinamento atuais e futuras. Como imaginar o sistema como uma oportunidade de aprendizagem futura?

A tabela 4.1 sumariza como DP, SO e a participação do departamento de RH de uma organização podem promover situações que vão ao encontro das questões apresentadas. Para cada um dos problemas apresentados o DP, a SO e o RH atuam de forma complementar contribuindo para a busca da solução. Na próxima seção é apresentada a fundamentação para promover a construção de um novo contexto organizacional por meio da aprendizagem promovida pelo design cooperativo.

Tabela 4.1: Papéis do Design Participativo, Semiótica Organizacional e participação do RH para a construção de um novo modelo organizacional

	Design Participativo	**Semiótica Organizacional**	**Participação do RH**
Aceitação do sistema pelo usuário	Os usuários podem alterar o sistema durante o desenvolvimento, eles têm influencia no design do sistema	Facilita a comunicação entre designers e usuários, e ambos podem ter um melhor entendimento das necessidades do usuário no contexto social de trabalho	Pode promover a participação do trabalhador nas atividades de design, refletindo o sentimento de que o sistema é resultado desta participação
Cooperação no design e o design para cooperação	O usuário participa da construção do sistema, facilitando a sua compreensão sobre os objetivos, problemas e métodos utilizados no design do sistema	O modelo da Semiótica pode ser utilizado para dirigir discussões sobre a ontologia do domínio e os objetivos do sistema	Facilita a interação e a discussão entre os grupos de trabalho e designers. Viabiliza a alocação de tempo para os funcionários nessas discussões
Oportunidades de aprendizagem trazidas pela abordagem	O DP pode ser utilizado para conduzir atividades que levam ao aprendizado durante o design do sistema	Os métodos da SO contêm artefatos para capturar aspectos da prática do trabalho e do modelo de software, para serem utilizados durante as atividades participativas	Facilita o uso do sistema como ferramenta para atividades de treinamento

4.4.1 Participação e Significação para Promoção da Aprendizagem Mútua

O design de aplicações computacionais que aprimorem não só a qualidade do produto, mas também a qualidade das práticas de trabalho, é um dos principais objetivos da abordagem do DP. A cooperação ativa entre usuários e designers tem sido constantemente defendida pelos especialistas em DP e muitas pesquisas foram conduzidas para estabelecer interações produtivas e com significado entre aqueles que são responsáveis diretos pelo design e uso da tecnologia (Schuler e Namioka, 1993).

O trabalho tem fundamentalmente natureza social e alguns dos aspectos sociais críticos dificilmente são abordados de maneira adequada na modelagem convencional de sistemas (Kyng, 1991). Para lidar com estes aspectos, nós adotamos uma visão de que um sistema computacional é parte do ambiente de trabalho humano e, portanto é necessário compreender o contexto social da organização no qual o trabalho é executado. O significado é construído como resultado da cooperação entre designers e trabalhadores (futuros usuários da tecnologia).

Abordagens convencionais para o desenvolvimento de tecnologia de informação apresenta uma separação entre o design, implementação e o uso do sistema. Estas abordagens assumem que existe um modelo conceitual do domínio comum, compartilhado por todos; desta maneira, o problema resume-se em identificar este modelo e codificá-lo. Entretanto, vários autores reconhecem o fato que modelos compartilhados do domínio não existem de fato mas são construídos socialmente pela prática (Fischer e outros, 1995;Kyng, 1991; Lave e Wenger, 1990).

Khun (1996) examinou maneiras pelas quais designers e usuários podem reformular o design e implementação de sistemas de tecnologia de manufatura. Ele propôs o *design centrado no usuário* que coloca as considerações humanas, sociais e organizacionais no mesmo nível que as considerações técnicas do desenvolvimento do sistema propriamente dito vendo os usuários finais (operadores, engenheiros, administradores) como um ponto central para um sistema de manufatura efetivo. De acordo com esta abordagem, para que uma tecnologia seja bem modelada ela deve fazer uso das forças humanas como habilidades, experiência, julgamentos, capacidade de aprendizagem, para criar um

sistema de produção robusto e flexível, ao invés de procurar minimizar e controlar a intervenção humana.

Kyng (1991) adota o termo "aprendizagem mútua" para enfatizar o fato de que ambos especialistas do domínio e desenvolvedores do sistema devem entrar em um processo de comunicação para construir um modelo inicial do domínio fundamentado na prática, e evolui-lo conforme cada parte aprende mais. A "aprendizagem mútua" implica que designers aprendam sobre o domínio da aplicação e especialistas do domínio aprendam sobre as novas oportunidades trazidas pela tecnologia. As interações entre participantes podem ser compreendidas em termos de relações semióticas estabelecidas por eles, uma vez que eles desenvolvem e compartilham um sistema de signos. Portanto, podemos estender este paradigma, com o objetivo de mostrar como o desenvolvimento de sistemas de informações pode ser abordado de maneira cooperativa por designers e trabalhadores.

Reconhecendo e pressupondo a necessidade de aprendizagem mútua entre as partes envolvidas, através de DP e SO podemos contribuir com idéias sobre o que o design no ambiente de trabalho poderia ser e quais papéis ele poderia ter em uma organização (Baranauskas e outros, 2002). Nesta perspectiva, o design cooperativo é o resultado do processo estabelecido na prática de um grupo no qual ocorre a aprendizagem mútua. Uma nova abordagem fundamentada em práticas participativas e no referencial da Semiótica apresenta-se como uma maneira de apoiar usuários e designers na aplicação de seus conhecimentos e experiências para o design do sistema.

4.4.2 Modelagem de um Contexto Organizacional Futuro

Quando aplicamos muitas das técnicas de Design Participativo, como por exemplo a *Starting Conference*, normalmente a situação atual é avaliada e grande parte da discussão (se não toda em alguns casos) diz respeito a uma situação futura. Aspectos da ferramenta computacional, do contexto organizacional, do design do sistema, entre outros, são discutidos através destas técnicas. Quando discutimos uma ferramenta antes de sua construção estamos falando do futuro; neste caso é quase que inevitável que em alguma hora no desenvolvimento do sistema o futuro seja discutido.

Estes aspectos têm impacto direto na construção do sistema, quando fazemos um modelo dele sempre estamos referenciando o futuro, pelo fato de que o sistema ainda não foi construído. Desta maneira estes sistemas não existem e não serão utilizados naquele momento, mas sim quando estiverem prontos. Nesta tese o sistema é proposto como parte de um contexto organizacional futuro, portanto designers e trabalhadores estão interessados em modelar o futuro com base no contexto atual.

Um diagrama de ontologia é um modelo do mundo "aqui-e-agora"; embora seja possível organizar os dados a respeito do passado e do futuro, o modelo contém apenas construções temporais restritas dadas pelas dependências ontológicas, que não dizem nada sobre como o passado e o futuro estão relacionados (Stamper, 2000). Mas, como modelar o futuro com um modelo feito para representar o mundo "aqui-e-agora"?

De acordo com Stamper (2000) não existe espaço para a realidade, exceto o "aqui-e-agora". "O passado e o futuro, são coisas idealizadas no horizonte, eles não existem "aqui-e-agora". Nós temos que criá-los usando signos para representá-los "aqui-e-agora". Ao contrário de tentar copiar as ciências naturais que podem continuar trabalhando com a velha ilusão de uma dimensão física do tempo, que elas vêem através das lentes transparentes dos sistemas de informações que utilizam, nós (profissionais de Sistemas de Informação) devemos investigar as propriedades destas lentes." (Stamper, 2000, p. 27)

Calendário e cronômetros são utilizados para gerar fluxo de eventos e intervalos nomeado-os de maneira conveniente. Hoje, o conhecimento sobre o passado e o futuro depende da habilidade "aqui-e-agora" para construir marcas no papel ou na memória. Para obter a variável de tempo utilizada na física, os eventos e intervalos são arranjados em uma ordem ou ordem parcial, com propriedades matemáticas adequadas. Segundo Stamper (2000) esta é a maneira como construímos o tempo, não temos nenhum conhecimento sobre o tempo a não ser os signos que utilizamos "aqui-e-agora" para representar o começo e o fim. Não é possível experimentar um evento como um *affordance* ou repertório de comportamento, pois uma vez aqui, ele já se foi.

Em uma técnica de DP os participantes estão discutindo sobre o passado e o futuro "aqui-e-agora" usando signos. Neste trabalho os diagramas de ontologias e a análise de normas são ferramentas para a discussão e para criar "aqui-e-agora" um futuro desejável para o contexto organizacional pelo uso de signos. Para "criar" o futuro pelo uso de

signos o grupo se engaja em um processo de "significação" (efeito que o *interpretante* final produz sobre o interprete) no qual o futuro é construído nos modelos e na mente de cada um dos participantes. Os supostos *affordances* e dependências ontológicas são modelados nos diagramas de ontologia e as normas modelam os aspectos dos níveis pragmático e social (incluindo causalidade).

4.5 Síntese e Considerações Finais do Capítulo

Neste capítulo foi discutido porque seria possível integrar o Design Participativo com a Semiótica Organizacional. O DP não é condicionado a uma técnica de modelagem, ou seja, podemos utilizar DP em conjunto com várias técnicas de modelagem; por outro lado, a SO não restringe a participação do usuário na construção de seus modelos, mais especificamente na Análise Semântica e de Normas eles são encorajados a participar embora não exista nenhuma sistemática em SO para promover esta participação.

Colocando DP e SO como abordagens complementares, o DP pode contribuir para a SO e vice-versa. A SO provê mecanismos para modelar aspectos da organização fundamentados na discussão entre designers e trabalhadores utilizando técnicas de DP que dificilmente podem ser modelados por uma visão objetivista. Por outro lado, o DP garante que os signos utilizados nos modelos da SO sejam resultantes dos agentes imersos no contexto organizacional.

Estamos propondo o uso de DP e SO como um instrumento para a construção de um novo contexto organizacional. A efetividade de um artefato tecnológico no contexto organizacional é dependente da capacidade dos trabalhadores de percebê-lo como um veículo para o aprimoramento de suas práticas de trabalho e para estabelecer novas práticas. Atividades para o aprimoramento desta capacidade, incluindo treinamento, não são atribuições usuais dos designers. Entretanto, a articulação entre designers e os funcionários de uma organização pode promover a aprendizagem durante o processo de design. Para tanto sugerimos que o DP, SO e a participação do Recursos Humanos de uma organização como veículo facilitador, pode possibilitar a aceitação do sistema pelo usuário, promovendo a cooperação no design e o design para a cooperação.

Através da significação e da participação, designers e trabalhadores entram em um processo chamado de "aprendizagem mútua" (Kyng, 1991). Neste processo os designers aprendem sobre o contexto organizacional e trabalhadores aprendem sobre as oportunidades trazidas pela tecnologia. Engajados neste processo, na abordagem proposta, eles podem construir "aqui-e-agora" um modelo do contexto organizacional desejado através do uso de signos presentes nos modelos da SO. O próximo capítulo apresenta o SPaM (Semiotic Participatory Method), um método de design que articula técnicas de DP e SO.

Capítulo 5

O Método Semiótico Participativo

Neste capítulo é apresentado um novo método que integra DP e SO de modo a capturar e representar as "funções do sistema de informação humano" (três níveis superiores do framework da Semiótica Organizacional) e construir sistemas computacionais de acordo com as necessidades encontradas. Para tanto, propomos combinar o uso de técnicas de DP com a Análise Semântica para explorar o nível semântico, e a Análise de Normas para explorar os níveis pragmático e social da organização. O objetivo final é a modelagem de um novo contexto organizacional no qual o sistema computacional é parte integrante. Este método incorpora novas técnicas e procedimentos que são apresentados nos capítulos seguintes.

5.1 Motivação para o SPaM

Não basta possuirmos um conjunto de técnicas e modelos isolados com diferentes propósitos e com resultados relevantes e complementares. É necessário modelar a organização e o sistema computacional como um todo; isto normalmente é tratado em um processo de desenvolvimento de software.

O *Rational Unified Process* (RUP), um processo de engenharia desenvolvido e comercializado pela Rational Software, tornou-se largamente utilizado pela indústria de software para o desenvolvimento de sistemas Orientados a Objetos (OO). O RUP (Kruchten, 1999) é uma instância específica e detalhada de um processo mais genérico, o Processo Unificado (UP - *Unified Process*) introduzido por Jacobson e outros (1999). O

RUP possui os detalhes requeridos para se desenvolver projetos utilizando o UP. Nesta seção o RUP é analisado em função do nível de participação do usuário e da sua adequação à modelagem semiótica.

Analisando a participação do usuário em um processo como o RUP, podemos dizer que ele foi construído segundo o paradigma do design *para* o usuário e não *com* o usuário. Apesar de ser possível incluir grupos de usuários em um *papel* (conceito empregado no UP para definir comportamento e responsabilidades de um grupo de indivíduos trabalhando juntos como um time), o RUP não inclui técnicas para estimular e coordenar a participação dos usuários nas atividades atribuídas aos papéis definidos no modelo de processo. Além disso, o RUP é baseado em seis práticas que segundo Booch (1999, pp. 6) são as melhores práticas de engenharia de software: (1) o desenvolvimento de software iterativo, (2) o gerenciamento de requisitos, (3) o uso de arquiteturas baseadas em componentes, (4) a modelagem visual, (5) verificação da qualidade do software e (6) controle de alterações. No RUP as chamadas "partes interessadas" (Stakeholders) têm atividades específicas e muito limitadas durante o processo, se comparadas com a participação do usuário promovida pelo DP.

O RUP tem suas bases na OO; grande parte dos artefatos definidos nos *workflows* (seqüência de atividades que produz um resultado de valor tangível) está ligada a representações orientadas a objetos. Não podemos simplesmente substituir os artefatos OO pelos modelos e técnicas da SO, uma vez que eles adotam posturas filosóficas diferentes, são construídos de maneira diferente e produzem resultados diferentes. Uma outra alternativa seria substituir alguns destes *workflows* por outros mais adequados à proposta da SO mantendo os outros *workflows* intactos. Entretanto esta alternativa não garante que haja uma articulação entre os resultados destes *workflows* para cumprir as fases previstas no RUP *(inicial, elaboração, construção* e *transição*).

Por estes motivos, no lugar de usar o UP ou transformá-lo descaracterizando-o, propomos um método com bases nos princípios de DP e SO, suficientemente flexível para ser articulado com outros conceitos e procedimentos de engenharia de software.

Um outro método de desenvolvimento de software que deve ser analisado no contexto desta tese é o Extreme Programming (XP) (Beck and Cunnigham, 19XX), um método que vem quebrando vários paradigmas tradicionais de Engenharia de Software,

entre eles o papel do usuário no desenvolvimento de software. O XP está baseado em quatro princípios principais: simplicidade, comunicação, feedback e coragem.

Apesar do XP ser um método de Engenharia de Software direcionado a abordar aspectos tecnológicos como, por exemplo, qualidade do código, interação entre os desenvolvedores e planejamento de versão, podemos dizer que DP e XP têm em comum várias motivações e abordagens. Assim como no DP, no XP os usuários têm um papel chave durante o processo de design, especificando e projetando o sistema em cooperação com os desenvolvedores.

Entretanto, por outro lado, em comparação do Design Participativo, o XP não fala sobre o processo de seleção dos participantes e presta pouca ou nenhuma atenção aos aspectos organizacionais relacionados ao modelo de participação do usuário. Ele não especifica exatamente como os usuários e designers interagem durante o processo; os jogos de planejamento no XP visam explorar com os usuários o ambiente de trabalho iterativamente através de estórias contadas pelos usuários. Já o DP emprega uma série de técnicas que possibilitam a construção de modelos sobre aspectos organizacionais pelos designers e trabalhadores com envolvimento mútuo.

Na comunidade de DP existe o reconhecimento do XP como método eficiente de Engenharia de Software e também o esforço para incluir aspectos comumente tratados em DP durante o desenvolvimento de software. Trabalhos como o de Rittenbruch e outros (2002) enfatizam possíveis extensões do XP pelo uso de técnicas de DP e seus benefícios como por exemplo: uma melhor compreensão do contexto durante o design, a melhor inteligibilidade das estórias relatadas pelos usuários e uma maior interação dos designers com os usuários durante a fase de planejamento. Entretanto, apesar destes esforços ainda não existe resultado conclusivo de como utilizar técnicas de DP em conjunto com XP.

A possibilidade de uso conjunto do XP e os métodos da SO esbarram no fato da pouca ou nenhuma atenção que é dada pelo XP à modelagem dos aspectos organizacionais. Enquanto o XP está baseado no uso de estórias contadas pelos os usuários das quais extraímos o comportamento desejado do sistema, a SO se baseia no estudo mais aprofundado da organização identificando, modelando e compreendendo o comportamento humano e social nestas organizações para que desta maneira possamos construir sistemas que as aprimorem. Desta maneira, SO junto com DP levam os usuários

e designers a uma visão mais ampla de seu ambiente não se restringindo a uma visão ligada apenas à tecnologia ou à implementação de um sistema.

Outro aspecto divergente do XP em relação a este trabalho, é particularmente relacionado à filosofia do DP de aprimorar não só o sistema mas também o contexto organizacional. O XP propõe a restrição do design e a implementação somente aos requisitos atuais e não pensar a respeito das "possibilidades" futuras, como meio de aumentar a produtividade no desenvolvimento do software. Em um contexto organizacional uma visão estratégica sobre as possibilidades de uso futuro é fator fundamental ao aprimoramento da organização e à sua vantagem tecnológica competitiva. Em via de regra, um sistema computacional não é construído para ser utilizado hoje, mas sim, por muitos anos; desta forma não podemos deixar de pensar no futuro. Embora projetar e implementar, pensando no futuro, possa trazer supostos aumentos de custos e dificuldades no andamento da implementação, esse é um preço que temos que pagar aos benefícios do aprimoramento organizacional que certamente serão maiores, contanto que sejam devidamente identificados e modelados. Desta maneira, o problema que o XP tenta evitar não é causado pelo fato de projetar e modelar aspectos que não são utilizados hoje, mas sim por criar funcionalidades que supostamente teriam impacto no futuro sem um planejamento adequado.

Portanto, assim como no caso do RUP, não pretendemos modificar o XP e transformá-lo em algo que ele não é, para adequar o uso conjunto de DP e SO.

5.2 A Descrição do SPaM

Nesta seção, inicialmente, o SPaM é descrito em linhas gerais e são apresentados os principais conceitos e fases. Em seguida, as suas fases são descritas de maneira mais detalhada. O conceito base adotado no SPaM (Semiotic Participatory Method) é o de design iterativo e prototipação. Entre outros benefícios que esta prática propicia (Pressman, 95) estão a construção de protótipos rápidos, que possibilitam que designers e trabalhadores discutam de maneira mais clara os efeitos da ferramenta no contexto organizacional e também a avaliação antecipada das dificuldades e riscos envolvidos no

projeto. No SPaM o ciclo de prototipação está divido em fases definidas em função dos conceitos trabalhados pelas técnicas de DP e SO com o intuito de atingir gradualmente um novo contexto organizacional.

A figura 5.1 (Bonacin e Baranauskas, 2003a) ilustra o método semiótico participativo. A seta circular representa o modelo de desenvolvimento de software como um processo cíclico que contém quatro fases principais:

1. *Design e revisão do contexto organizacional.* Nesta fase do ciclo de prototipação designers e usuários tentam juntos compreender e modelar o contexto organizacional, revisar a práticas de trabalho e propor um novo modelo para este contexto a partir das oportunidades trazidas pela tecnologia. Após a primeira iteração no ciclo de prototipação, nesta fase, usuários e designers avaliam os modelos organizacionais e o protótipo construído durante o ciclo anterior. O plano para atingir o contexto organizacional desejado a partir do atual também é revisado a cada ciclo. Além disso, os modelos semântico e de normas são construídos, discutidos e revisados para descrever a organização e as práticas de trabalho desejadas;
2. *Levantamento de requisitos do sistema e design de alto nível.* Na segunda fase, designers e usuários identificam os *affordances* que constituem um ambiente de trabalho futuro que o sistema deve dar suporte. Logo após, ainda durante esta fase, são discutidas e avaliadas alternativas para suporte às práticas de trabalho relacionadas aos *affordances* identificados. Esboços das principais telas da interface do sistema podem ser construídas e/ou alteradas com o intuito de clarificar o problema e discutir possíveis alternativas para a construção e refinamento do protótipo. Os diagramas de ontologias são traduzidos para diagramas iniciais da tecnologia empregada para a construção do protótipo;
3. *Design detalhado do sistema.* Na terceira fase, o projeto do sistema computacional é construído com base na representação do contexto organizacional e funções do sistema descritas na fase anterior. O modelo construído durante a fase anterior é refinado, expandido e detalhado para constituir um modelo de projeto; por exemplo, nesta fase os diagramas

UML (Unified Modelling Language) derivados dos diagramas de ontologia são trabalhados para tornarem-se diagramas de projeto completos caso a tecnologia adotada seja OO. A participação do usuário também é encorajada durante esta fase; alternativas de projeto devem ser discutidas e avaliadas por eles;

4. *Construção do protótipo.* Na última fase o protótipo é implementado. A participação do usuário não está restrita à avaliação e validação do sistema; ele deve participar de todas as decisões técnicas que afetam de alguma forma o uso do sistema e conseqüentemente as suas práticas de trabalho, através do uso de técnicas de DP. O usuário também deve ser estimulado a participar da implantação do sistema na organização.

No primeiro ciclo de prototipação a carga de trabalho está extremamente concentrada na primeira fase; no decorrer do tempo ela vai se deslocando para as demais fases até que ao final o trabalho esteja concentrado na quarta fase. Conforme exibe a figura 5.1 a avaliação não é uma fase especial. Ela é incluída como parte de todo o ciclo de desenvolvimento com o objetivo de engajar o grupo em um processo de resignificação a respeito dos elementos envolvidos em todo o processo de design.

O SPaM utiliza DP para dar suporte à modelagem semiótica que, por sua vez resulta em modelos e protótipos que podem ser utilizados em outras atividades de DP, como uma ferramenta participativa, explorando oportunidades para aprimorar as práticas de trabalho atuais. Os modelos resultantes e protótipos são continuamente submetidos a atividades participativas que incluem não só aspectos ligados à avaliação do sistema computacional mas também a avaliação do contexto organizacional e do próprio processo de design, permitindo a exploração de novas alternativas para o design do sistema e para aprimorar a organização.

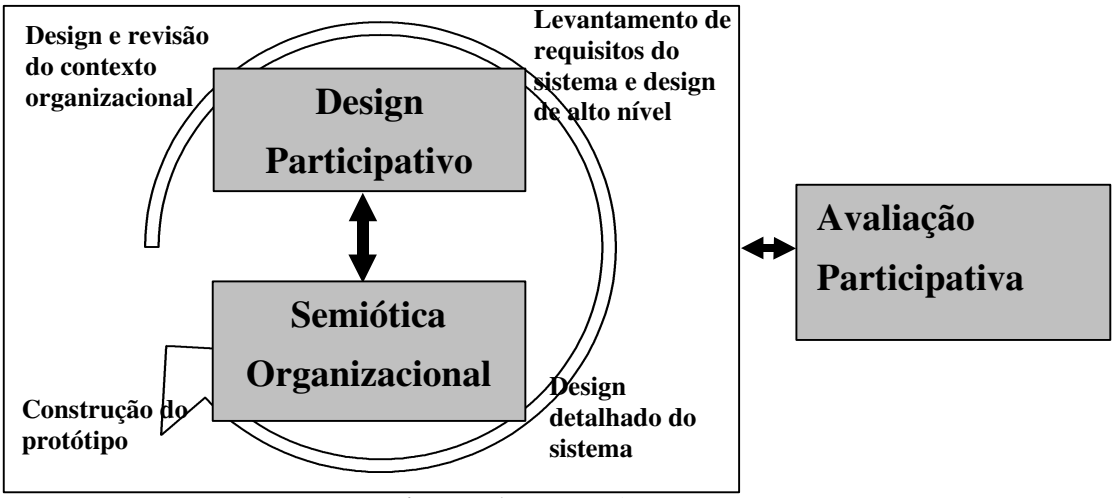


Figura 5.1 - O Método SPaM

A avaliação do processo como um todo é realizada pela relação mútua entre DP, SO e atividades de avaliação do sistema; esta relação é representada pelas setas preenchidas na figura 5.1. As técnicas de DP dão suporte à construção do modelo semiótico e protótipos que são submetidos a avaliação e recebem o *feedback* para a prática de novas técnicas participativas que complementam o modelo e aprimoram o processo de design do protótipo. Da mesma maneira, o modelo da SO captura conceitos das práticas de DP, e suporta a modelagem de protótipos que são avaliados (modelo e processo) recebendo o *feedback* para a reconstrução dos modelos semióticos na iteração seguinte.

A figura 5.2 apresenta em detalhes a primeira fase do ciclo de prototipação. Fazendo uma leitura da esquerda para direita temos: os artefatos que são trabalhados a cada passo, uma breve descrição do passo e os principais artefatos resultantes. A seta de cima para baixo representa o tempo e a ordem sugerida para realizar cada passo.

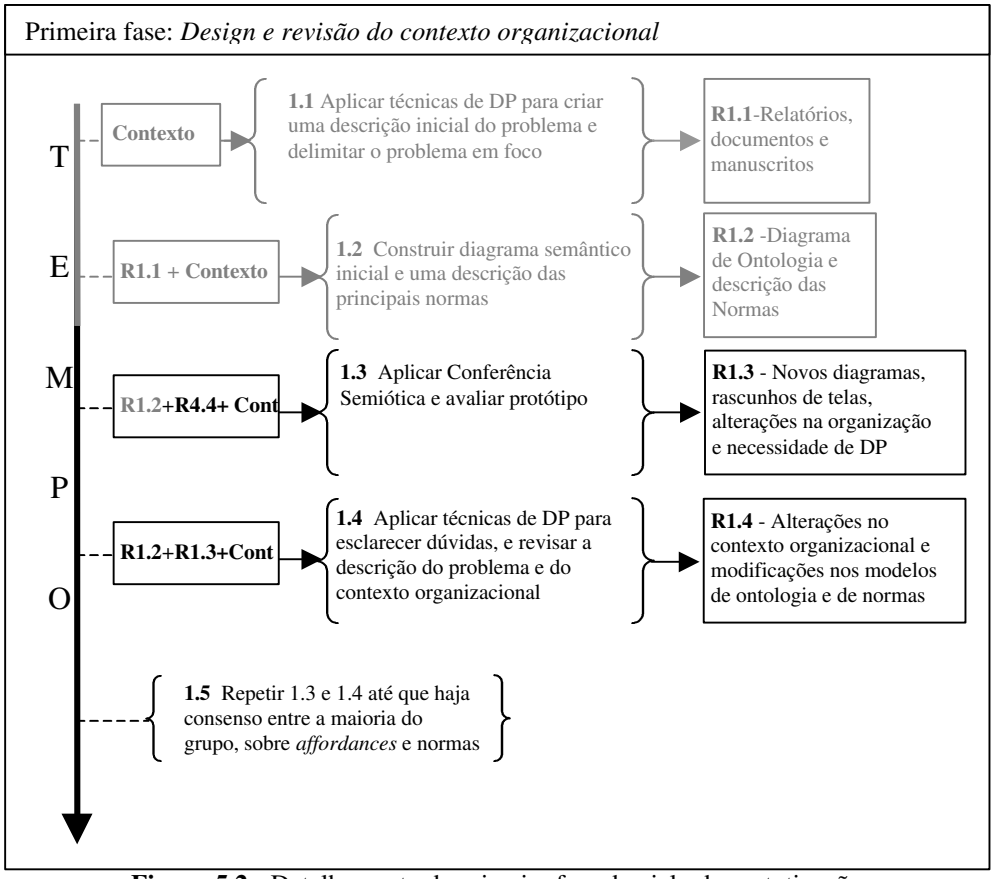


Figura 5.2 - Detalhamento da primeira fase do ciclo de prototipação

As partes em cinza na figura 5.2 indicam que os passos 1.1 e 1.2 são realizados somente durante o primeiro ciclo de prototipação. A seguir é apresentada a descrição de cada um dos passos que constituem a primeira fase (exibidos na figura 5.2):

1.1 Neste passo, devem ser utilizadas técnicas de DP para delinear o problema em foco e elaborar uma definição inicial do problema. Para tanto devem ser aplicadas técnicas de design participativo que possuem como objetivo principal discutir e entender a situação atual da organização. Analisando o conjunto de técnicas apresentadas em Müller e outros (1997) (onde podemos encontrar a descrição resumida das técnicas) sugerimos algumas que podem ser utilizadas neste passo de acordo com necessidades e características específicas do problema: *Starting Conference, Contextual Inquiry*, *Ethnographic Practices, Future Workshop, Artifact Walkthrough, CUTA, Group Elicitation Method, PictureCARD, Icon Design Game (adaptado para todos os tipos de signos), ORBIT,*

Scenarios, Translators e *Work Mapping*. Estas técnicas estão ordenadas em função do início sugerido para sua aplicação. Certamente na maioria dos casos não é viável utilizar todas as técnicas propostas. A escolha sobre qual técnica aplicar é dependente do problema e dos recursos disponíveis. Todo material resultante destas técnicas deve ser devidamente registrado para que este possa ser utilizado no decorrer dos próximos passos e fases;

1.2 Uma descrição inicial do problema deve ser criada em conjunto com os usuários. Esta descrição é baseada nos resultados das técnicas utilizadas no passo anterior. Os designers devem construir um diagrama de ontologia inicial utilizando a seqüência proposta por Liu, 2000 (ver figura 3.6) para a análise semântica; entretanto destacamos que a descrição do problema não é a única base para a construção do diagrama de ontologia. Todo material resultante da aplicação das técnicas de DP deve ser analisado para a definição da ontologia do domínio. Um modelo de normas inicial também é derivado desta descrição;

1.3 Neste passo, a Conferência Semiótica (detalhada na seção 5.4) deve ser aplicada para discutir o diagrama de ontologia e as normas associadas a ele (construído no passo anterior ou no último ciclo de prototipação) e garantir que ele seja resultante do entendimento construído pelo grupo. Telas do protótipo produzido no último ciclo (caso não seja o primeiro ciclo) também são discutidas durante a Conferência Semiótica. Técnicas de DP são utilizadas para avaliar o protótipo (caso haja um), algumas técnicas sugeridas são: Participatory Heuristic Evaluation, Cooperative Evaluation, ACOST project, CARD,CISP e Storyboard Prototyping;

1.4 A Conferência Semiótica aplicada no passo 1.3 pode indicar, como resultado, a necessidade de se rever conceitos presentes no contexto organizacional. Técnicas de DP devem ser utilizadas para rever conceitos e remodelar o contexto; estas técnicas devem ser escolhidas de acordo com as características e complexidade dos conceitos a serem revistos. As técnicas de DP sugeridas no passo 1.1 são exemplos de técnicas que

podem ser utilizadas neste passo. Todas as alterações devem ser refletidas no modelo de ontologia;

1.5 Os passos 1.3 e 1.4 devem ser repetidos até que haja consenso da maioria do grupo a respeito dos *affordances* e normas modeladas. Este é um ponto chave para que a democracia no ambiente de trabalho seja respeitada e para que os diagramas sejam resultado da prática subjetivista da modelagem do domínio.

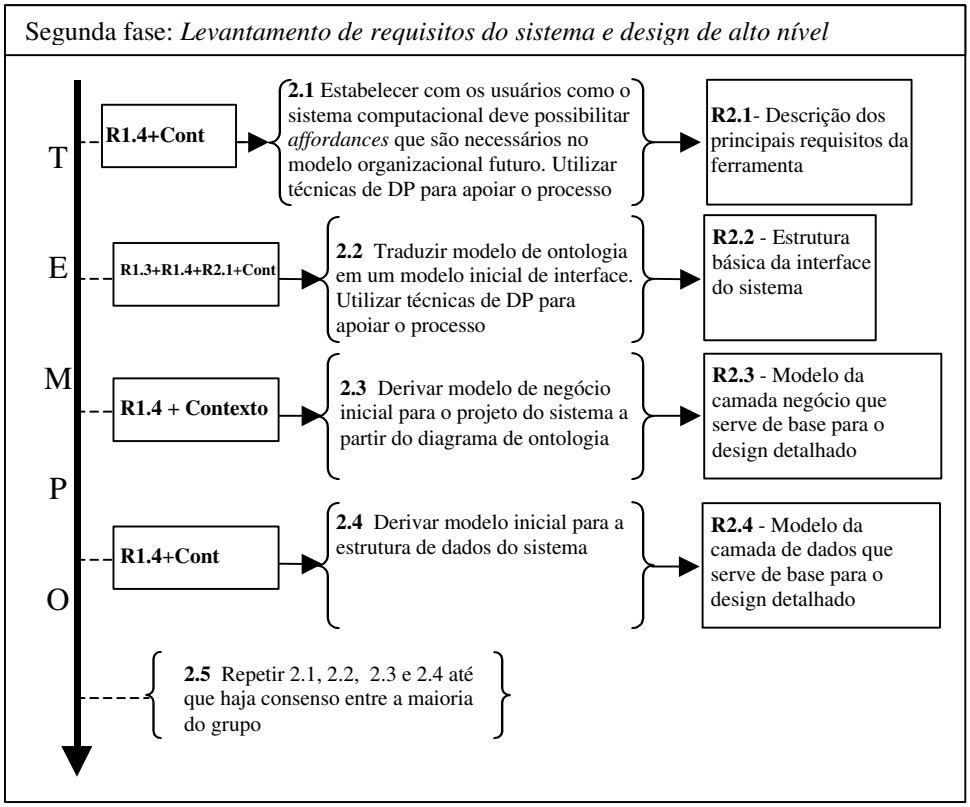


Figura 5.3 - Detalhamento da segunda fase do ciclo de prototipação

A seguir é apresentada a descrição dos passos contidos na segunda fase (exibidos na figura 5.3):

2.1 Estabelecer, em conjunto com os usuários, como o sistema computacional deverá possibilitar que os agentes possuam os *affordances* que constituem o modelo organizacional futuro. É necessário que o sistema seja definido como parte integrante deste contexto, explicitando como ele deveria dar suporte a atividades humanas e quais são os requisitos para possibilitar o

aprimoramento da organização. Técnicas de DP devem ser utilizadas para incluir os usuários nesse processo. Exemplos de técnicas que podem ser aplicadas a esta fase são: Artifact Walkthrough, Blueprint Mapping, Conceptual Toolkit in CSCW design, Contextual Inquiry, Card, Cooperative Requirements Capture, Ethography, Florence Project, Future Workshop, Fórum Theatre, Mock-Ups, ORBIT, PictureCARD, Starting Conference, Translators, Layout, Organization e Specification Games. Estas, entre outras técnicas de DP, devem ser utilizadas para clarificar os requisitos do sistema, de acordo com o problema e os recursos disponíveis;

2.2 A Conferência Semiótica aplicada no passo 1.3 resulta, muitas vezes, em rascunhos que propõem modificações no protótipo; estas modificações devem ser refletidas nele. A interface do sistema também deve refletir os requisitos levantados no passo anterior. Para a modelagem da interface é proposto o uso de técnicas de DP, por exemplo: Contextual Design, Icon Design Game, HOOTD, Lunchbox Project e PICTIVE. A seção 6.3 sugere uma arquitetura que incorpora aspectos da análise semântica e de normas à interface do sistema;

2.3 Com base na última versão do diagrama de ontologia, construir um diagrama inicial de projeto, no de negócio, que deverá ser trabalhado na próxima fase. Este diagrama inicial deve estar de acordo com a metodologia empregada no projeto do sistema. A seção 6.2 sugere como construir um diagrama inicial de classe a partir dos resultados da análise semântica;

2.4 Com base na última versão do diagrama de ontologia, construir uma especificação inicial da estrutura de dados que deverá ser trabalhada na próxima fase. Esta especificação inicial deve estar de acordo com o procedimento empregado no projeto do sistema. A seção 6.1 sugere como construir um modelo inicial para banco de dados relacionais a partir do diagrama de ontologia;

2.5 Os passos 2.1, 2.2, 2.3 e 2.4 devem ser repetidos até que haja consenso da maioria do grupo a respeito dos modelos do sistema a ser construído.

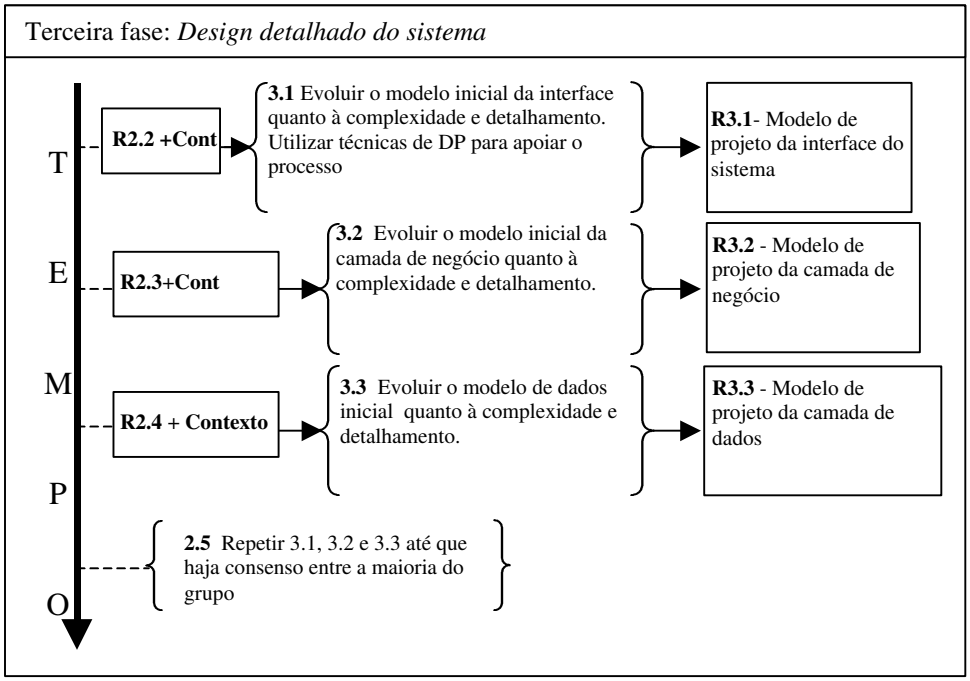


Figura 5.4 - Detalhamento da terceira fase do ciclo de prototipação

A seguir é apresentada a descrição dos passos que constituem a terceira fase do processo (exibidos na figura 5.4):

3.1 Evoluir a estrutura básica da interface do sistema, construída durante o passo 2.2, tanto em relação à complexidade, quanto em relação ao seu nível de detalhe. Neste passo são empregadas técnicas de projeto de interfaces e deve ser construído um modelo de projeto dependente da tecnologia empregada. Por exemplo, se for utilizada uma linguagem OO as classes para implementar o modelo proposto devem ser especificadas. Técnicas de DP podem ser utilizadas para auxiliar o processo, algumas técnicas são: Artifact Walkthrough, Buttons Project, Collaborative Design Workshops, Contextual Design, HOOTD, Icon Design Game e Interface Theatre;

3.2 Evoluir o modelo inicial da camada de negócio construído durante o passo 2.3 quanto à sua complexidade e detalhamento. Neste passo são empregadas técnicas de projetos de acordo com a tecnologia utilizada. Por

exemplo, podemos utilizar projeto OO se o sistema for implementado em uma linguagem OO. Técnicas de DP podem ser utilizadas para promover a discussão sobre decisões de projeto; estas técnicas são escolhidas de acordo com a necessidade;

3.3 Evoluir o modelo inicial da camada de dados construído durante o passo 2.3 quanto à sua complexidade e detalhamento. Neste passo são empregadas técnicas de projeto de acordo com a tecnologia utilizada. Técnicas de DP podem ser utilizadas para promover a discussão sobre decisões de projeto. Estas técnicas são escolhidas de acordo com a necessidade;

3.4 Os passos 3.1, 3.2, e 3.3 devem ser repetidos até que haja consenso da maioria do grupo a respeito dos modelos e especificações do projeto do sistema.

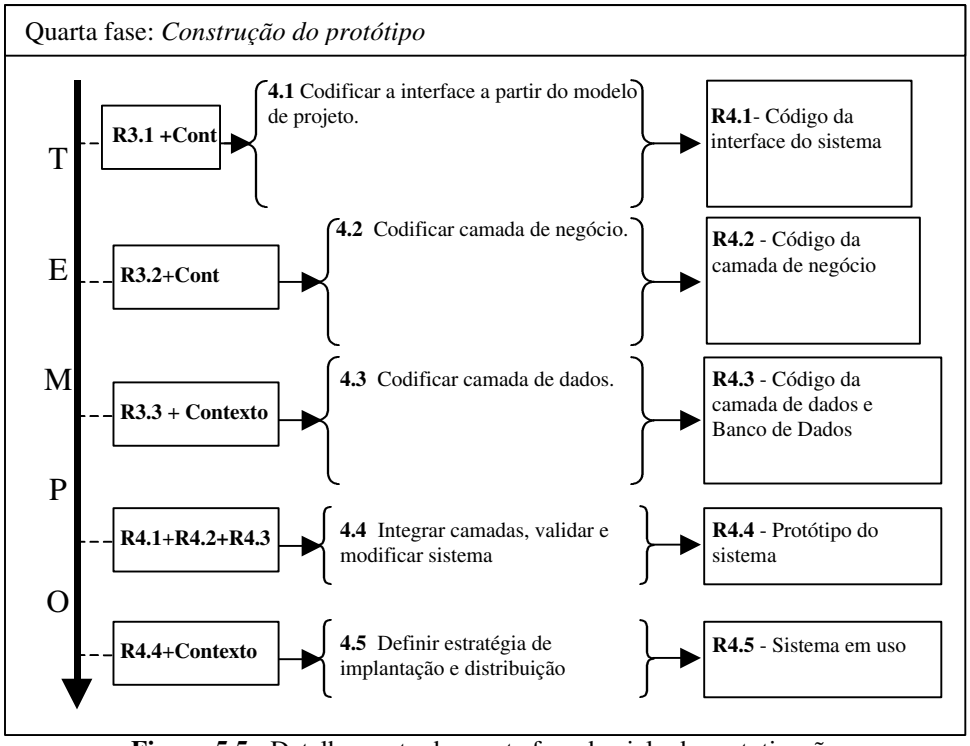


Figura 5.5 - Detalhamento da quarta fase do ciclo de prototipação

A seguir é apresentada a descrição dos passos que constituem a quarta fase do processo (exibidos na figura 5.5):

4.1 Codificar a interface a partir do projeto detalhado. Neste passo são empregadas técnicas para a implementação do sistema de acordo com a tecnologia utilizada, por exemplo linguagens OO. Técnicas de DP podem ser aplicadas para promover a discussão sobre decisões de implementação, caso estas afetem de alguma forma o uso do sistema e conseqüentemente as práticas de trabalho. As técnicas de DP são escolhidas de acordo com a necessidade;

4.2 Codificar a camada de negócio a partir do projeto. Técnicas de DP podem ser utilizadas para promover a discussão sobre decisões de implementação caso estas afetem de alguma forma o uso do sistema e conseqüentemente as suas práticas de trabalho; estas técnicas são escolhidas de acordo com a necessidade;

4.3 Codificar a camada de dados a partir do projeto. Técnicas de DP podem ser utilizadas para promover a discussão sobre decisões de implementação caso estas afetem de alguma forma o uso do sistema e conseqüentemente as suas práticas de trabalho; estas técnicas são escolhidas de acordo com a necessidade;

4.4 Integrar as três camadas e realizar testes de integração. Técnicas de validação (verificar se o software está de acordo com os requisitos) podem ser aplicadas neste passo. Modificar o que for necessário com a participação do usuário;

4.5 Definir a estratégia de implantação e distribuição, que pode variar desde a instalação nas máquinas de alguns usuários-pilotos para os primeiros protótipos do sistema até a distribuição em larga escala de uma última versão do sistema. Os usuários também devem participar diretamente deste passo.

5.3 Aspectos Práticos do SPaM

Antes de aplicar o SPaM, as condições necessárias ao Design Participativo devem ser consideradas. Conforme destacado anteriormente, para que o DP ocorra é necessário

que os trabalhadores estejam dispostos a investir o esforço necessário e ir além das suas atribuições habituais para obter melhorias em seu ambiente de trabalho. Como o SPaM pressupõe a participação do usuário em todas as fases de desenvolvimento, sua aplicação torna-se pouco favorecida quando trabalhadores não estão dispostos a engajar-se no desenvolvimento do sistema.

As técnicas de DP apresentadas na última seção são apenas sugestões baseadas na sua descrição e na sua aplicação em um contexto real. Quando o SPaM é utilizado, a decisão de aplicá-las deve ser resultado de discussões durante o desenvolvimento do sistema. Além disso no SPaM algumas técnicas são sugeridas para serem utilizadas durante uma certa fase, porém em muitos casos os seus resultados podem também ser úteis a outras fases, por isso, é recomendado sempre guardar os resultados decorrentes do uso das técnicas. Além disso, pelas próprias características das técnicas de DP, elas também devem ser adaptadas ao contexto do projeto.

O SPaM é um método baseado em prototipação com quatro fases em cada ciclo. Conforme destacado, a ênfase em cada fase desloca-se no decorrer do tempo da primeira para última. Como um método de prototipação é recomendado estabelecer metas a serem atingidas no final de cada ciclo. Uma estratégia para que o protótipo esteja sempre evoluindo, é dosar adequadamente a ênfase em cada fase do ciclo, a partir destas metas. No primeiro ciclo muitas vezes não é necessário um protótipo funcional, por isso é proposto que neste caso parte (ou toda) da terceira e quarta fase sejam substituídas pela aplicação de técnicas de DP que possibilitem criar protótipos rápidos. Algumas técnicas de DP que podem ser empregadas neste caso são: ACOST Project, CISP, Codevelopment, JAD e Storyboard Prototyping (em protótipos construídos pelo BrainDraw, CARD, CUTA, Mock-Ups e PICTIVE). Por outro lado, nos últimos ciclos, a primeira e a segunda fase podem focar a avaliação do sistema em uso no contexto organizacional como alavanca para oportunidades futuras de aprimoramento das práticas de trabalho atuais.

5.4 A Conferência Semiótica

Na primeira fase do SPaM é proposto o design e a revisão do contexto organizacional através do uso do DP e da SO e na segunda são propostos o levantamento de requisitos e a modelagem em alto nível. A Conferência Semiótica é uma nova técnica de DP proposta para permitir que os usuários participem diretamente e democraticamente da construção dos modelos propostos na SO. Nesta tese, a Conferência Semiótica é vista como parte do método SPaM. Porém, ela também pode ser utilizada de maneira independente do método de design empregado (Bonacin e Baranauskas, 2003b). Nesta seção são apresentados os principais aspectos teóricos e práticos da Conferência Semiótica.

5.4.1 Motivação para a Conferência Semiótica

A literatura em Engenharia de Software e Sistemas de Informação (Floyd, 1987; Müller et al 1998; Mahemoff e Johnston, 1998) distingue duas perspectivas básicas para o design de software: a perspectiva *orientada a produto*, que foca o artefato de software, e a perspectiva *orientada a processo*, que foca o processo de trabalho humano para o qual o artefato computacional pretende dar suporte. Como Floyd (1987), argumentamos que estas abordagens deveriam ser complementares e consideradas de maneira balanceada. Para Floyd, a perspectiva orientada a produto vê o software como um produto que existe por si próprio e consiste de um conjunto de programas e documentos que o especifica. Nesta visão o mundo é estático, e possui dois estados: antes e depois do sistema de software ser introduzido. Por outro lado, a perspectiva *orientada a processo* vê o software em conexão com a aprendizagem humana, trabalho e comunicação, em um mundo em constante evolução no qual as necessidades mudam.

O contato próximo entre os designers e usuários, e a participação direta dos últimos durante o desenvolvimento do sistema são apontados como uma maneira de promover o design orientado a *processo*. Nesta seção, é proposta uma técnica de design participativo chamada Conferência Semiótica como uma prática que possibilita explorar aspectos

orientados a *processos* do contexto social do trabalho e suas conexões com aspectos ligados ao produto que está sendo construído. A Conferência Semiótica foi proposta com o intuito de promover aspectos de significação e participação (Baranauskas e outros, 2002a) durante o processo de design. Esta técnica também possibilita que os designers balanceiem as perspectivas orientadas a *produto* e *processo* quando utilizam o SPaM.

O software é um produto da interação social e um novo contexto organizacional emerge como uma conseqüência do design do software e seu uso. Esta técnica foi motivada pelo conceito de aprendizagem mútua (Kyng, 1991), que é promovida na técnica através da participação do usuário e da construção de um modelo de ontologia durante o processo de significação.

5.4.2 Descrição da Conferência Semiótica

A Conferência Semiótica pode ser descrita em termos de seis atributos propostos por Müller (1997;1998) para descrever técnicas de DP: (1) o *modelo de objeto* que descreve os materiais utilizados na prática, (2) o *modelo de processo* que descreve como as pessoas se comunicam, como elas tomam decisões e como manipulam o material descrito no modelo de objetos, (3) o *modelo de participação* que descreve quem está envolvido na técnica e qual o papel desempenhado, (4) *resultados esperados* que devem conter os benefícios tangíveis conseguidos através do uso da técnica, (5) *posição da técnica no ciclo de vida do produto* e (6) *tamanho do grupo*. Na Conferência Semiótica estes atributos são:

- ***Modelo de Objeto:*** Cópias dos modelos da Semiótica Organizacional particularmente diagramas de ontologia (análise semântica) e o resultado da análise de normas (análise do nível pragmático e social), telas do protótipo impressas em papel e canetas são os materiais utilizados para aplicar esta técnica de design participativo.
- ***Modelo de processo:*** Em formato de uma conferência, os designers exibem os conceitos do local de trabalho modelados pelo diagrama de ontologia que são possíveis e/ou alterados pela proposta do sistema em um novo contexto organizacional. O diagrama de ontologia deve conter somente signos utilizados pelos usuários em seu dia-a-dia. Este diagrama pode ter sido

construído previamente pelos designers através de resultados de outras técnicas de DP (como ocorre no primeiro ciclo do SPaM) ou pode ter sido construído como resultado do último ciclo de prototipação (como ocorre nos demais ciclos do SPaM). Cópias dos modelos são distribuídos para todos os participantes e, caso necessário, o designer deve clarificar a notação e conceitos descritos pelo diagrama. O designer deve então ler o modelo de ontologia para o grupo. Durante esta leitura, para cada conceito existente no modelo que alguma pessoa do grupo aponte que é importante revisar, os participantes discutem as dependências ontológicas com outros conceitos e as normas formais e informais associadas. Os membros do grupo, como resultado da discussão, propõem alterações nos modelos de ontologia e de normas. Cópias em papel da interface do protótipo do sistema (caso exista um) devem ser distribuídas para possibilitar a discussão de como as alterações nos modelos podem ser refletidas na interface do sistema. As modificações na interface do protótipo podem ser diretamente desenhadas sobre as telas impressas, ou se necessário, outras técnicas de DP são utilizadas para modelá-las.

- ***Modelo de Participação.*** Participam da conferência membros de diferentes níveis e funções na organização junto com o time de design. Caso necessário podem participar facilitadores para mediar a interação entre os usuários e designers.
- ***Resultados:*** São produzidos modelos semânticos e de normas revisados durante a conferência por pessoas do contexto organizacional. Como resultado da aplicação desta técnica também temos o estabelecimento das normas sociais que possivelmente deverão ser válidas no ambiente de trabalho com o sistema que está sendo construído. A interface do sistema é construída através da "aprendizagem mútua" promovida pela aplicação da técnica.
- ***Posição no ciclo de vida do produto:*** No SPaM esta técnica pode ser aplicada várias vezes durante a fase de *design e revisão do contexto organizacional;* caso necessário para clarificar algum conceito ela também

pode ser utilizada para o *levantamento de requisitos do sistema e design de alto nível*. Se aplicada em um processo tradicional ela pode ser utilizada durante a Identificação e Clarificação do Problema, Análise de Requisitos e Sistema, Design de Alto Nível, Avaliação e Reengenharia.

- ***Tamanho do Grupo:*** 4-10 pessoas (o número de designers não deve ultrapassar o número de usuários)

5.4.3 Aspectos Práticos da Conferência Semiótica

Como a Conferência Semiótica é uma técnica de DP, é necessário considerar as condições necessárias ao DP antes de aplicá-la em um contexto específico. Um outro aspecto prático está relacionado aos modelos da SO: durante a aplicação da técnica são distribuídos para os usuários diagramas de ontologia, que servem de base para discussão do grupo. Desta maneira uma questão natural seria se é ou não necessário um treinamento prévio dos usuários para entender os diagramas da SO.

A necessidade de treinamento, na prática, poderia em muitos casos inviabilizar a aplicação da técnica em contextos reais devido ao tempo de treinamento e motivação dos usuários para aprender um modelo de design. Entretanto, dados empíricos (Anexo A) demonstraram que não é necessário treinamento prévio, visto que os participantes aprendem no decorrer da própria aplicação da técnica. É necessário apenas em um primeiro momento uma explicação sobre o que é o diagrama de ontologia, quais são os seus propósitos e como "ler" os conceitos representados. Esta explicação deve ser rápida (não mais que 20 minutos) e acompanhada de exemplos de diagramas reais inspirados no contexto organizacional. Além disso, conforme está descrito no modelo de processo da Conferência Semiótica, quem lê o diagrama de ontologia no início da conferência é o designer.

Em um primeiro momento basta que os usuários acompanhem a leitura do diagrama para serem capazes de discutir e propor soluções relacionadas ao contexto organizacional modelado, uma vez que este é o aspecto mais relevante. As discussões entre usuários e designers resultam em mudanças que devem ser refletidas no modelo. Nas primeiras seções os designers poderão fazer alterações no modelo junto dos usuários e "reler" o

novo modelo para o grupo. No decorrer das iterações de design (cerca de duas ou três conferências) os usuários tornam-se aptos a propor mudanças diretamente no modelo; desta maneira eles podem não só discutir e propor soluções relacionadas ao contexto organizacional, mas também refleti-las no modelo e apresentá-las ao grupo para discussão.

Outro aspecto relevante à aplicação da Conferência Semiótica diz respeito ao modelo de participação. O modelo propõe a participação de usuários de diferentes níveis hierárquicos e diferentes funções, para que todas as classes de trabalhadores (quando possível) que são afetadas pela introdução do sistema tenham o direito democrático de opinar sobre as decisões de design. Entretanto, viabilizar esta participação nem sempre é fácil; portanto, propomos um caminho para permiti-la, através da participação do departamento de Recursos Humanos (RH) como facilitador para viabilizar as primeiras conferências até que elas sejam entendidas pela organização e trabalhadores como parte de seu trabalho e não como atividade extra. Conforme exibe a tabela 4.1 (capítulo 4), o RH pode facilitar a interação e a discussão entre os grupos de trabalho e designers e também viabilizar a alocação de tempo para os funcionários.

5.5 Síntese e Considerações Finais do Capítulo

Neste capítulo foi apresentado o SPaM (Semiotic Participatory Method), uma proposta que visa integrar DP e SO em um método de design único capaz de capturar e representar "funções do sistema de informação humano" (três níveis superiores do framework da semiótica organizacional) e construir sistemas computacionais de acordo com as necessidades encontradas. Para tanto, propusemos combinar o uso de técnicas de DP com a Análise Semântica para explorar o nível semântico, e a Análise de Normas para explorar o nível pragmático e social da organização. O objetivo final é a modelagem de um novo contexto organizacional no qual o sistema computacional é parte integrante. Este método incorpora novas técnicas e procedimentos.

No SPaM o modelo de desenvolvimento de software é um processo cíclico que contém quatro fases principais que vão desde a modelagem da organização até a implantação do sistema. As fases do SPaM são: (1) Design e revisão do contexto

organizacional, (2) Levantamento de requisitos do sistema e design de alto nível, (3) O design detalhado do sistema e (4) Construção do protótipo. Entre os aspectos práticos ligados ao uso do SPaM está a necessidade e interesse do trabalhador em participar das decisões e do desenvolvimento do sistema, a escolha das técnicas de DP de acordo com o problema e recursos disponíveis, e a necessidade de estabelecer metas para cada ciclo de prototipação.

Métodos e técnicas para o design e desenvolvimento de sistemas são tradicionalmente construídos sobre o paradigma objetivista, que considera uma realidade objetiva a ser descoberta, modelada e representada no software. Diferentemente, a SO adota o paradigma subjetivista, que compreende a realidade como uma construção social baseada no comportamento dos agentes que participam dela. Esta perspectiva é explorada na Conferência Semiótica, que utiliza modelos da Semiótica Organizacional e telas impressas do sistema (protótipo em papel) como uma ferramenta para promover a aprendizagem mútua e explorar aspectos ligados ao processo e ao produto. Entretanto, para utilizar a técnica, alguns aspectos práticos devem ser considerados tais como: as condições para aplicá-la, como o usuário poderia começar a compreender os modelos da SO utilizados na técnica e como viabilizar o modelo de participação proposto.

Capítulo 6

De Modelos Organizacionais Para Sistemas Computacionais

Neste capítulo são propostos e apresentados procedimentos para construir o sistema a partir da Análise Semântica e de Normas. O SPaM propõe uma arquitetura de três camadas (Sommerville, 2000) para a construção de sistemas; estas camadas são: dados, negócio e interface. Portanto, é apresentado como construir cada uma delas: a seção 6.1 apresenta as alternativas existentes para construir modelos e implementar banco de dados baseados na SO (utilizada no passo 2.4 do SPaM) ; a seção 6.2 apresenta um novo procedimento, constituído de um conjunto de passos e heurísticas, para construir um modelo inicial de projeto OO baseado na Análise Semântica (utilizada no passo 2.3 do SpaM); e na seção 6.3 é proposta uma arquitetura para interfaces de sistemas baseada na Análise Semântica e de Normas, que possibilita sua adaptação a mudanças na organização através de alterações na especificação de normas (utilizada no passo 2.2 do SPaM).

6.1 O Uso de Modelos de SO na Modelagem de Banco de Dados

No nível de dados, para fazer a transição entre o modelo de ontologia e o primeiro modelo da estrutura de dados (passo 2.4), é proposta utilização dos procedimentos apresentados em Liu (2000, p. 119-179). Esta alternativa foi adotada por entendermos

que a transição entre os modelos da SO para a modelagem de banco de dados está suficientemente detalhada em outros trabalhos. Desta maneira, foi possível dar um enfoque maior para a camada de negócios e de interface que são pontos ainda pouco explorados na literatura anterior em SO.

Para a camada de dados ressaltamos duas opções:

1. Utilizar uma Base de Dados Semântica Temporal (BDST) que é uma tecnologia para o gerenciamento de dados com propriedades semânticas e temporais trazidas da análise semântica. Uma BDST possui características importantes para o projeto de sistemas, onde a responsabilidade legal está envolvida e o tempo em que as transações de negócio ocorrem é um fator crítico. Estas bases de dados podem ser especificadas por meio de uma linguagem chamada LEGOL, que possui a mesma estrutura básica de uma norma. Para implementar a base de dados basta traduzir a Análise Semântica e de Normas para esta linguagem e utilizar um software chamado Normbase. Mais detalhes sobre esta alternativa podem ser encontrados em Liu (2000, p. 119-179).
2. Uma outra alternativa é utilizar um banco de dados relacional para as fases 3 e 4 do SPaM. Conforme destacado em Liu (2000, p.159) é possível construir uma base de dados relacional informada pelo modelo de ontologia. Esta alternativa não preserva todos os aspectos da análise semântica, mas tem a vantagem de possibilitar a utilização direta de bases de dados comerciais que estão integradas aos ambientes e às linguagens de programação mais utilizadas comercialmente. Exemplos de sistemas construídos utilizando esta alternativa podem ser encontrados em Thönissen (1990) e Ades (1989).

No estudo de caso apresentado nesta tese (capítulo 7) foi utilizada a segunda solução de maneira simplificada para construir um banco de dados inicial a partir do diagrama de ontologia. Apesar de simplificada esta alternativa foi suficiente para obter resultados satisfatórios no estudo de caso proposto. Os passos adotados para construir a base de dados no estudo de caso foram:

1. Criar tabelas para *affordances* e agentes modelados no diagrama de ontologia (alguns *affordances* e agentes que não possuam determinantes podem virar atributos de outras tabelas);
2. Cada determinante transforma-se em um atributo;
3. As dependências ontológicas e demais relações do diagrama de ontologia sugerem relacionamentos entre as tabelas. Estes relacionamentos devem conter "integridade referencial" para garantir as dependências existenciais, por exemplo: se tivermos uma dependência ontológica entre os *affordances* "organização" e "departamento" que sugerem a construção das tabelas "organização" e "departamento", então a dependência ontológica irá sugerir a necessidade de modelar uma relação com "integridade referencial" pois não pode existir um "departamento" sem a existência de uma "organização" relacionada a ele;
4. Adicionar novos atributos e relacionamentos de acordo com a necessidade de dados a serem armazenados e recuperados pela aplicação;
5. Aplicar normalização.

6.2 A Semiótica Organizacional informando o Projeto Orientado a Objetos

De acordo com Xie e outros (2003, p. 89) "Compreender o negócio por ele próprio é fundamental para que o desenvolvimento de qualquer software seja bem sucedido. Para analistas de sistemas e designers um desafio em seus trabalhos é a comunicação com especialistas do domínio assim como o entendimento adequado deste, a sua interpretação, e a aplicação dos conhecimentos sobre o negócio no design e na implementação do sistema."

Durante os últimos anos, novos padrões surgiram na indústria de software. Particularmente os padrões baseados no paradigma da Orientação a Objetos (OO) tornaram-se os mais difundidos após a popularização das linguagens de programação OO. Atualmente a UML (OMG, 2003) é largamente utilizada por analistas de sistemas, designers e desenvolvedores para a modelagem OO. Conforme destacado anteriormente,

o RUP (Kruchten, 1999) também é largamente utilizado pela indústria de software. Entretanto, literatura em Semiótica Organizacional (SO) tem apontado alguns aspectos do contexto organizacional que se revelam frágeis nas abordagens de modelagem baseadas na visão objetivista quando aplicadas à modelagem da organização e do negócio (Liu, 2000; Stamper, 2000; Liu and Xie, 2003; Xie e outros, 2003).

O SPaM propõe o uso de técnicas e modelos da SO em suas duas primeiras fases: (1) *o design e revisão do contexto organizacional* e a (2) *levantamento de requisitos do sistema e design de alto nível*. Entretanto, para a terceira e quarta fases: (3) *o design detalhado do sistema* e (4) *a construção do protótipo*, é proposto o uso de técnicas, modelos e processos dependentes da tecnologia empregada para implementação. Nesta seção é proposto um procedimento para construir modelos de projeto iniciais da UML, informados pela análise semântica, para a camada de negócio (passo 2.3 do SPaM). Desta maneira, assumindo que a abordagem da SO pode contribuir com o aprimoramento da modelagem da organização e do negócio, podemos ter ambos: por um lado, a SO com uma visão diferente e de valor para a modelagem da organização e por outro lado um padrão de fato na indústria de software baseado na OO.

Nesta tese não é esperado que os diagramas de ontologia e os diagramas da UML modelem a mesma coisa, visto que eles representam conceitos diferentes. A seguir são destacadas diferenças entre o diagrama de ontologia e modelos de negócio baseados na UML:

- Enquanto o modelo de negócios do RUP descreve-o como um processo, a análise semântica não foca na visão de processo, mas na descrição dos signos na organização e na relação entre eles;
- Nós temos diferentes conceitos na OO e na análise semântica, e conseqüentemente os modelos (formados a partir destes conceitos) exibem representações diferentes da realidade.

Estas diferenças não inviabilizam a utilização dos diagramas de ontologia em conjunto com os diagramas da UML, visto que o próprio conjunto de modelos da UML é formado de diagramas que possuem visões complementares do problema (por exemplo: diagrama de casos de uso e diagrama de classes). Entretanto, existem relações entre eles

que podem ser explicitadas e um diagrama pode dar suporte à construção de outro provendo os benefícios presentes nas diferentes visões do sistema. Portanto, para utilizar o diagrama de ontologia em conjunto com os diagramas da UML é necessário construir um procedimento que torne explícitas as relações entre eles e possibilite que um diagrama suporte a construção do outro.

Como ponto de partida para elaborar este procedimento, foram utilizados os princípios básicos propostos por Liu (2000) para construir diagramas OO a partir de conceitos trabalhados pela Análise Semântica. Estes princípios são: "

1. Toda informação do modelo semântico deve ser utilizada na derivação. Qualquer alteração no modelo semântico deve ser aprovada pelo usuário;
2. Existem regras para mapeamento entre os termos da análise semântica e o projeto orientado a objeto:
    - Agentes e *affordances* que sugerem entidades → objetos
    - Determinantes → atributos
    - *affordances* que sugerem ações → comunicação entre objetos
    - papéis → atributos e restrições
    - parte-todo → objetos aninhados
    - generalização → herança
3. Normas associadas com o modelo semântico devem ser satisfeitas como condições e restrições dinâmicas às ações dos objetos" (Liu, 2000, p. 161)

Os passos e heurísticas, que constituem o procedimento proposto, foram gerados a partir do refinamento e generalização do processo utilizado para desenvolver o sistema Pokayoke, um sistema de CSCW para dar suporte a resolução de problemas em uma organização de manufatura. Este sistema foi desenvolvido utilizando o SPaM; desta maneira, diagramas de ontologia foram produzidos durante as duas primeiras fases e o sistema Pokayoke foi implementado utilizando uma linguagem OO (Java).

Um procedimento inicial para construir diagramas de classes a partir dos diagramas de ontologia foi produzido com base nos princípios de transformação apresentados por Liu (2000). Este procedimento foi então refinado e generalizado, para constituir uma

seqüência de passos e um grupo de regras heurísticas para construir diagramas de classes a partir dos resultados da Análise Semântica.

Na próxima seção os conceitos básicos são revisados e é definido um processo para informar o diagrama de classes da UML com os resultados da Análise Semântica. A seção seguinte exemplifica a abordagem proposta.

6.2.1 Heurísticas para construir diagramas de Classes a partir de diagramas de Ontologia

A Figura 6.1 exibe os quatro passos que são propostos para a construção da primeira versão de um diagrama de classes para ser trabalhado posteriormente durante o design detalhado, o projeto e a implementação do sistema. A Figura 6.2 apresenta um exemplo de um diagrama de ontologia, que será utilizado para explicar a abordagem proposta.

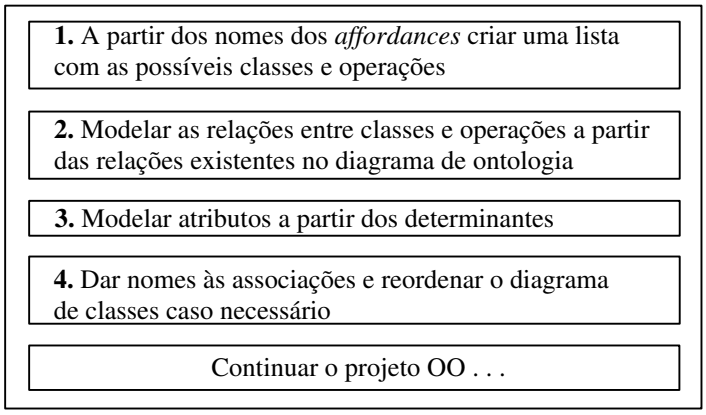


Figura 6.1 - Os passos propostos

Quando iniciamos a modelagem OO com a Análise Semântica como base, provavelmente a primeira questão a ser respondida é "Onde estão os objetos no diagrama de Ontologia?" Durante a Análise Semântica o mundo é modelado pela identificação de construções sociais chamadas *affordances* e durante a análise OO o mundo é modelado pela identificação de objetos presentes no mundo. A presença de um *affordance* no diagrama de ontologia sugere classes para serem modeladas no diagrama de classes, por exemplo: Um "departamento" pela perspectiva da Análise Semântica é um *affordance* da

sociedade e sob a perspectiva da OO ele pode ser um objeto com atributos internos e operações.

Se o *affordance* "departamento" foi representado no diagrama de ontologias, isto sugere que, pela perspectiva OO, existe um objeto no contexto e provavelmente é possível referenciar a sua classe utilizando o nome "departamento". Esta é a primeira heurística proposta (baseada no conhecido procedimento de extrair nomes de classes a partir de substantivos): nomes de *affordances* que são "substantivos" podem ser traduzidos para classes.

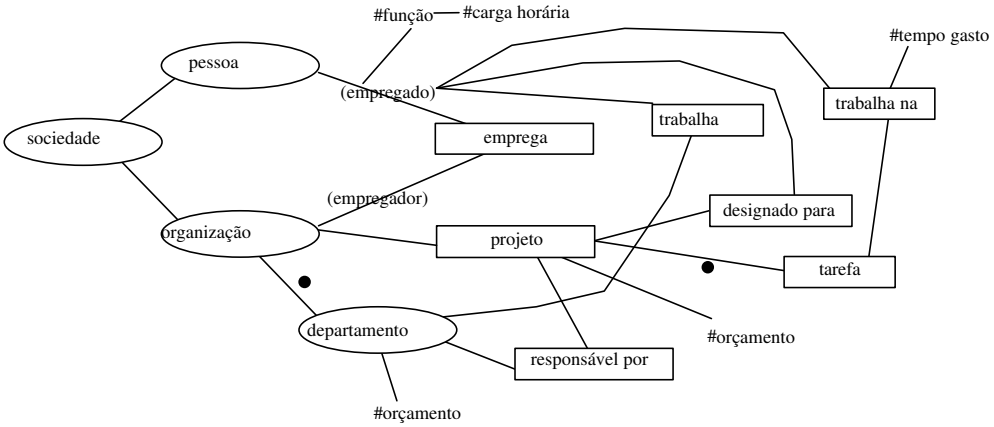


Figura 6.2 - Diagrama de Ontologia para a gerência de projetos (de Liu, 2000, pp. 79)

A Figura 6.2 exibe alguns signos que estão relacionados a objetos na perspectiva OO. Nós podemos dizer que "sociedade", "organização", "departamento", "pessoa", "empregado", "empregador" e "tarefa" são todos nomes de objetos pela perspectiva OO. Após listar alguns objetos que potencialmente serão parte do diagrama de classes inicial, podemos fazer outra pergunta "O quê os outros *affordances* sugerem?". Por exemplo, "trabalha" é um *affordance* na Figura 6.2 e ele pode ser visto como uma operação de algum objeto pela perspectiva OO. A presença de "trabalha" no diagrama de ontologia sugere que existe alguma classe com a operação "trabalha". A segunda heurística sugerida é: nomes de *affordances* que são "verbos" podem ser traduzidos para operações. Desta maneira é possível construir uma tabela com as classes e as operações potenciais. A Tabela 6.1 foi construída pela aplicação destas duas heurísticas ao diagrama da Figura 6.2; este é o resultado do primeiro passo proposto.

Tabela 6.1 – Classes e Operações Potenciais

Classes Potenciais	Operações Potenciais
sociedade	trabalha
organização	designado para
departamento	emprega
pessoa	trabalha na
empregado	responsável por
empregador	
projeto	
tarefa	

O segundo passo envolve descobrir as relações entre classes e operações para construir a primeira versão do diagrama de classes. O diagrama de ontologia pode sugerir algumas relações entre os conceitos da tabela de classes e operações. Por exemplo: existe uma dependência entre os *affordances* "sociedade" e "pessoa" na figura 6.2, e eles são classes na tabela 6.1. A dependência ontológica sugere que existe uma associação entre as classes "sociedade" e "pessoa".

"Por quê a dependência ontológica sugere este tipo de relação?" Se existe uma dependência ontológica no diagrama de ontologia, isso significa que o *affordance* dependente somente é possível durante a existência do *affordance* antecedente. Pela perspectiva OO nós podemos dizer que objetos derivados do *affordance* dependente (por exemplo "pessoa") são criados e destruídos durante a existência do objeto derivado do *affordance* antecedente (por exemplo "sociedade"). Esta interpretação sugere que o objeto "pessoa" somente existe se o objeto "sociedade" também existir. Este tipo de dependência "existencial" não pode ser diretamente representado no diagrama de classes, visto que este diagrama não especifica quando os objetos são construídos e destruídos. Para representar este tipo de dependência podemos utilizar os diagramas de comportamento da UML (esta dependência pode ser armazenada em uma lista de dependências de ciclo de vida para ser utilizada durante a construção dos diagramas de comportamento). Como uma terceira heurística nós propomos modelar uma associação entre classes, sempre que um objeto não pode existir se outro não existir.

O caso descrito no último parágrafo é somente um entre os casos possíveis de relações entre *affordances* no diagrama de ontologia que resultam em relações entre elementos da tabela com as classes e operações potenciais. No total foram identificados 10 casos diferentes. As Tabelas 6.2 até a 6.5 descrevem as regras heurísticas para serem aplicadas para cada caso e a fundamentação para cada uma delas. Estes casos estão distribuídos em quatro grupos:

- *Grupo A* (Tabela 6.2). Regras para serem aplicadas em relações entre *affordances* cujos nomes são "substantivos" no diagrama de ontologia. Por exemplo, na Figura 6.2 as relações deste grupo são: "sociedade-pessoa", "sociedade-organização", "pessoa-empregado", "organização-empregador", "organização-departamento", "organização-projeto" e "projeto-tarefa";

Tabela 6.2 - Regras do Grupo A

Regra	Descrição da Regra e Exemplo	Fundamentação
a.i	Se a relação é "parte-todo" no diagrama de ontologia então o diagrama de classes terá uma composição. Por exemplo o *affordance* "departamento" é parte da "organização" no diagrama de ontologia; então, no diagrama de classes a "organização" ira compor a classe "departamento"	O relacionamento "parte-todo" significa que um *affordance* não é só parte de seu antecedente mas também é ontologicamente dependente dele. Na Orientação a Objetos uma "agregação" representa um objeto como parte de outro, mas não especifica que a parte não pode existir sem o todo. Por isso, é proposto o uso de "composição", um tipo de especial de "agregação", que especifica que o objeto que compõe é responsável por criar e destruir as suas partes. Desta maneira o ciclo de vida da "parte" está restrito ao ciclo de vida do "todo"
a.ii	Se a relação é uma "dependência ontológica" então existirá uma associação entre as classes correspondentes aos *affordances* que fazem parte desta dependência. A dependência existencial deve ser representada nos diagramas de comportamento. Por exemplo: se o *affordance* "problema" é ontologicamente dependente do *affordance* "organização", então no diagrama de classes existirá uma associação entre "problema" e "organização", e nos diagramas de comportamento o objeto da classe "problema" é instanciado e destruído somente se existir um objeto da classe "organização"	Uma "dependência ontológica" significa que o *affordance* dependente somente é possível se existir o antecedente. Pela perspectiva OO nós podemos dizer que o objeto derivado do *affordance* dependente deveria ser criado e destruído durante a existência do objeto derivado do antecedente. Este tipo de dependência não pode ser representado no diagrama de classes, visto que ele não especifica quando os objetos são construídos e destruídos. Esta é a razão pela qual propomos o uso de diagramas de comportamento para representar isto. Nós propomos também uma associação entre as classes, pois esta dependência entre *affordances* sugere uma associação entre as classes
a.iii	Se a relação é "papel" (*role-name)* no diagrama	Um "papel" no diagrama de ontologia, significa

	de ontologia, a classe correspondente ao papel será uma subclasse da classe correspondente ao affordance antecedente e terá uma associação com a classe correspondente ao affordance da sua direita (se esta não for uma de suas operações). Por exemplo: o "papel" "empregado" na figura 6.2 é dependente de "pessoa" e "emprega", então no diagrama de classes "empregado" será uma sub-classe de "pessoa" e terá uma relação com a classe que contém a operação "emprega"	que um agente possui um papel específico. Pela perspectiva OO nós poderíamos ter um objeto que foi derivado do papel como uma especialização do objeto derivado do *affordance* antecedente; ou seja, ele é um objeto de uma subclasse da classe do "agente". Em alguns casos uma alternativa é o uso do conceito de "papel" da orientação a objetos, mas este conceito tem um significado diferente, pois ele especifica o papel de um objeto em uma associação e não um objeto que tem um papel específico
a.iv	Se a relação é uma "especialização" ela pode ser traduzida para uma relação hierárquica no diagrama de classes. Por exemplo se "pessoa física" e "pessoa jurídica" são termos específicos para "pessoa legal" então no diagrama de classes "pessoa física" e "pessoa jurídica" serão subclasses da classe "pessoa legal"	Uma "especialização" na Análise Semântica significa que nós temos um *affordance* genérico e outro especializado. Na perspectiva OO nós temos um objeto genérico e outro especializado; este conceito é modelado como uma relação hierárquica no diagrama de classes

- *Grupo B* (Tabela 6.3). Regras para serem aplicadas a relações entre *affordances* onde o nome do antecedente é um "substantivo" e o nome do dependente é um "verbo" (ou "expressão verbal"). Por exemplo na figura 6.2, nós temos: "empregado-trabalha", "departamento-trabalha", "empregado-trabalha na", "tarefa-trabalha na", "departamento-responsável por", "projeto-responsável por", "projeto-designado para" e "empregado-designado para";

Tabela 6.3 - Regras do Grupo B

Regra	Descrição da Regra e Exemplo	Fundamentação
b.i	Se a relação é uma "dependência ontológica" nós temos duas opções: (1) a operação correspondente ao *affordance* dependente será parte (como operação) da definição correspondente a um de seus antecedentes, ou (2) se ele já é parte de alguma classe, então o diagrama de classe terá uma associação entre esta classe (classe na qual o *affordance* foi modelado como operação) e a classe correspondente ao *affordance* antecedente. A escolha sobre qual classe irá conter a operação é feita de acordo com os princípios da OO (por exemplo: "andar" pode ser uma operação da classe "pessoa"). A dependência existencial será representada nos diagramas de comportamento. Por exemplo: o *affordance*	Uma "dependência ontológica" significa que o *affordance* dependente somente é possível se existir o antecedente. Se a operação (derivada do *affordance* dependente) é parte da definição da classe (derivada do *affordance* antecedente) a operação somente é possível se a classe é possível, visto que não existe operação sem classe na abordagem OO refletindo assim a dependência ontológica. Se a operação já é parte da definição de outra classe nós propomos uma associação entre as classes, porque a operação (derivada do *affordance* dependente) somente é possível se a classe (derivada do *affordance* antecedente) for possível. O diagrama de classe não especifica quando os objetos são construídos e destruídos, para isso propomos o

	"andar" é dependente de "pessoa" e "superfície" no diagrama de ontologia; no diagrama de classes "andar" será uma operação de "pessoa" e existirá uma associação entre "pessoa" e "superfície". Os diagramas de comportamento devem refletir o fato de que o método "andar" não é possível sem um objeto da classe "superfície"	uso de diagramas de comportamento
b.ii	Se a relação é uma "especialização" no diagrama de ontologia, então o diagrama de classe terá a operação derivada do affordance especializado (no diagrama de ontologia) como parte da definição da classe derivada do affordance genérico. Por exemplo: o *affordance* genérico "atitude" e os *affordances* especializados "desejo", "querer" e "gostar", no diagrama de classe "atitude" terá as operações "desejo", "querer" e "gostar"	Uma "especialização" significa que nós temos um *affordance* genérico e outro especializado. Não existe relações hierárquicas entre classes e operações em um diagrama de classes. Por isso é proposto que a operação deveria ser parte da definição da classe derivada do *affordance* genérico visto que a "especialização" no diagrama de ontologia sugere que os affordances têm algum comportamento em comum

- *Grupo C* (Tabela 6.4). Regras para serem aplicadas em relações entre *affordances* onde o nome do antecedente é um "verbo" (ou "expressão verbal") e o nome do dependente é um "substantivo". Um exemplo deste grupo pode ser (não existe este caso no exemplo da Figura 6.2): o *affordance* "erro" (de transmissão) é ontologicamente dependente do verbo "transmitir", ou seja erros de transmissão existem enquanto algum agente transmite algo;

Tabela 6.4 - Regras do Grupo C

Regra	Descrição da Regra e Exemplo	Fundamentação
c.i	Se a relação é uma "dependência ontológica" o diagrama de classes terá uma associação entre a classe relacionada ao dependente e a classe onde a operação (derivada do antecedente) foi especificada. A dependência existencial será representada nos diagramas de comportamento. Por exemplo, o *affordance* "erro" é ontologicamente dependente do *affordance* "transmite"; então, o diagrama de classes terá uma associação entre a classe que possui o método "transmite" (por exemplo a classe "servidor de email") e a classe "erro"	Uma "dependência ontológica" significa que o *affordance* dependente somente é possível se existir o antecedente. Como o primeiro é um "verbo" e o segundo é um "substantivo" então temos uma operação e uma classe. O fato de um objeto existir somente se uma certa operação também existir pode ser interpretado como o objeto deve ser criado e destruído durante a execução de uma certa operação (possivelmente através de invocação em cadeia). Este fato pode ser representado pelos diagramas de comportamento da UML e as invocações entre classes sugerem associações no diagrama de classes

- *Grupo D* (Tabela 5). Regras para serem aplicadas em relações entre *affordances* cujos nomes são "verbos" no diagrama de ontologia. Um exemplo deste grupo pode ser (não existe este caso no exemplo da Figura 6.2): o *affordance* "tropeçar" é ontologicamente dependente do *affordance* "andar".

Tabela 6.5 - Regras do Grupo D

Regra	Descrição da Regra e Exemplo	Fundamentação
d.i	Se a relação é "parte-todo" nós temos duas opções: (1) se as operações são parte da mesma classe então a operação derivada do antecedente deverá invocar (talvez não diretamente) a outra operação e (2) se elas são parte da definição de classes diferentes então existirá uma associação entre as duas classes, e o antecedente irá invocar (talvez não diretamente) a outra operação. Por exemplo: uma parte do *affordance* "registrar" é o *affordance* "preencher"; desta maneira, o diagrama de classe poderia ter "registrar" e "preencher" como parte da mesma classe (por exempo a classe pessoa) e os diagramas de comportamento iriam refletir que a execução de "preencher" somente é possível durante a execução de "registrar"	A relação "parte-todo" significa na análise semântica que um *affordance* é parte de outro *affordance*. Na perspectiva OO isto pode significar que uma operação é parte de outra operação. Então propomos que uma operação poderia invocar (talvez não diretamente) a outra operação. Os diagramas de comportamento devem refletir esta relação. Se elas forem parte da definição de classes diferentes então deverá existir uma associação uma vez que invocações entre classes nos diagramas de comportamento sugerem associações a serem modeladas no diagrama de classes
d.ii	Se a relação é uma "dependência ontológica" nós temos duas opções (como na regra d.i): (1) se as operações são parte da mesma classe então a operação derivada do antecedente deverá invocar (talvez não diretamente) a outra operação e (2) se elas são parte da definição de classes diferentes então existirá uma associação entre as duas classes, e o antecedente irá invocar (talvez não diretamente) a outra operação. Por exemplo o *affordance* "tropeçar" é ontologicamente dependente do *affordance* "andar"; no diagrama de classes, se eles são parte da definição da mesma classe (por exemplo "pessoa") então os diagramas deverão refletir que a execução de "tropeçar" ocorre durante a execução de "andar"	Uma "dependência ontológica" significa que o *affordance* dependente somente é possível se existir o antecedente. Como os dois são "verbos" pela perspectiva OO, uma operação existe somente se outra operação também existir. Desta maneira, uma operação deveria estar executando se e somente se outra também estivesse. Por esta razão propomos que a operação relacionada ao antecedente deveria invocar (talvez não diretamente) a outra operação. Estes aspectos são representados pelos diagramas de comportamento na UML. Se elas forem parte da definição de classes diferentes, então deverá existir uma associação uma vez que invocações entre classes nos diagramas de comportamento sugerem associações a serem modeladas no diagrama de classes
d.iii	Se a relação é uma "especialização" no diagrama de ontologia, nós temos duas opções: (1) se os *affordances* genérico e especializado foram traduzidos para operações da mesma classe, a especializada deveria utilizar a genérica durante sua execução e (2) se os *affordances* genérico e especializado foram traduzidos para operações de classes diferentes, isso pode sugerir uma relação hierárquica entre	Uma "especialização" significa que nós temos um *affordance* genérico e outro especializado. Como ambos são "verbos" na perspectiva OO nós supostamente teríamos uma operação "genérica" e outra "especializada". No diagrama de classes não existe relação hierárquica direta entre operações. A operação mais especializada poderia utilizar a genérica para executar algum comportamento genérico. Se eles estão em

	as classes. Por exemplo os *affordances* "andar" e "correr" são uma especialização do *affordance* "locomover", se eles foram traduzidos para a mesma classe (por exemplo "pessoa") as operações "andar" e "correr" poderiam utilizar a operação "locomover"	classes diferentes, isto poderia sugerir que as classes têm pelo menos algum tipo de comportamento comum para estas operações, sugerindo assim que deveríamos indagar se pode ou não existir uma relação hierárquica entre as classes

6.2.2 Exemplificando a Abordagem Proposta

Para o exemplo da Figura 6.2, nós poderíamos aplicar as seguintes regras para relacionar os elementos da Tabela 6.1: a regra (a.i) para as relações entre "organização-departamento" e "projeto-tarefa", a regra (a.ii) para as relações entre "sociedade-pessoa", "sociedade-organização" e "organização-projeto"; a regra (a.iii) para as relações entre "pessoa-empregado" e "organização-empregador", e a regra (b.i) para as relações "empregado-trabalha", "departamento-trabalha", "empregado-trabalha na", "tarefa-trabalha na", "departamento-responsável por", "projeto-responsável por", "projeto-designado para" e "trabalhador-designado para".

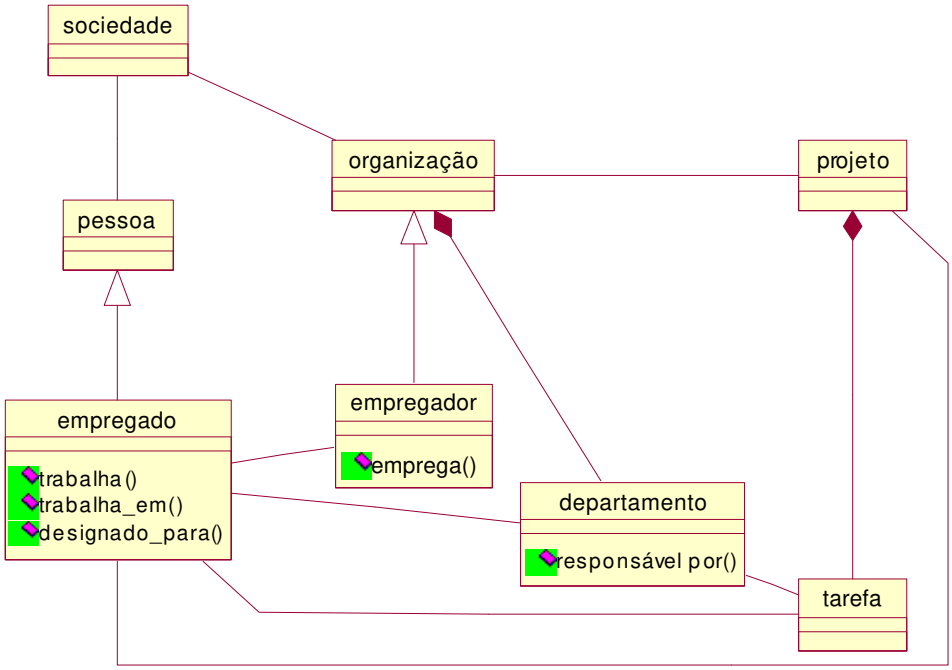


Figura 6.3 - Diagrama Intermediário

A Figura 6.3 apresenta o resultado do passo 2; ela contém um diagrama de classes gerado a partir da aplicação das regras propostas. O passo 3 (veja figura 6.1) compreende a tradução dos determinantes para atributos e classes. Por exemplo, na Figura 6.2, os determinantes "função" e "carga horária" foram traduzidas para atributos de empregado (veja Figura 6.4). Determinantes de *affordances* que são verbos podem ser traduzidos para variáveis de operações. Alguns determinantes também podem sugerir a necessidade de modelar novas classes.

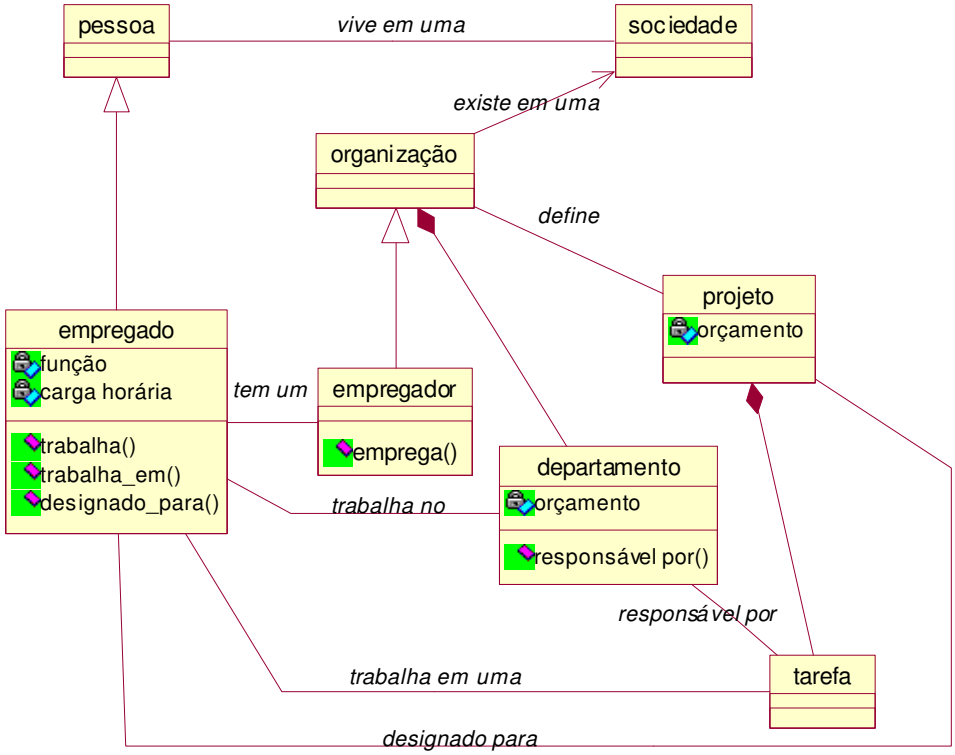


Figura 6.4 - Um diagrama de classes produzido

No passo 4 (veja Figura 6.1) nós propomos refinar o diagrama de classes, atribuir nomes para as associações e alterar a posição das classes no diagrama (ou representar esta mudança de ordem pelas direções das associações). Por exemplo, na Figura 6.3 a associação entre a "organização" e a "sociedade" poderia receber o nome "existe em uma" e a ordem deveria ser alterada para facilitar a leitura em uma perspectiva OO. A escolha dos nomes das associações depende da interpretação do contexto de acordo com a perspectiva OO; por exemplo, é possível dar o nome "tem" para a associação entre

“sociedade” e “organização”. Algumas associações podem ter os mesmos nomes das operações (por exemplo “trabalha”, “trabalha na” e “atribuída” na figura 6.4); entretanto esta redundância pode ser eliminada posteriormente durante a terceira e quarta fase do SPaM.

6.3 O Uso de Modelos de SO no Design de Interfaces

A Semiótica Organizacional pode contribuir de diferentes formas no design de interfaces humano-computador. Pesquisadores de SO têm explorado várias alternativas para construir interfaces utilizando como base teorias e técnicas da SO (Andersen, 2001; Baranauskas e outros, 2002b; Connolly e Phillips, 2002; Barr e outros, 2003; Sjöström e Goldkuhl, 2003). Sobre o enfoque mais específico de como a Análise Semântica pode contribuir para o projeto de interfaces, Simoni e Baranaukas (2003) apontam como alguns elementos do diagrama de ontologia podem sugerir a estrutura de uma interface: “

- a relação ontológica indica o que pode estar disponível ou indisponível (habilitado ou desabilitado) para um agente específico ou *affordance*;
- A relação todo-parte pode informar sobre os elementos de seleção em menus, botões etc, assim como indicar o domínio de *login* (a quem está permitido ver o quê através da interface);
- a relação específico-genérico informa a escolha de elementos para menus e também auxilia a especificar os determinantes que necessitam estar disponíveis na interface para cada elemento específico;
- os papéis podem indicar privilégios e servem como filtros para operações. Eles também informam sobre determinantes que devem estar disponíveis na interface;
- *affordances* indicam as operações disponíveis para os agentes e suas dependências de outros agentes;
- determinantes podem ser campos a serem preenchidos ou a serem exibidos pela interface.”

Nas próximas seções um outro aspecto ligado a SO e interfaces é explorado; nelas é apresentado como a Análise Semântica e de Normas contribui para o design de interfaces que podem ser apresentadas a mudanças organizacionais, diminuindo os esforço necessário para a manutenção do sistema. Nestas seções é proposta uma arquitetura e um ambiente composto de ferramentas que possibilitam modificações na interface através de alterações na especificação de normas. Esta arquitetura e o ambiente proposto podem ser utilizados para delinear os passos 3.1 e 4.1 do SPaM (descritos no capítulo 5).

6.3.1 O Design de Interfaces para Mudanças Organizacionais

Mesmo que um sistema seja modelado como parte integrante do contexto da organização isto pode não garantir uma boa adequação deste sistema em longo prazo, uma vez que as organizações estão em constante evolução. Portanto, argumentamos que o sistema de informação técnico deve ser modificado no decorrer do tempo para mantê-lo de acordo com os aspectos presentes nos sistemas formais e informais da organização.

Alterações no sistema de informação técnico são freqüentemente associadas com altos custos; a manutenção de um sistema computacional pode superar várias vezes o valor gasto no seu desenvolvimento inicial (Boehm, 1975; Manderson and Layzell, 1999). Conseqüentemente, para que um software evolua com a organização é necessário um mecanismo que torne isto viável. Uma possível solução é o uso de linguagens de alto nível (quarta geração), que possibilite fazer modificações no software, sem a necessidade de alterações diretas no código. Com base nisto é proposta a construção de interfaces de CSCW que possuam a capacidade de se acomodar a modificações nas organizações.

A literatura em CSCW tem defendido que o contexto organizacional deve ser compreendido com o propósito de se construir sistemas apropriados a eles; entretanto, as organizações estão em desenvolvimento contínuo mesmo durante o design do sistema. Além disto, o mercado competitivo direciona as organizações para o aprimoramento contínuo de suas práticas de trabalho. Estas modificações normalmente têm efeitos sociais e o sistema de CSCW deveria ser adaptado ao novo contexto organizacional. Nas próximas seções é proposta uma arquitetura e um método baseado na Semiótica

Organizacional, mais precisamente a análise Semântica e de Normas (Stamper e outros, 1988) para dar suporte a mudanças durante o design do sistema e seu uso.

O design de sistemas com todas as características configuráveis/flexíveis não é viável na prática. Com o objetivo de lidar com a complexidade deste problema no contexto organizacional é proposta uma arquitetura organizada em duas partes: uma parte estática que contém elementos menos prováveis de serem afetados por mudanças na organização e uma parte dinâmica com elementos mais prováveis de serem afetados.

De acordo com a nomenclatura proposta por Sommerville (2000), a parte estática da arquitetura proposta refere-se aos "requisitos permanentes" e a parte dinâmica aos "requisitos voláteis". "Requisitos Permanentes" são estáveis e são derivados do núcleo das atividades da organização; por exemplo um hospital sempre terá médicos, enfermeiros, etc. Requisitos voláteis são aqueles que se alteram durante o desenvolvimento ou quando o sistema está em uso. Em um hospital, por exemplo, os requisitos derivados da política de saúde.

A Análise Semântica, conforme destacado por Liu (2000), é um método que produz um modelo estável da organização; por isso ela é utilizada para construir a primeira parte do sistema na arquitetura proposta. O modelo semântico é traduzido em um modelo de implementação orientado a objetos. Os conceitos utilizados para definir o modelo de classes são resultantes da aplicação da *teoria dos affordances* (Gibson, 1968), que é utilizada na definição dos diagramas de ontologia durante a análise semântica. Conforme apresentado anteriormente, nesta teoria os objetos não são considerados como coisas que estão para elas mesmas com um conjunto de propriedades e comportamento, mas eles são compreendidos como repertório de comportamento dos agentes, disponíveis pela ação dos agentes e seu ambiente.

A Análise de Normas é um método utilizado para discutir aspectos dos níveis pragmático e social da organização. Na arquitetura proposta as normas estão relacionadas a vários aspectos da interface do sistema de CSCW; quando uma norma é alterada a interface do sistema também é alterada automaticamente, refletindo assim alterações nos níveis pragmático e social da organização mesmo durante o uso do sistema. Estas trocas são realizadas por um especialista em interface em conjunto com os usuários através de um Ambiente de Configuração de Interfaces (ICE – Interface Configuration

Environment) sem a necessidade de manutenção direta no código do sistema (Bonacin e outros, 2003b; Bonacin e outros, 2004).

6.3.2 Um Ambiente Dirigido a Normas para a Configuração e Manutenção de Interfaces

A Figura 6.5 ilustra os principais elementos da arquitetura proposta. Ela é organizada em duas partes: a Interface do Usuário Final e o Ambiente de Configuração de Interfaces (ICE - Interface Configuration Environment). Os "Usuários Finais" são as pessoas que interagem diretamente com o sistema em seu ambiente de trabalho; o "Especialista em Interface" é a pessoa que configura a interface do usuário final (supostamente com a participação dos usuários neste processo) através do ICE. O especialista em interfaces realiza modificações nas normas através do ICE, que configura a interface do usuário automaticamente.

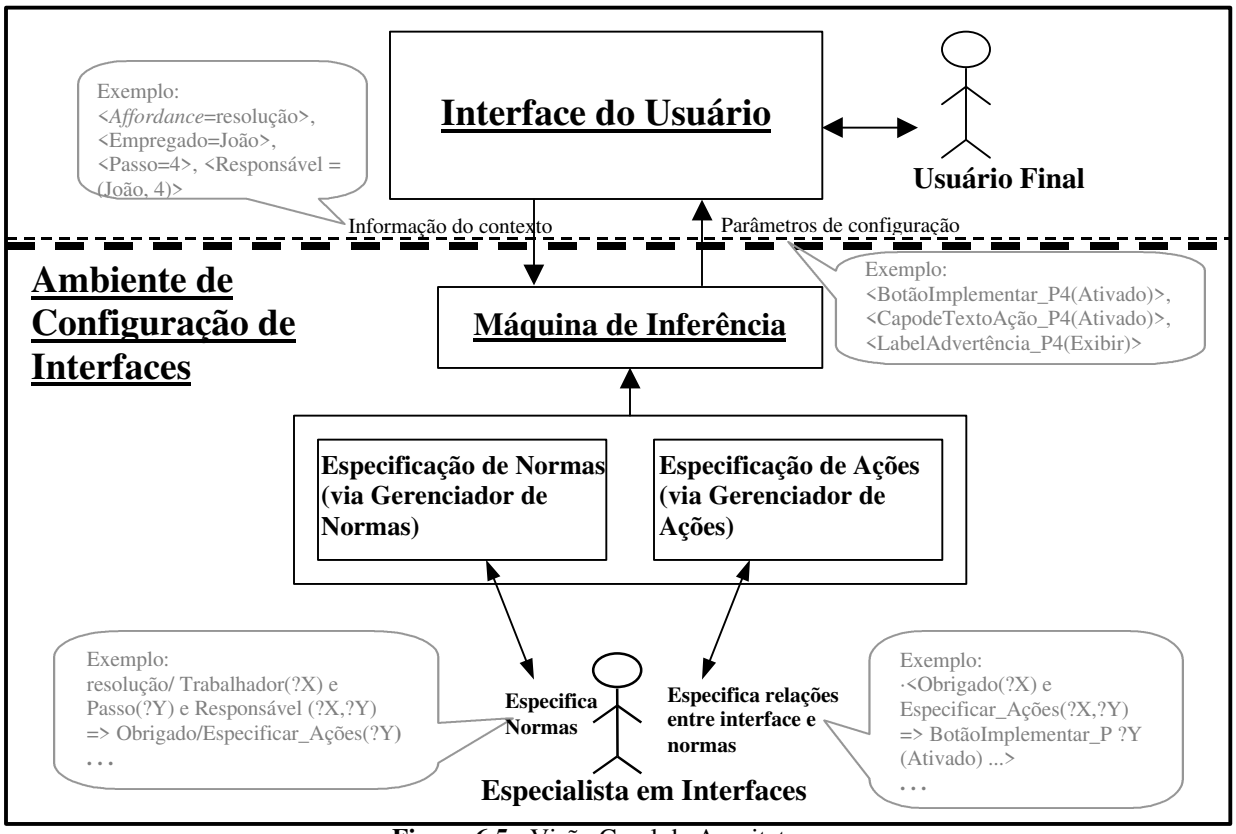


Figure 6.5 - Visão Geral da Arquitetura

A Figura 6.5 apresenta uma versão simplificada da abordagem proposta, que será detalhada nas próximas subseções. Um cenário de execução será utilizado nas próximas subseções para ilustrar a parte dinâmica da arquitetura. A seguir é apresentada a descrição em linguagem natural deste cenário:

1. Um usuário chamado "João" está envolvido em uma tarefa de resolução de problemas; ele está no quarto passo do procedimento utilizado para a resolução de problemas. A interface do usuário final passa estes dados (sobre o contexto de uso do sistema) para o ICE, conforme ilustra a seta com a identificação "informação do contexto" na Figura 6.5.
2. O ICE interpreta os dados recebidos e utilizando uma máquina de inferência conclui que "João" é obrigado a especificar as ações a serem tomadas no quarto passo da resolução de problemas. Isto está representado na Figura 6.5 pela caixa com o nome "Máquina de Inferência".
3. O ICE traduz esta "obrigação" para parâmetros de configuração da interface que definem a interface que será exibida para o usuário, através de regras que conectam as normas com os elementos de interface. Isto está também representado pela caixa com o nome "Máquina de Inferência" na Figura 6.5.
4. A interface é apresentada para o "João" de acordo com os parâmetros especificados pelo ICE, conforme ilustra a seta com o nome de "parâmetros de configuração" na Figura 6.5.

As normas e regras avaliadas durante os passos dois e três no cenário descrito são gerenciadas pelo "Especialista em Interface" utilizando duas ferramentas do ICE: o "gerenciador de normas" e o "gerenciador de ações". Isto é representado na Figura 6.5 pelas caixas com os nomes "Especificação de Normas" e "Especificação de Ações" respectivamente.

6.3.2.1 A Interface do Usuário Final

Conforme exibe a Figura 6.6, a interface do usuário final é dividida em duas partes: uma estática, que é construída com base na análise semântica e uma dinâmica, que

constrói ou configura a interface em tempo de execução, e é definida pela especificação de normas. Para construir a interface do usuário final é assumido que sistemas de CSCW não são construídos para substituir os trabalhadores mas para ajudá-los no aprimoramento de suas práticas de trabalho. Nesta perspectiva, Designers e Usuários deveriam estar engajados na modelagem participativa para aprimorar as práticas de trabalho, construindo um contexto organizacional aprimorado (Bonacin e Baranauskas, 2003a, Bonacin e outros, 2003a).

Utilizando termos da SO, usuários e designers deveriam construir "aqui-e-agora" um modelo de trabalho futuro e o sistema deveria dar suporte ao comportamento humano neste modelo. Conseqüentemente nós não temos os *affordances* futuros dos trabalhadores mas a informação a respeito do que eles pretendem *aqui-e-agora* ter no futuro através das oportunidades trazidas pela tecnologia de CSCW.

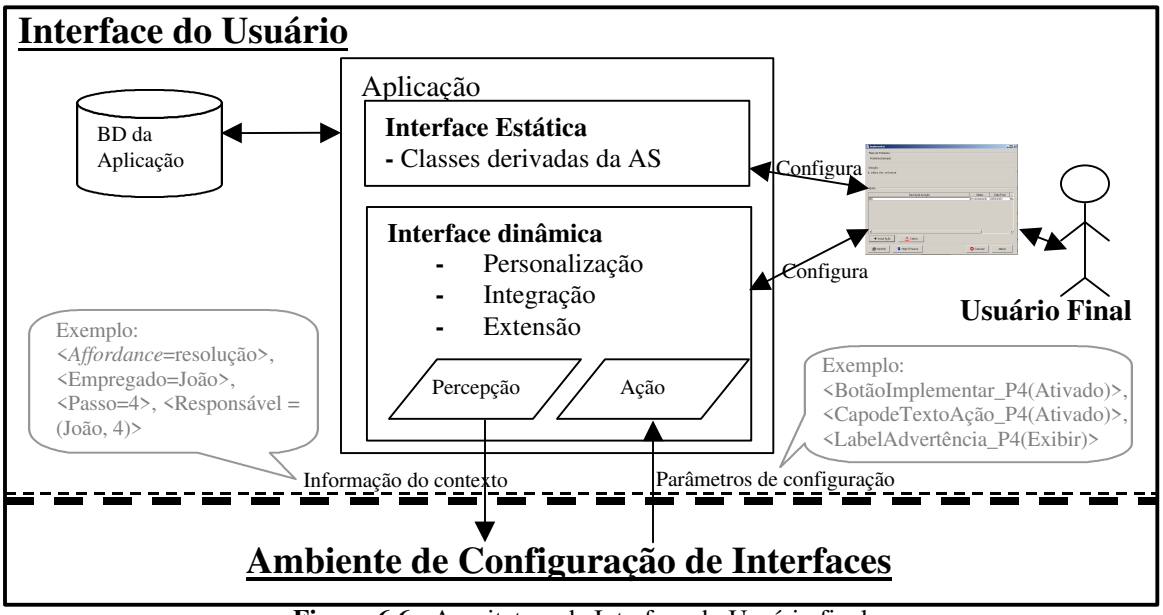


Figura 6.6 - Arquitetura da Interface do Usuário final

Durante a construção da interface do usuário, deveríamos focar nos *affordances* dos agentes humanos para gerar a parte estática da interface que contém os objetos que são modificados pela parte dinâmica. Para cada *affordance* do diagrama de ontologia que representa as práticas de trabalho para as quais pretendemos dar suporte é necessário responder as seguintes questões:

1. *Deveria ou não o sistema dar suporte a este affordance?* Não são todos os *affordances* que estão diretamente relacionados às práticas de trabalho para as quais pretendemos dar suporte com o sistema. Outros *affordances* talvez não se refiram ao comportamento dos trabalhadores;
2. *Como dar suporte ao comportamento do agente?* As interfaces são basicamente a parte perceptível do sistema para os usuários finais; então, no lugar de pensar na interface como uma coisa que está para ela mesma com um conjunto de propriedades, nós deveríamos pensar como elas deveriam ser percebidas pelos agentes que interagem com o sistema. Técnicas de DP podem ser utilizadas neste estágio. Após isto o designer poderá traduzir a sua compreensão sobre a percepção dos usuários como a tecnologia de interface utilizada.

A Figura 6.7 é parte de um diagrama de ontologia construído para ilustrar a abordagem utilizada para modelar a parte estática da interface do usuário. Este diagrama diz que o *affordance* "Enviar" é ontologicamente dependente de "Trabalhador" e "Mensagem". Tentando responder a primeira questão destacada anteriormente o *affordance* "Enviar" pode ser compreendido como parte do comportamento do agente "Trabalhador" e é possível supor que esta parte da prática de trabalho deveria ser suportada pelo sistema que está sendo construído.

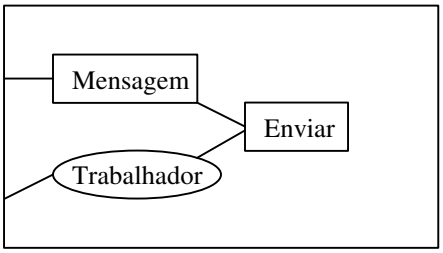


Figura 6.7 - Exemplo de parte de um Diagrama de Ontologia

Para responder a segunda questão deveríamos primeiro pensar a respeito "do significado de enviar uma mensagem" no contexto social dos trabalhadores que enviarão

a mensagem. Pensando desta maneira é possível evitar alguns erros como por exemplo utilizando um "botão" como a única maneira de enviar mensagens no sistema que é parte de um contexto social no qual o "botão" não está associado ao *affordance* "Enviar". Para clarificar este ponto é utilizada uma analogia com uma fechadura. Uma maçaneta pode ser associada com o *affordance* "Abrir" (a porta), e não somente com o *affordance* "Girar". Mas o fato de uma fechadura ser associada ao *affordance* "Abrir", não significa que é necessário colocar fechaduras em todas as portas; em alguns contextos portas automáticas (sem maçanetas) são mais eficientes. Normalmente, nos grandes supermercados as pessoas não olham para a fechadura em sua entrada, porque o contexto (lugar, formato da porta,comportamento dos outros, etc) sugere que a porta abra pela detecção de movimento. De maneira similar não é só um "botão" que pode ser associado ao *affordance* "Enviar", mas também a tecla "Enter" pode ser uma opção; o que determinará qual a opção a ser empregada é o contexto social.

Dependências ontológicas significam que certos *affordances* são somente possíveis quando outros também são possíveis. Desta maneira, algumas poderiam ser utilizadas como uma condição para exibir os elementos de interface. Por exemplo, quando um *affordance* "A" é ontologicamente depende do *affordance* "B", um elemento de interface associado ao *affordance* "A" deveria ser exibido somente durante a existência de elementos de software que representem a existência de "B". Na Figura 6.7 o *affordance* "Enviar" é dependente do *affordance* "Mensagem"; conseqüentemente a interface que corresponde ao *affordance* "Enviar" deveria ser exibida somente quando algum elemento de software (classe, dados ou interface) confirmasse a existência do *affordance* "Mensagem".

A segunda parte da interface do usuário final é dinamicamente gerada através da interpretação de normas. As mudanças que podem ocorrer nesta parte da interface são restringidas pela tecnologia computacional: é possível configurar parâmetros da interface somente se existirem opções de configuração para serem preenchidas durante o tempo de execução; é possível integrar partes da interface somente se for utilizado um mecanismo que permita a integração de componentes de software predefinidos durante o tempo de execução; e é possível estender a interface incluindo novas telas e procedimentos

somente se for utilizado um mecanismo que permita a inclusão e geração de código durante o tempo de execução.

Da mesma maneira, as modificações permitidas pelas alterações de normas podem ser limitadas pela tecnologia utilizada para implementar a interface do usuário final. A seguir são apresentados exemplos de como modificações nas normas requerem diferentes tipos de tecnologias utilizadas para implementar a interface:

- se uma modificação em uma norma restringir o acesso a algum tipo de informação, pode ser necessário somente configurar parâmetros do sistema durante o tempo de execução;
- se uma modificação em uma norma obriga ou permite que o trabalhador preencha um formulário, pode ser necessário incluir este formulário na interface do usuário através da integração de componentes durante o tempo de execução;
- se modificações em normas especificarem um novo procedimento para solucionar um dado problema, pode ser necessária a geração automática de novo código.

Conseqüentemente não é possível especificar um sistema de normas totalmente novo; desta maneira o especialista em interface deve conhecer os limites da tecnologia empregada. Uma alteração somente será viável se for associada com um mecanismo de interface; na abordagem proposta é especificada através do "Gerenciador de Ações", conforme apresentamos na próxima seção. Desta maneira, normas cuja modificação não possa ser refletida na interface podem ser diretamente programadas no código do sistema.

Conforme exibe a Figura 6.6 a parte dinâmica da interface contém dois mecanismos, que foram chamados de "percepção" e "ação". Antes de exibir uma interface para o usuário final, a aplicação obtêm os parâmetros de configuração da interface pelo ICE. Entretanto, o ICE necessita primeiro de informação a respeito do contexto de uso. Isto é necessário para saber sobre o agente que utiliza o sistema no momento, a tarefa que ele está realizando e outros dados do contexto de uso da aplicação. O mecanismo "percepção" é a parte do sistema que transmite os dados sobre o contexto

para o ICE, através de mensagens em um protocolo específico. Utilizando estes dados o ICE pode identificar o grupo de normas a serem analisadas.

A seguir é apresentado um exemplo de informação sobre o contexto com base no primeiro passo do cenário exibido anteriormente na subseção 6.3.2 (veja também a Figura 6.6):

- "<*Affordance*=resolução>" significa que os elementos de interface que serão apresentados para os usuários referem-se ao *affordance* "resolução" (este é o *affordance* mais abstrato);
- "Trabalhador = João". João é o trabalhador que está utilizando o sistema;
- "<Passo=4>" ele está no quarto passo em um processo de resolução de problemas;
- "<Responsável = (João,4)>" Diz que João é o responsável pelo quarto passo.

Após receber os dados a respeito do contexto, o ICE retorna uma lista de parâmetros que contêm os elementos de interface e suas respectivas ações. O mecanismo de "ação" recebe estes dados e os traduz em termos de modificações na interface do usuário. Estas modificações podem ser uma configuração de um elemento de interface (por exemplo: habilitar e desabilitar um botão), a integração de novos elementos (por exemplo: adição de um formulário pré-definido) ou o desenvolvimento de novo código interno.

A seguir é apresentado um exemplo de configurações de interface com base no quarto passo do cenário exibido anteriormente na subseção 6.3.2 (veja também a Figura 6.7):

- <BotãoImplementar_Passo4(Ativado)> Esta sentença significa que o botão com o nome "implementar" da tela que implementa o quarto passo deve estar ativado;
- <CampodeTextoAção_Passo4(Ativado)> Esta sentença significa que o campo de texto com o nome "Ação" da tela que implementa o quarto passo deve estar ativado;

- <LabelAdvertência_Passo4(Exibir)> Esta sentença significa que o Label com o nome “Advertência” da tela que implementa o quarto passo deve ser exibido;

Nesta subseção foi apresentada a abordagem para a construção da interface do usuário final na arquitetura proposta. A parte estática é baseada na análise semântica, onde os designers identificam os *affordances* a serem suportados pelo sistema e identificam as soluções de interface. A parte dinâmica tem dois mecanismos um de “percepção” e outro de “ação”. A interface dinâmica informa o contexto ao ICE utilizando o mecanismo de “percepção” e recebe de volta os parâmetros de configuração de interface a serem interpretados pelo mecanismo de “ação”. Na próxima subseção o ICE é apresentado.

6.3.2.2 Um Ambiente de Configuração de Interfaces

A Figura 6.8 ilustra o ICE. O primeiro bloco de cima para baixo, contém uma máquina de inferência, uma base de conhecimento e uma base de dados com as normas traduzidas para a linguagem utilizada para a representação de conhecimento.

O anexo C apresenta o NBIC (*Norm-Based Interface Configurator*), um sistema que dá suporte a implementação da arquitetura proposta. No NBIC foi utilizada a máquina de inferência e a linguagem do JESS (Java Expert System Shell). O Jess é uma *Shell* para sistemas especialistas que pode ser facilmente integrada com a linguagem Java (Friedman-Hill, 2001).

Na arquitetura proposta a máquina de inferência recebe a informação do contexto e através de “encadeamento para frente” infere as ações que devem ser tomas em relação à interface do usuário. A informação gravada na base de dados, representada pelo primeiro bloco da figura 6.8, refere-se à “especificação de normas” que é resultado da análise de normas e à “especificação de ações”, que são regras que relacionam as normas com elementos configuráveis da interface do usuário final. A especificação de normas e ações é construída por meio do uso do “Gerenciador de Normas” e do “Gerenciador de Ações” respectivamente. As Figuras 6.9 e 6.10 apresentam telas do NBIC que implementa o “Gerenciador de Normas” e o “Gerenciador de Ações”.

Através do Gerenciador de Normas (Figura 6.9), o "Especialista em Interface" especifica e mantém as normas de maneira declarativa. Cada norma é traduzida para a linguagem de representação de conhecimento (Jess no caso do NBIC) através de um módulo de tradução (segundo bloco da Figura 6.8). A máquina de inferência lê estas normas e imediatamente começa a trabalhar com a nova base de conhecimento. Conseqüentemente trocas em normas são diretamente refletidas na interface do usuário final sem a necessidade de modificações no código e novas compilações do sistema.

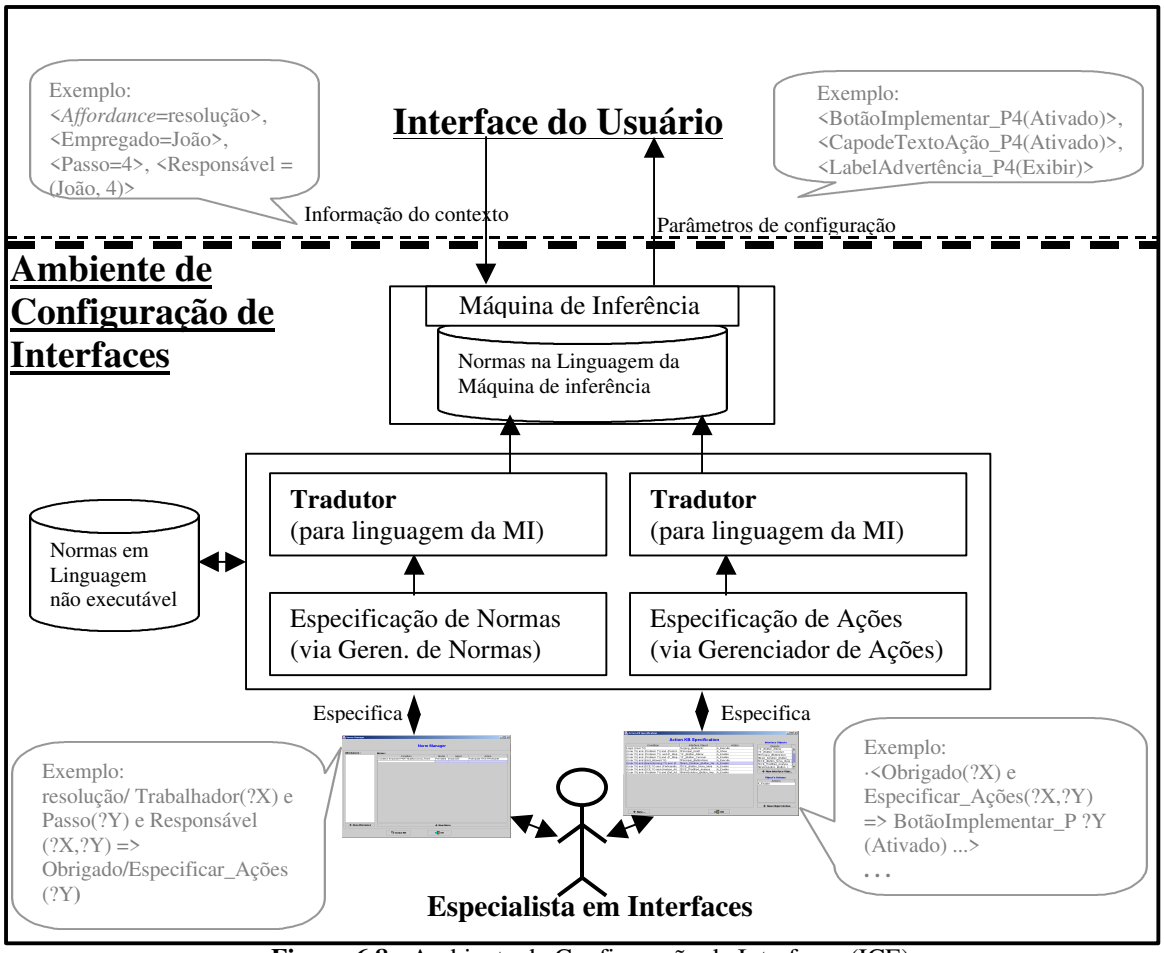


Figura 6.8 - Ambiente de Configuração de Interfaces (ICE)

No anexo C é descrita a sintaxe completa utilizada para especificar normas e as limitações do gerenciador de normas do NBIC. A seguir é apresentado um exemplo de norma que pode ser escrita no Gerenciador de Normas elaborado sobre a segunda parte do cenário exibido na subseção 6.3.2 (veja também a Figura 6.8):

- <resolução/Trabalhador(?X) e Passo(?Y) e Responsável (?X,?Y) → Obrigado/Especificar_Ações(?Y)> significa que: *no affordance resolução, sempre que o agente trabalhador for responsável por um passo então ele é obrigado a especificar as ações para corrigir o problema neste passo do processo de resolução.*

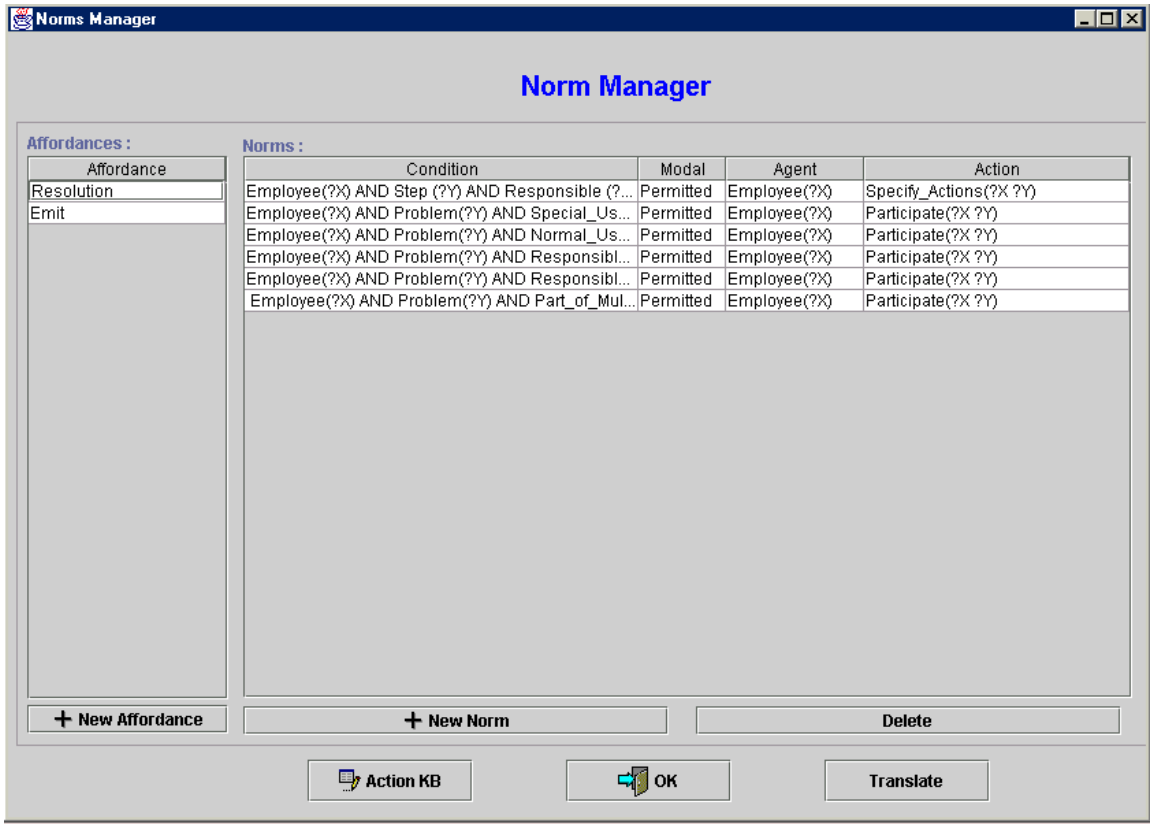


Figura 6.9 - Tela do Gerenciador de Normas do NBIC

As normas especificadas no Gerenciador de Normas são relacionadas a objetos da interface do usuário final através de regras especificadas no Gerenciador de Ações. Esta solução foi adotada para manter a independência entre as normas e a interface da aplicação; as normas estão em um nível mais abstrato e a especificação de ações relacionam estas normas com a interface do sistema.

A Figura 6.10 exibe o Gerenciador de Ações, no qual o especialista em interface especifica os objetos de interface e as ações possíveis para cada um dos objetos que estão presentes na implementação da interface. Foi atribuído o nome “objeto de interface” para

qualquer elemento de interface e o nome “ação do objeto” para as ações possíveis para um determinado objeto, como exemplo: um simples botão tem ações para exibir, habilitar, inserir um ícone, etc, ou uma tela complexa disponível pela integração de componentes de software possui vários parâmetros utilizados para configurá-los. O termo “objeto” foi utilizado porque no nível de implementação o especialista em interface pode referenciar a estes elementos de interface como objetos, uma vez que o utilizado na implementação da interface é o da orientação a objeto.

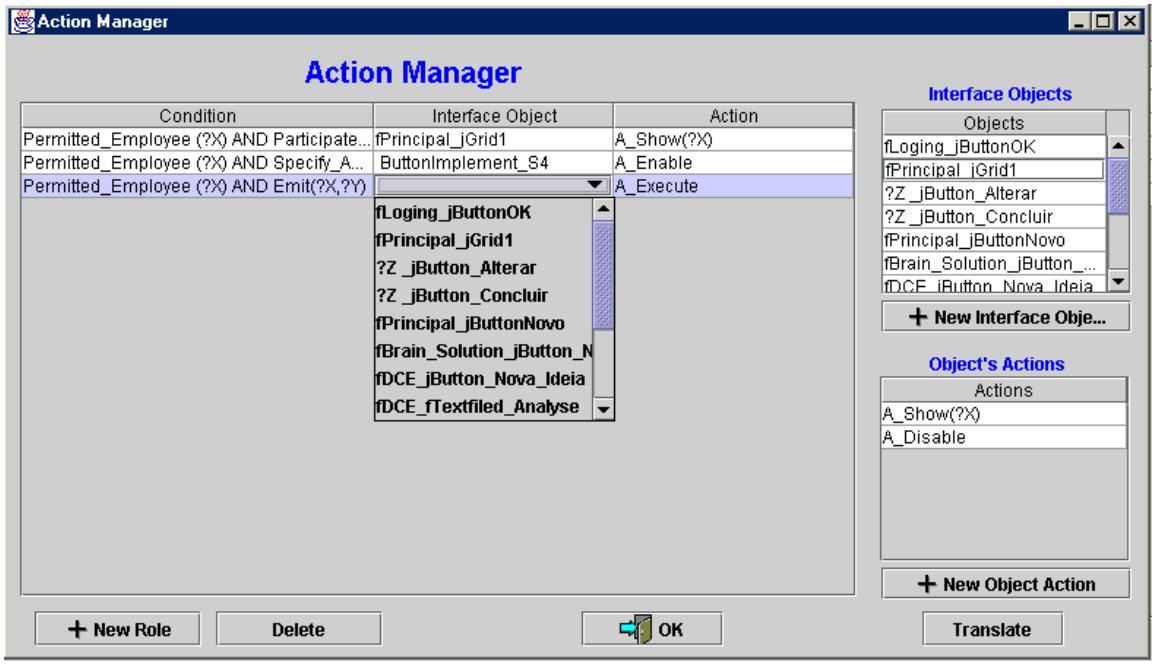


Figura 6.10 - Tela do Gerenciador de Ações do NBIC

Após especificar no gerenciador de ações os objetos e as ações associadas, o especialista em interface deverá relacionar o campo que especifica as “ações” das normas (especificado através do Gerenciador de Normas), que está em um nível mais alto de abstração, com “objetos” e “ações do objeto” da interface utilizando regras no Gerenciador de Ações. Os parâmetros relativos à condição das regras de ações (lado esquerdo da Figura 6.10) deverá ser associado a ações das normas (lado esquerdo da Figura 6.9). O anexo C destaca como fazer esta relação. Quando a máquina de inferência conclui que uma ação associada a uma norma deveria ser tomada, então regras de ações são executadas (através de encadeamento para frente), mapeando a ação resultante da

norma para uma lista de configuração de interface que especifica o que é apresentado ao usuário final.

A máquina de inferência retorna os parâmetros de configuração em uma lista de elementos que inclui o nome dos objetos e a ação correspondente. Esta lista com objetos de interface e ações é interpretada pelo mecanismo de "ação" presente na parte dinâmica da interface.

A seguir é apresentado um exemplo de regra que pode ser incluída no Gerenciador de Ações com base na terceira parte do cenário descrito na subseção 6.3.2 (veja também a Figura 6.5):

- <Obrigado(?X) e Especificar_Ações(?X,?Y) → BotãoImplementar_Passo ?Y (Ativado) ...> significa que se o agente ?X é obrigado especificar uma ação no passo ?Y então o botão com o nome "implementar" deverá estar habilitado.

Este exemplo completa o cenário de execução exibido na seção 6.3.2; traduzido nos termos utilizados para ilustrar as partes da arquitetura proposta (veja a Figura 6.11 para uma versão mais detalhada da arquitetura):

1. O Mecanismo de "percepção" da parte dinâmica da interface do usuário final informa a máquina de inferência: *<Affordance=resolução>, <Trabalhador=João>, <Passo=4>, <Responsável=(João, 4)>;*
2. A máquina de inferência através de encadeamento para frente (e unificação) usa a seguinte norma: *resolução/Trabalhador (?X) e Passo (?Y) e Reponsável (?X, ?Y) → Obrigado (?X) e Especificar_Ações(?X,?Y)* e infere *Obrigado (?X) e Especificar_Ações(?X,?Y);*
3. Novamente através de encadeamento para frente (e unificação) a máquina de inferência ativa a regra: *<Obrigado (?X) e Especificar_Ações(?X,?Y) → BotãoImplementar_Passo ?Y (Habilitado) . . .*;
4. O ICE retorna uma lista para o mecanismo de "ação" da parte dinâmica da interface do usuário: "<BotãoImplementar_Passo4(Ativado)>, <CampodeTextoAção_Passo4(Ativado)> e <LabelAdvertência_Passo4(Exibir)>".

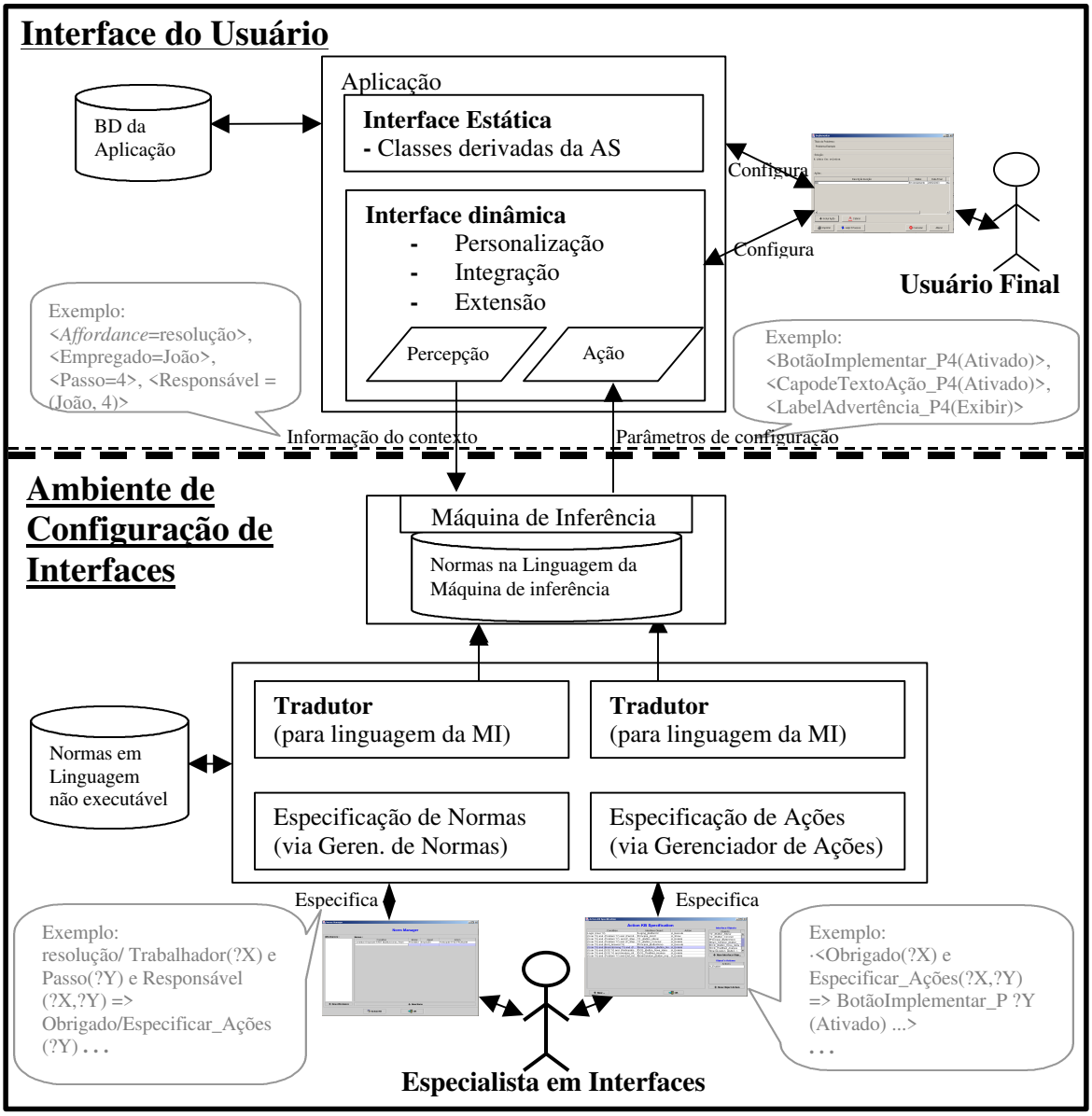


Figure 6.11 - Detalhamento do Ambiente de Configuração de Interfaces

6.4 Síntese e Considerações Finais do Capítulo

Neste capítulo foram apresentados e propostos procedimentos para construir o sistema de software a partir da Análise Semântica e de Normas em uma arquitetura de

três camadas: dados, negócio e interface. No nível de dados, para fazer a transição entre o modelo de ontologia e o primeiro modelo da estrutura de dados (passo 2.4), é proposta a utilização dos procedimentos apresentados em Liu (2000, p. 119-179). Para esta camada, ressaltamos duas opções: (1) utilizar uma Base de Dados Semântica Temporal (BDST) que é uma tecnologia para o gerenciamento de dados com propriedades semânticas e temporais trazidas da análise semântica, ou (2) utilizar um banco de dados relacional derivado da análise semântica. A primeira alternativa através de um software chamado Normbase pode ser utilizada para realizar os passos 2.4, 3.3 e 4.3 do SPaM, a segunda alternativa é utilizada no passo 2.4 e gera diagramas a serem trabalhados nos passos 3.3 e 4.3 do SPaM.

Para a camada de negócio, foi proposto um procedimento a ser utilizado durante o passo 2.3 para construir um diagrama de classes inicial informado pelo diagrama de ontologia. Este diagrama de classe servirá como base para os passos 3.2 e 4.2 do SPaM. O procedimento é formado por uma seqüência de passos e um conjunto de heurísticas, construídos pelo refinamento e generalização de uma proposta inicial aplicada no estudo de caso apresentado no próximo capítulo. Os passos propostos envolvem: (1) a partir dos nomes dos *affordances* criar uma lista com as possíveis classes e operações, (2) modelar as relações entre classes e operações a partir das relações existentes no diagrama de ontologia, (3) modelar atributos a partir dos determinantes e (4) dar nomes às associações e reordenar o diagrama de classes caso necessário.

Para a camada de interface é proposta uma arquitetura e um ambiente composto de ferramentas que possibilitam modificações na interface através de alterações na especificação de normas. Esta abordagem está baseada na divisão da interface em uma parte estática que contém elementos com menor probabilidade de serem alterados uma vez que são derivados da análise semântica e uma parte dinâmica com elementos com maior probabilidade de serem alterados uma vez que são derivados da análise de normas. Os elementos da parte dinâmica são modificados automaticamente através de mudanças na especificação das normas. Esta arquitetura e o ambiente proposto também podem ser utilizados para delinear os passos 2.2, 3.1 e 4.1 do SPaM.

No próximo capítulo é apresentado o estudo de caso que aplica a abordagem proposta. Primeiramente, o contexto da organização, seu paradigma de produção e o

problema em foco a ser solucionado são descritos. É apresentado o sistema e o processo utilizado em sua construção como um exemplo da aplicação do SPaM em um contexto real.

Capítulo 7

Aplicação do SPaM – Um Estudo de Caso

Antes de apresentar um estudo de caso é fundamental descrever em que contexto ele foi conduzido para que seja possível compreender e analisar melhor seus resultados. Neste capítulo é apresentada a descrição e a análise do contexto em que foi aplicado o SPaM. A seção 7.1 apresenta o contexto organizacional, o paradigma de produção adotado pela organização e as suas necessidades para a construção do sistema computacional proposto.

Este capítulo também detalha o estudo de caso com o objetivo de apresentar como o SPaM e demais procedimentos propostos nesta tese foram utilizados na prática. A seção 7.2.1 apresenta uma descrição do sistema Pokayoke do ponto de vista do usuário final; desta maneira, são apresentadas as suas principais funcionalidades. A descrição do sistema é importante para situar a discussão conduzida durante as outras seções deste capítulo. A seção 7.2.2 apresenta a aplicação do SPaM no desenvolvimento do sistema Pokayoke, detalhando seus ciclos de prototipação, a aplicação das técnicas de DP, como estas técnicas contribuíram para a Análise Semântica e de Normas, o uso da Conferência Semiótica no contexto do estudo de caso, e aspectos do projeto do sistema Pokayoke detalhando sua arquitetura e a construção da interface. A seção 7.2.3 discute a avaliação e o processo de significação durante o desenvolvimento do sistema Pokayoke como meio de promoção da aprendizagem mútua. Ao final, na seção 7.2.4 é discutido o sistema em

uso e são analisadas as modificações propostas de acordo com o esforço necessário para realizá-las.

7.1 Análise do Contexto

O contexto do estudo de caso apresentado nesta tese é um projeto (Nied, 1999; Nied, 2002) que foi realizado através de uma parceria entre nossa instituição e uma organização multinacional que adota o sistema de "produção enxuta" no processo de manufatura de componentes para sistemas automotivos. Nesta seção, são apresentados alguns princípios do paradigma da "produção enxuta" relevantes a esta tese e as necessidades da organização que levaram à construção do protótipo proposto.

7.1.1 O Paradigma da Produção Enxuta

Os anos 90 marcaram a década das mudanças e alterações de paradigmas nos sistemas produtivos. Organizações que insistem em manter antigos modelos não conseguem acompanhar as novas necessidades da sociedade. O mundo está encolhendo (Coleman, 1995) e a tecnologia deve ser um apoio para as organizações contra competidores nacionais e internacionais, que tendem a surgir cada vez mais intensamente. Segundo Andrew S. Grove, presidente da Intel, (em Coleman (1995, pg. 35)), os negócios estão espalhados por todo o mundo, enquanto as informações se proliferam. As competições tornam-se cada vez mais intensas e, neste contexto, uma colaboração suportada por sistemas computacionais tende a se tornar, talvez, o mais importante fator de vantagem.

Este estudo de caso usa como pano de fundo o contexto de organizações industriais dentro de uma nova filosofia de produção chamada "produção enxuta". Enquanto a "produção em massa" tem como foco a economia de escala de produção, vendas e lucro (Mazzone, 1993), a filosofia da "produção enxuta", que teve início em um trabalho pioneiro de Deming (Deming, 1992) durante os anos 40, visa a qualidade dos produtos, satisfazendo, da melhor maneira possível, a crescente e diversificada clientela, fruto do mundo cada vez mais globalizado. Um conjunto de novas técnicas e estratégias é

utilizado para aprimorar a qualidade e a produtividade, modificando o foco do produto por ele próprio para a qualidade no processo de manufatura.

Esta nova filosofia busca não só suprir as necessidades de adaptações a novas tecnologias, mas também melhorias na qualidade de vida do trabalhador. Além de mudanças nas estratégias de produção de uma empresa, a valorização dos fatores humanos da organização é fundamental. No sistema de produção em massa o poder de decisão estava concentrado no topo da hierarquia e trabalhadores da linha de produção seguiam rotinas pré-definidas. Em oposição à produção em massa onde os funcionários são especialistas, estando somente restritos às suas funções particulares, no ambiente da "produção enxuta" os funcionários devem ser flexíveis; suas habilidades devem ir além de suas funções particulares. Eles devem possuir senso crítico e um bom desempenho no trabalho em grupo, sendo, assim, considerados elementos inteligentes necessários para aprimorar a qualidade do processo de produção.

O incentivo à participação do trabalhador nas mudanças organizacionais e na definição de novas práticas de trabalho na "produção enxuta" é um fator fundamental no desenvolvimento deste estudo de caso, uma vez que este princípio facilita a aplicação das técnicas de DP.

Entre as principais características do sistema de "produção enxuta" está a distribuição do poder de decisão entre os trabalhadores, o incentivo ao trabalho em time e uma regra mais dinâmica para os trabalhadores de chão de fábrica. Juntas, estas idéias sugerem que pessoas de níveis organizacionais diferentes, incluindo os trabalhadores que estão próximos aos problemas, deveriam criar soluções para problemas de maneira colaborativa. Considerando que o trabalho na organização deveria ser mais colaborativo e os operadores mais dinâmicos e multifuncionais no trabalho em grupo (Womack, 1990), artefatos que suportam a colaboração e o trabalho cooperativo estão alinhados com a filosofia da "produção enxuta".

7.1.2 Necessidades Organizacionais e Descrição dos Cinco Passos

Rotineiramente, as organizações enfrentam inúmeros problemas que exigem discussão e colaboração entre indivíduos de um grupo para resolvê-los. Estes problemas, sejam administrativos, gerenciais ou de produção, necessitam que conhecimentos especializados, simulação de resultados, geração de idéias, sugestões e críticas sejam inseridas no tratamento do problema, tornando importante a integração e coordenação de esforços para tomadas de decisão dentro da organização. O sistema resultante do estudo de caso apresentado nesta tese visa aprimorar um procedimento para resolução de problemas chamado "Cinco Passos", por meio do apoio à cooperação.

Os "Cinco Passos" representam um método disciplinado para lidar com problemas de rotina da produção. Toda vez que uma inconformidade é identificada, uma ação deve ser tomada para corrigi-la e evitar uma nova ocorrência. Sempre que uma situação potencial de inconformidade é indicada, uma ação Poka Yoke (*error proofing*) deve ser tomada; isto é, deve-se buscar uma ação que impossibilite a ocorrência do problema. Esta é a razão pela qual o sistema proposto chama-se Pokayoke.

Na fábrica em que trabalhamos os "cinco passos" para a resolução de problemas são:

1. *Descrição da Não Conformidade.* É uma descrição detalhada do problema. Ela deve conter dados relevantes para a caracterização do problema como por exemplo:
    - Requisitos definidos pelo Cliente, normas e expectativas;
    - Descrição do artefato do problema;
    - Foto, caso disponível;
    - Os sintomas do problema;
    - Quantificar o problema - como, quanto, freqüência;
    - Definição de perda - termos de custo, termos de satisfação;
    - Informação sobre datas;
    - Definição - O quê, Onde, Quando, Quem, Por quê;
    - Contato com o Cliente definido (telefone, email, etc).
2. *Ação Imediata.* Neste passo, um plano de contenção é elaborado; nele devemos considerar alguns aspectos como por exemplo:
    - Identificação das pessoas e áreas chaves envolvidas;

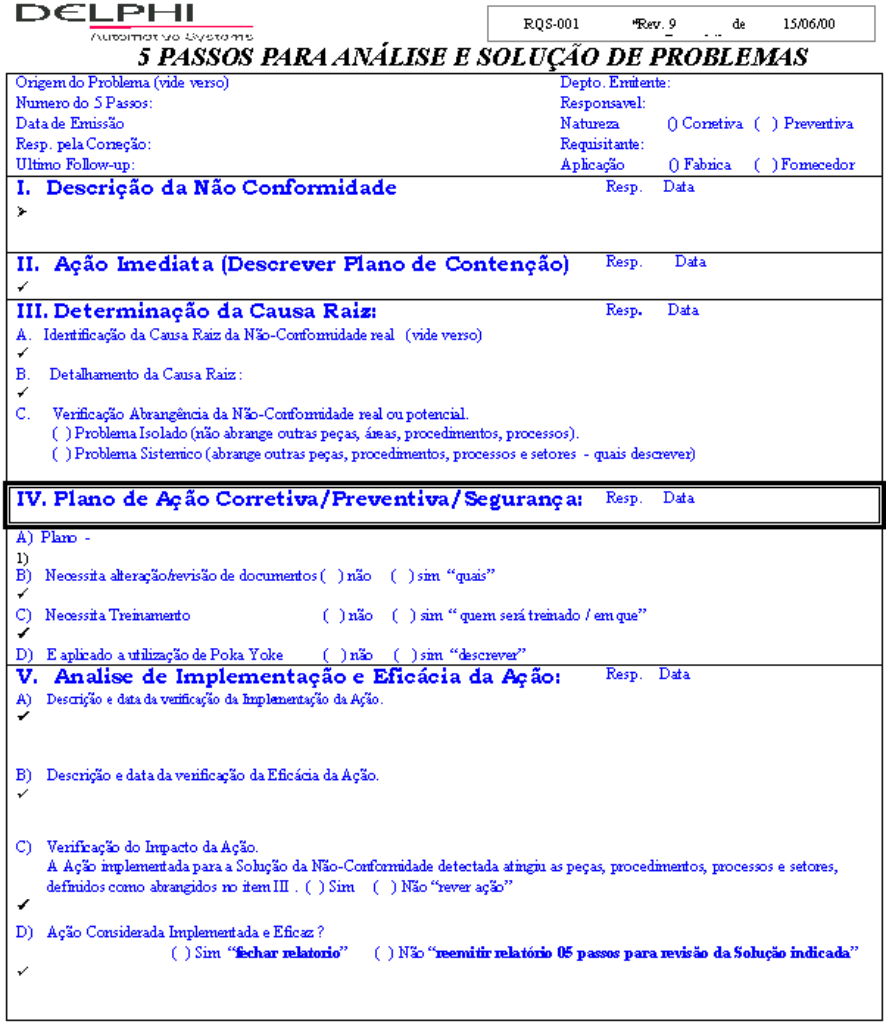
- Definição da situação;
- Metodologia para condução e coleta de informações;
- *Brainstorming*;
- Desenvolvimento de soluções possíveis, priorização;
- Ações seguras de contenção completas e mensuradas;
- Formação de um "time de ações" para realizar as ações imediatas;
- Verificação das ações de contenção;
- Produtos com os problemas detectados não podem ser enviados para o cliente sem autorização do mesmo.

3. *Determinação da Causa Raiz.* O problema é analisado com o intuito de se encontrar a causa raiz. Alguns aspectos que devem ser levados em consideração:
    - Não devem ser realizadas conclusões infundadas;
    - Quando aplicável, preencher um Diagrama de Causa e Efeito (diagrama de *Ishikawa*);
    - Quando necessário, aplicar a técnica conhecida como *5-porquês;*
    - Identificar todas as possíveis causas através da análise do problema;
    - Isolar e verificar a Causa Raiz, analisando várias vezes o problema;
    - Monitorar a evolução da solução estratégica definida.
4. *Plano de Ação Corretiva/Preventiva/Segurança.* Um plano de implementação de ações que assegure que uma nova ocorrência do mesmo problema não irá acontecer deve ser definido. Alguns aspectos que devem ser levados em consideração:
    - Devem ser aplicadas modificações permanentes no processo?
    - Quais processos devem ser modificados?
    - O time para implementação da ação requisitada está definido?
    - É requerido algum treinamento? Os recursos foram analisados?
    - Estão registrados dados e informações complementares?
    - Quais controles devem ser analisados para julgar a ação? Eles estão implementados/realizados?

- Notificar o Cliente do início de fornecimento com a modificação implementada (quando aplicável);
- Assegurar que somente serão enviados produtos após a Liberação do Cliente.

5. *Análise de Implantação e Eficácia da Ação.* Os resultados obtidos são analisados com o intuito de garantir que o problema não ocorra novamente. Alguns aspectos que devem ser levados em consideração:
    - Comparar resultados obtidos com objetivos planejados;
    - Assegurar que a Causa Raiz foi eliminada de forma Irreversível;
    - Discriminar método utilizado para analisar a efetivação da Ação;
    - Ações devem ser analisadas quanto as suas performances durante as auditorias de processo/sistema;
    - Verificar documentos que foram utilizados para *follow-up*;
    - Procurar oportunidades para melhorias contínuas;
    - Discutir *follow-up* com o cliente (quando aplicável);
    - Analisar em times as ações consideradas efetivas.

A figura 7.1 apresenta a versão do relatório de "Cinco Passos" anterior ao Pokayoke. O responsável por um passo deveria preencher o que ocorreu no passo utilizando, para isto, um editor de texto. A comunicação entre os responsáveis ocorria basicamente através da ferramenta de email existente no IBM Lotus notes.



DELPHI
Automotive Systems

RQS-001 *Rev. 9 de 15/06/00

5 PASSOS PARA ANÁLISE E SOLUÇÃO DE PROBLEMAS

Origem do Problema (vide verso) | Depto. Emitente:
Numero do 5 Passos: | Responsavel:
Data de Emissão | Natureza () Corretiva ( ) Preventiva
Resp. pela Correção: | Requisitante:
Ultimo Follow-up: | Aplicação () Fabrica ( ) Fornecedor

I. Descrição da Não Conformidade Resp. Data

II. Ação Imediata (Descrever Plano de Contenção) Resp. Data

III. Determinação da Causa Raiz: Resp. Data

A. Identificação da Causa Raiz da Não-Conformidade real (vide verso)

B. Detalhamento da Causa Raiz:

C. Verificação Abrangência da Não-Conformidade real ou potencial.
( ) Problema Isolado (não abrange outras peças, áreas, procedimentos, processos).
( ) Problema Sistemico (abrange outras peças, procedimentos, processos e setores - quais descrever)

IV. Plano de Ação Corretiva/Preventiva/Segurança: Resp. Data

A) Plano -
1)
B) Necessita alteração/revisão de documentos ( ) não ( ) sim "quais"
C) Necessita Treinamento ( ) não ( ) sim " quem será treinado / em que"
D) E aplicado a utilização de Poka Yoke ( ) não ( ) sim "descrever"

V. Analise de Implementação e Eficácia da Ação: Resp. Data

A) Descrição e data da verificação da Implementação da Ação.

B) Descrição e data da verificação da Eficácia da Ação.

C) Verificação do Impacto da Ação.
A Ação implementada para a Solução da Não-Conformidade detectada atingiu as peças, procedimentos, processos e setores, definidos como abrangidos no item III . ( ) Sim ( ) Não "rever ação"

D) Ação Considerada Implementada e Eficaz ?
( ) Sim **"fechar relatorio"** ( ) Não **"reemitir relatório 05 passos para revisão da Solução indicada"**

Figura 7.1 - Versão do relatório de "Cinco Passos" anterior ao sistema Pokayoke

Durante o procedimento de "Cinco Passos" são utilizadas ferramentas para dar suporte às atividades a serem desenvolvidas em cada um deles. Entre as ferramentas utilizadas estão: *Brainstorming, Diagrama de Causa Efeito (*conhecido também como *Digramas de Ishikawa, 6M* ou *espinha de peixe)* e *5-Porquês.* A figura 7.2 apresenta o Diagrama de Causa Efeito utilizado pela fábrica antes do desenvolvimento do sistema. Este diagrama tem como objetivo encontrar a causa raiz do problema. Ele é caracterizado por ser bem estruturado e é muito utilizado em trabalhos de melhoria de qualidade.

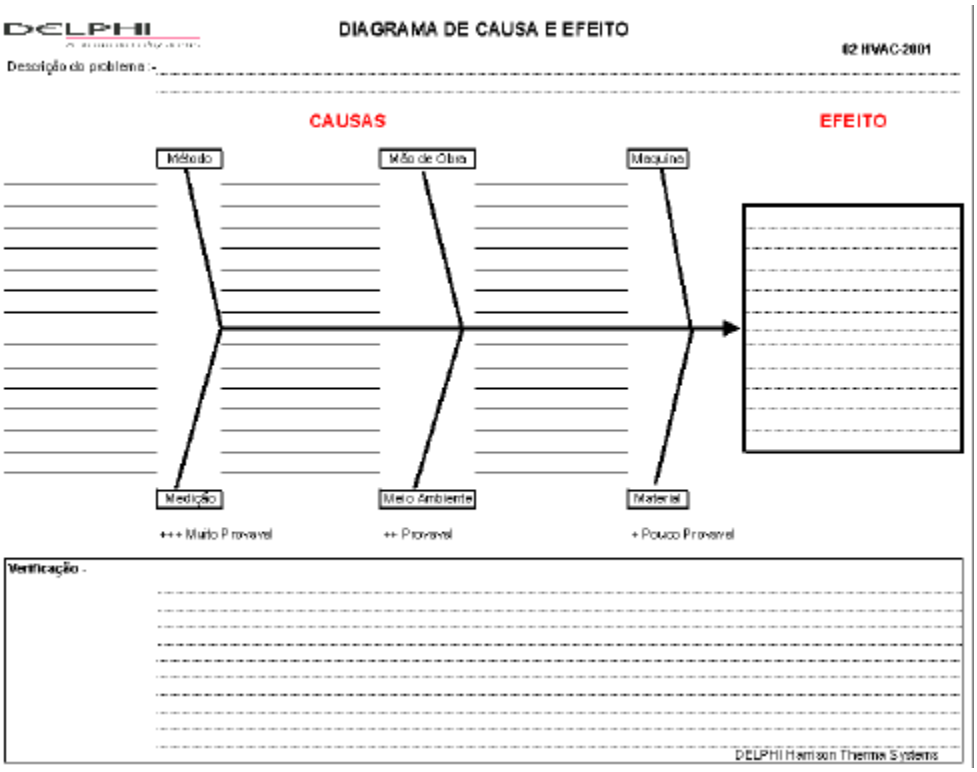


Figura 7.2 - Versão do Diagrama de Causa Efeito anterior ao sistema

Durante a aplicação da técnica de design participativo *Starting Conference*, os participantes enfatizaram a importância de utilizar ferramentas computacionais para possibilitar a resolução de problemas em todos os departamentos e níveis hierárquicos da organização de maneira cooperativa. O procedimento de "Cinco Passos" utilizado previamente na resolução de problemas estava restrito às pessoas do Departamento de Qualidade e algumas pessoas do Departamento de Engenharia. As quatro principais dificuldades enfatizadas pelos trabalhadores que levavam ao uso restrito do procedimento na fábrica eram:

1. "*O procedimento de cinco passos dificulta a participação dos funcionários devido a problemas de comunicação*";
2. "*Para utilizar algumas ferramentas em papel são necessárias reuniões longas*";
3. "*É necessário um treinamento para possibilitar que os trabalhadores utilizem as ferramentas de acordo com os procedimentos da organização.*"

4. "*Cada pessoa armazena seus formulários de cinco passos dificultando reaproveitar as soluções*"

Em relação ao primeiro item levantado, podemos destacar alguns problemas existentes (outros problemas estão descritos no Anexo A) que dificultavam a participação e a cooperação no processo:

- A comunicação, via sistema computacional, estava restrita aos emails enviados entre os responsáveis em cada passo (usualmente duas pessoas são responsáveis por todos os passos), mas um escopo muito maior de pessoas deveria participar do processo. Entre estas pessoas está o "time multifuncional" que é um grupo de trabalhadores que são escalados para participar diretamente da discussão e resolução do problema. Não existia também comunicação, via sistema, entre os responsáveis pelos passos e os responsáveis pelas ações a serem tomadas, durante o segundo e quarto passo, para solucionar o problema;
- Não existia maneira de visualizar todos os "Cinco Passos" em resolução na organização, dificultando a participação dos trabalhadores no processo, uma vez que eles não tinham informação sobre os problemas e o seu estado no processo em resolução;
- Os funcionários não eram "chamados" e "lembrados" freqüentemente a participar da resolução de maneira sistemática.

Durante o procedimento eram utilizadas ferramentas baseadas em papel para discussão das causas e soluções para o problema. Conforme o segundo item apontado estas ferramentas levavam a reuniões longas e pouco produtivas. Podemos destacar alguns problemas existentes (outros problemas estão descritos no Anexo A):

- Não existia como utilizar estas ferramentas à distância; o funcionário tinha que sair do seu local de trabalho para participar;
- Estas ferramentas não podiam ser trabalhadas de maneira assíncrona; conseqüentemente, deveriam estar disponíveis ao mesmo tempo.

Em relação ao terceiro item levantado, podemos destacar alguns problemas existentes (outros problemas estão descritos no Anexo A) que apontam para as causas da necessidade de treinamento:

- O conhecimento sobre como conduzir o procedimento de "Cinco Passos" estava extremamente concentrado no Departamento de Qualidade; portanto, muitos funcionários de outras áreas não o utilizavam;
- Os formulários não eram auto-explicativos;
- Não era possível que os usuários criassem problemas fictícios para exercitar o procedimento em conjunto.

A discussão sobre um problema antigo e a solução adotada dificilmente era reaproveitada na solução de um problema novo. Em relação ao terceiro item levantado, podemos destacar alguns problemas existentes (outros problemas estão descritos no Anexo A):

- Cada "planta" da fábrica (unidade de negócio responsável pela produção de determinados produtos) possuía um procedimento de "Cinco Passos" ligeiramente diferente, dificultando o reaproveitamento de soluções entre elas;
- Não existia um mecanismo para centralizar todos os problemas solucionados e disponibilizá-los de maneira rápida durante a solução de um outro problema.

Na seção 7.2 são apresentadas alternativas adotadas no sistema proposto para dar suporte à resolução cooperativa baseada no "Cinco Passos" e para minimizar os problemas da sua versão em papel.

7.2 Aplicação do SPaM – O Sistema Pokayoke

Nesta seção o estudo de caso é detalhado com o objetivo de apresentar como o SPaM e demais procedimentos propostos nesta tese foram utilizados na prática. Para tanto, é apresentado o sistema Pokayoke, detalhes sobre a aplicação das técnicas de DP durante o estudo de caso, resumos de relatórios de visitas à organização, como cada técnica de DP contribuiu para a construção de modelos da SO, a aplicação da conferência semiótica, aspectos do projeto do sistema Pokayoke, uma avaliação e exemplificação do processo de significação ocorrido durante o desenvolvimento e a análise de resultados obtidos pela utilização do sistema na organização.

7.2.1 Descrição do Sistema POKAYOKE

O Pokayoke é um sistema computacional construído com o objetivo de explorar na prática o SPaM. O Pokayoke dá suporte à resolução de problemas em um contexto de uma organização de manufatura que adota o paradigma de "produção enxuta". Ele inclui ferramentas para a resolução de problemas que eram utilizadas anteriormente pelo departamento de qualidade.

A Figura 7.3 é a janela principal do protótipo do sistema Pokayoke, no qual problemas são discutidos e os respectivos estágios em que estes problemas se encontram no processo de resolução são exibidos para os usuários.

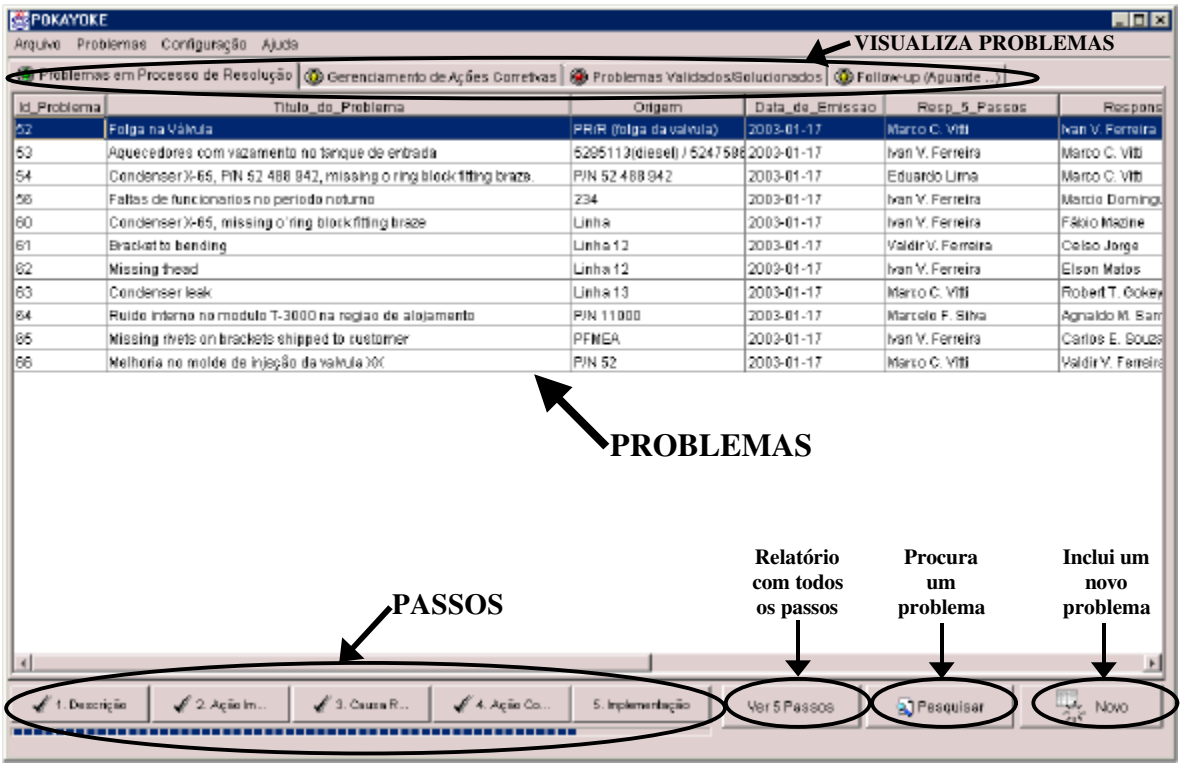


Figura 7.3 - A Janela Principal do Pokayoke

Na parte superior da figura 7.3, a identificação "visualiza problemas" destaca quatro pastas (ou painéis) com as seguintes funcionalidades: (1) visualizar os problemas em processo de resolução, ou seja, todos os problemas que ainda não tiveram o quinto passo encerrado, (2) visualizar o gerenciamento de ações corretivas, uma planilha com os problemas em resolução e os respectivos estados em que eles se encontram no processo, (3) visualizar os problemas já solucionados; por esta pasta o usuário pode acessar detalhes que levaram à solução de cada um deles, (4) visualizar planilha de "follow-up", que apresenta a descrição das ações adotadas para solucionar os problemas em processo, seus respectivos estados e os responsáveis por sua implementação. Os painéis dois e quatro são resultados de uma dissertação de mestrado finalizada (Souza, 2003). Esta dissertação visava incluir o conceito de controle de fluxo de informação no sistema através da tecnologia de *workflow*. A parte do sistema que inclui estas funcionalidades recebeu o nome de Pokayoke-flow (Souza e outros, 2003).

Quando um usuário entra no sistema pode visualizar todos problemas para os quais tenha permissão de acesso através do primeiro painel. Ele deve selecionar um problema da lista e, por meio dos botões na parte inferior da figura 7.3 (destacados com a identificação "passos") poderá visualizar os passos concluídos e participar do passo que está em resolução (caso tenha permissão para participar). A barra de progresso da parte inferior da figura 7.3 indica o estado em que o problema selecionado se encontra no processo como um todo. O usuário também pode visualizar todos os passos através de um formulário gerado pelo sistema, pesquisar problemas em processo ou solucionados e incluir novos problemas (botões da parte inferior direita da figura 7.3).

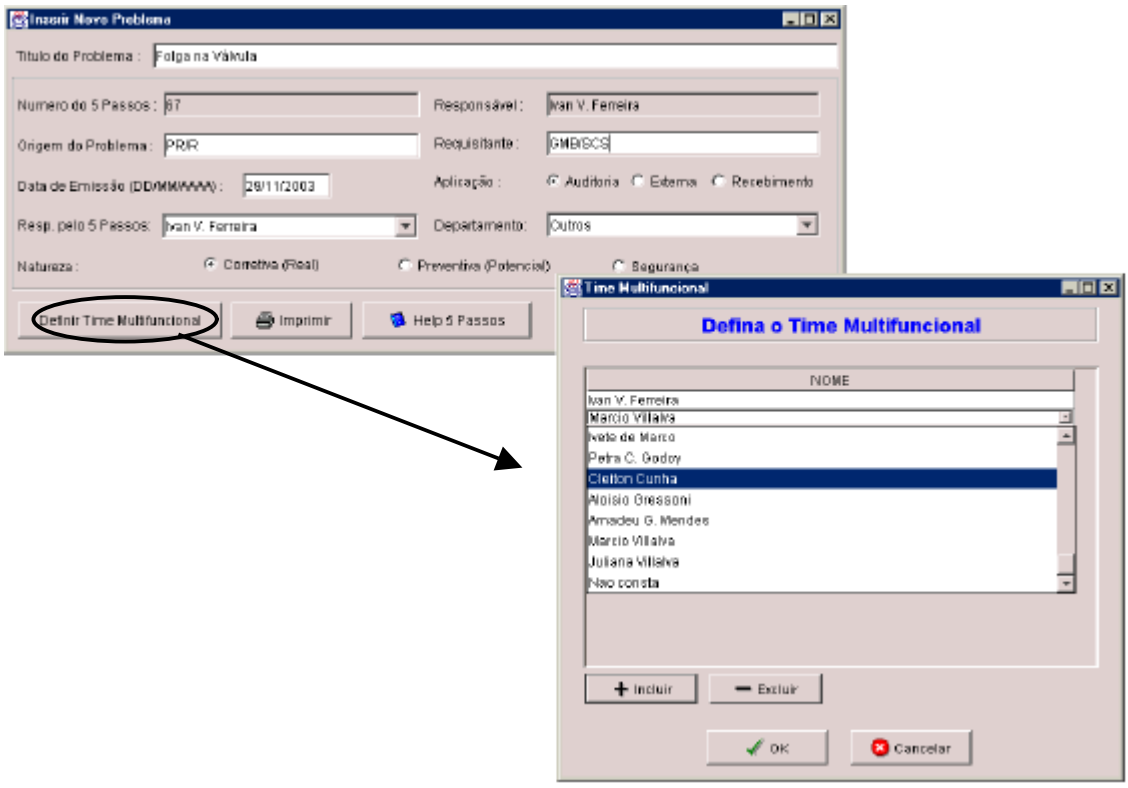


Figura 7.4 - Inserir um problema no sistema Pokayoke

As figuras 7.4 a 7.13 ilustram uma seqüência de operações que podem ser feitas no sistema Pokayoke desde a inclusão do problema até a sua solução. A figura 7.4 exibe a janela utilizada para incluir um problema no sistema Pokayoke; nela o usuário pode incluir um título para o problema (texto que aparece na lista de problemas exibido na figura 7.3), incluir a origem, incluir a data de emissão, escolher o responsável pela solução do "Cinco Passos" e identificar o requisitante do problema. De acordo com a sua procedência, um problema pode ser: (1) de *auditoria*, ou seja, problema que é detectado

dentro da fábrica, neste caso ele deverá estar relacionado ao departamento em que ocorreu, (2) *externo*, ou seja, problema que é detectado pelo cliente; neste caso o usuário deve identificar o cliente que detectou o problema e (3) de *recebimento*, ou seja problemas no material recebido de fornecedores; neste caso, o usuário deve identificar o fornecedor do material com problema. O usuário também deve definir o "time multifuncional", que são as pessoas que podem participar diretamente da resolução do problema (ver exemplo na figura 7.4).

Normas atreladas ao sistema via NBIC (veja anexo C) definem que a pessoa que emitiu o problema, ou seja, o incluiu no sistema será o responsável pelo primeiro (*Descrição da Não Conformidade*) e último passos (*Análise de Implantação e Eficácia da Ação*); esta pessoa é definida dentro do contexto como o "responsável pelo problema". O "responsável pelo cinco passos" (ver exemplo na figura 7.4) será o responsável pelo segundo (*Ação Imediata*), terceiro (*Determinação da Causa Raiz*) e quarto passos (*Plano de Ação Corretiva/Preventiva/Segurança*). Estas normas também estabelecem que os membros do "time multifuncional" devem contribuir diretamente para a solução do problema, e terão acesso as ferramentas do sistema utilizadas para solucioná-lo.

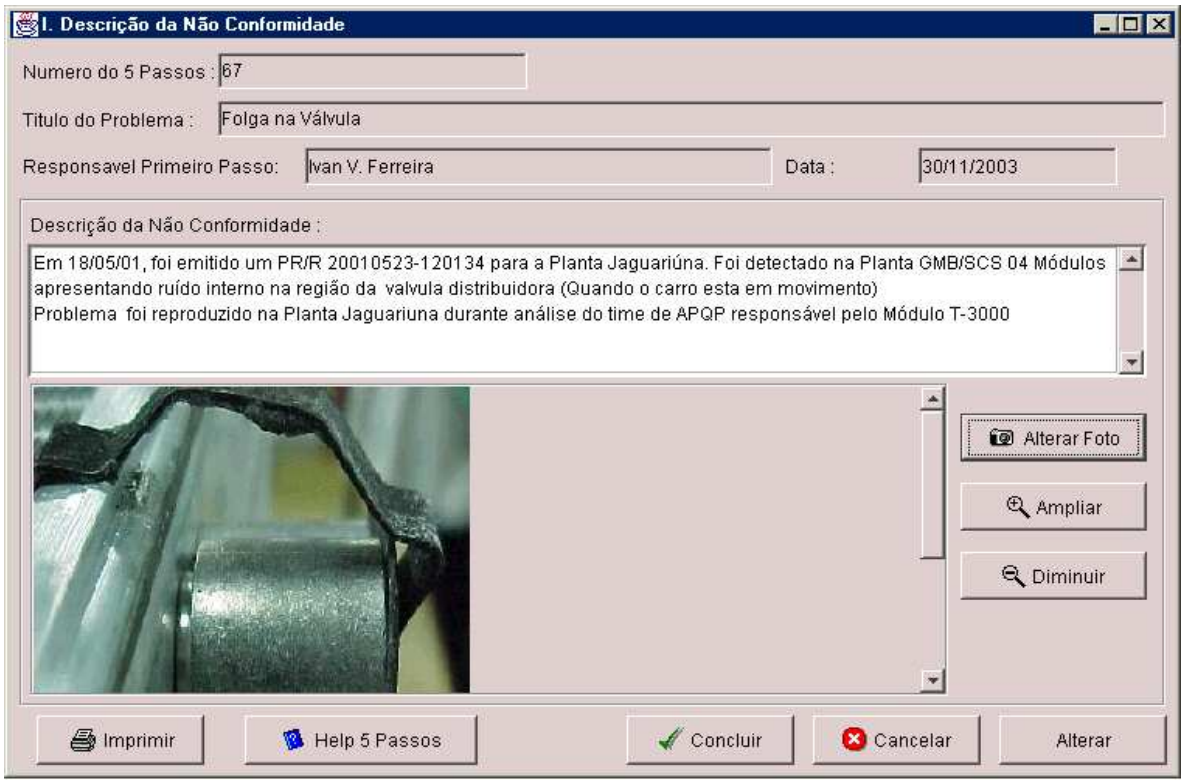


Figura 7.5 - Janela do primeiro passo do sistema Pokayoke

A figura 7.5 apresenta a janela do sistema Pokayoke para o primeiro passo no processo de resolução de problemas. Nela há um espaço para o responsável descrever a não conformidade, e um outro para anexar uma foto caso ela esteja disponível. No canto inferior direito da figura 7.5 existem três botões: o primeiro para concluir o passo, outro para cancelar as alterações realizadas e um terceiro para alterar os dados sem concluir o passo. Conforme a especificação de normas, uma vez que um passo é concluído, os participantes podem começar a trabalhar no próximo passo, e alterações no passo anterior são permitidas apenas em casos excepcionais. Esta descrição e a foto estarão automaticamente disponíveis a todos os envolvidos na resolução do problema, uma vez que isto é fundamental para a caracterização e entendimento do problema.

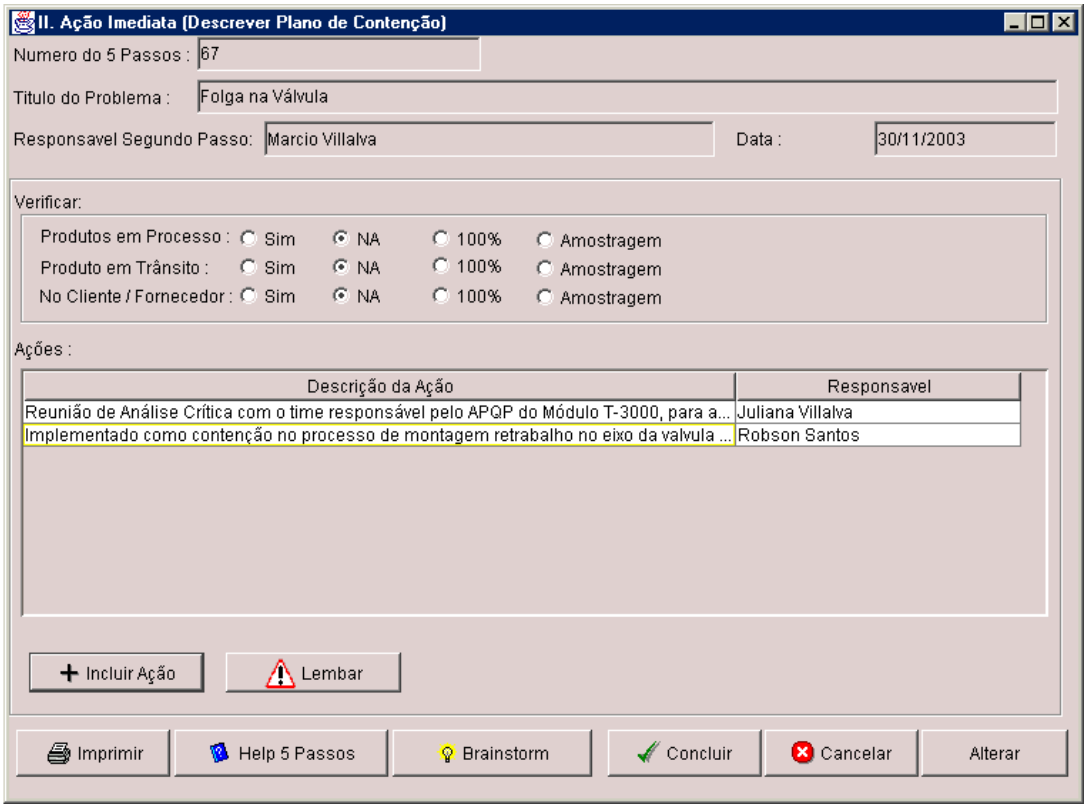


Figura 7.6 - Janela do segundo passo do sistema Pokayoke

A figura 7.6 apresenta a janela do Pokayoke para o segundo passo do processo de resolução de problemas. Nela o responsável pelo passo deve verificar os produtos afetados pelo problema, descrever as ações imediatas para o problema em foco, e atribuir responsáveis para implementar cada ação imediata. O responsável pelo segundo passo

pode avisar os responsáveis pelas ações através do botão “lembrar” que envia emails conforme proposto em Souza e outros (2003).

As ações imediatas especificadas durante este passo (pelo responsável) são resultantes de um *brainstorming* com os membros do time multifuncional. A figura 7.7 exibe a janela que dá suporte a este *Brainstorming*. Nele os membros do time multifuncional devem propor idéias para que o problema seja solucionado em curto prazo e o responsável pelo passo deve especificar ações com base nestas idéias e definir quem serão os responsáveis por interpretá-las. Normas da organização especificam que qualquer membro do time multifuncional pode contribuir com suas idéias sobre ações imediatas para o problema através da janela exibida na parte inferior direita da figura 7.7.

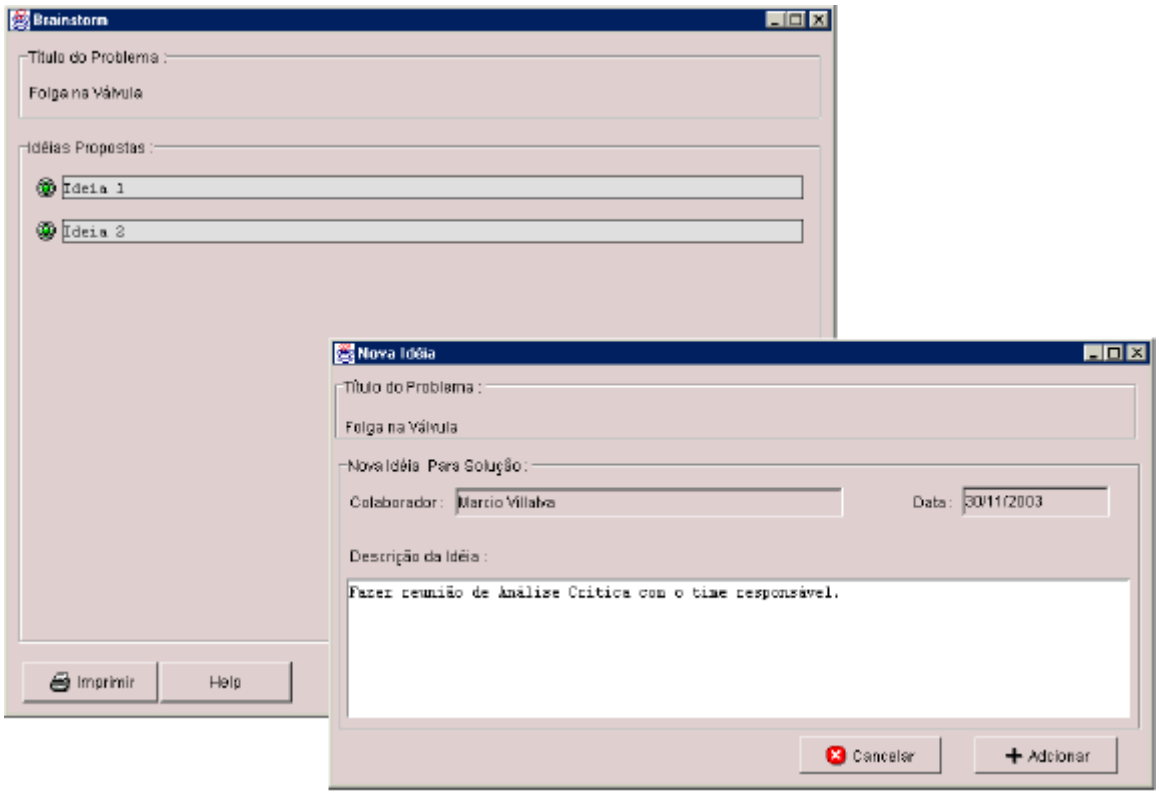


Figura 7.7 - Janela de *Brainstorming* que ocorre no segundo passo do sistema Pokayoke

A figura 7.8 apresenta o terceiro passo no processo de resolução de problemas no sistema Pokayoke. Neste passo, o responsável deve escolher a principal causa raiz através de uma lista com 8 opções: Documentação, Controle inadequado do processo, Planejamento deficiente, Falta de treinamento e/ou treinamento ineficaz, Condições de

trabalho inadequadas, Recursos inadequados (Humanos ou Materiais), Variabilidade inerente ao processo e "Outros". Após essa identificação, ele deve escrever um texto detalhando a causa raiz e, caso o problema seja sistêmico deve descrever sua abrangência em relação às peças, procedimentos, processos e setores envolvidos.

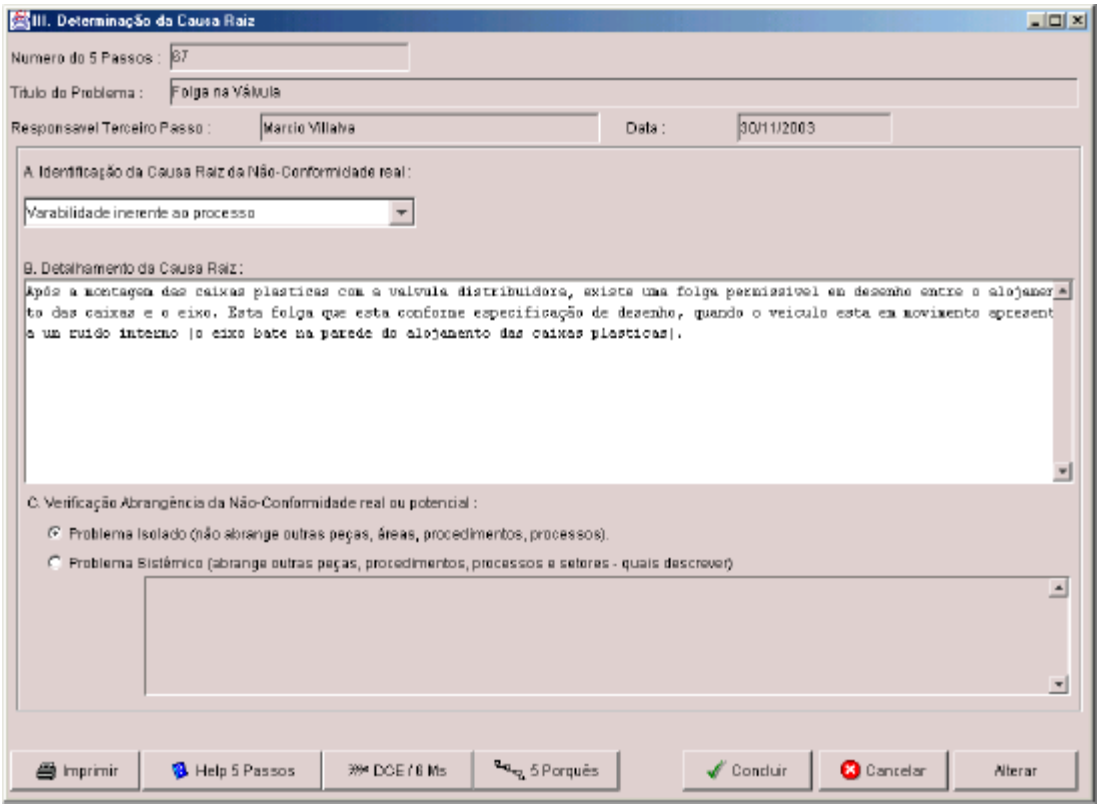


Figura 7.8 - Janela do terceiro passo do sistema Pokayoke

A identificação e o detalhamento da causa raiz é resultado da aplicação de ferramentas para análise de problemas. O sistema Pokayoke dá suporte à construção do Diagrama de Causa Efeito (DCE) e uma versão futura também dará suporte à ferramenta conhecida na produção enxuta como "5 Porquês". A figura 7.9 apresenta a janela para o Diagrama de Causa Efeito no sistema Pokayoke; nela, o responsável pelo passo deve incluir uma descrição do efeito, e os demais participantes devem enviar idéias sobre as causas possíveis que, de alguma forma, poderiam ter gerado o efeito.

No DCE idéias sobre as prováveis causas raízes devem ser incluídas em um dos seguintes itens de acordo com a sua natureza: Método, Mão-de-Obra, Máquina, Medição, Meio Ambiente e Material. Para incluir uma nova idéia os usuários utilizam a janela

exibida na parte inferior direita da figura 7.9. Nesta janela, além da idéia para a possível causa raiz, eles também devem dizer se consideram a idéia como pouco provável, provável ou muito provável de ser a verdadeira causa raiz para o problema em discussão. Normas estabelecem que qualquer membro do time multifuncional poderá contribuir com idéias para o Diagrama de Causa Efeito. Ao final, as normas estabelecem que o responsável deverá fazer a verificação do diagrama (canto inferior esquerdo da figura 7.9).

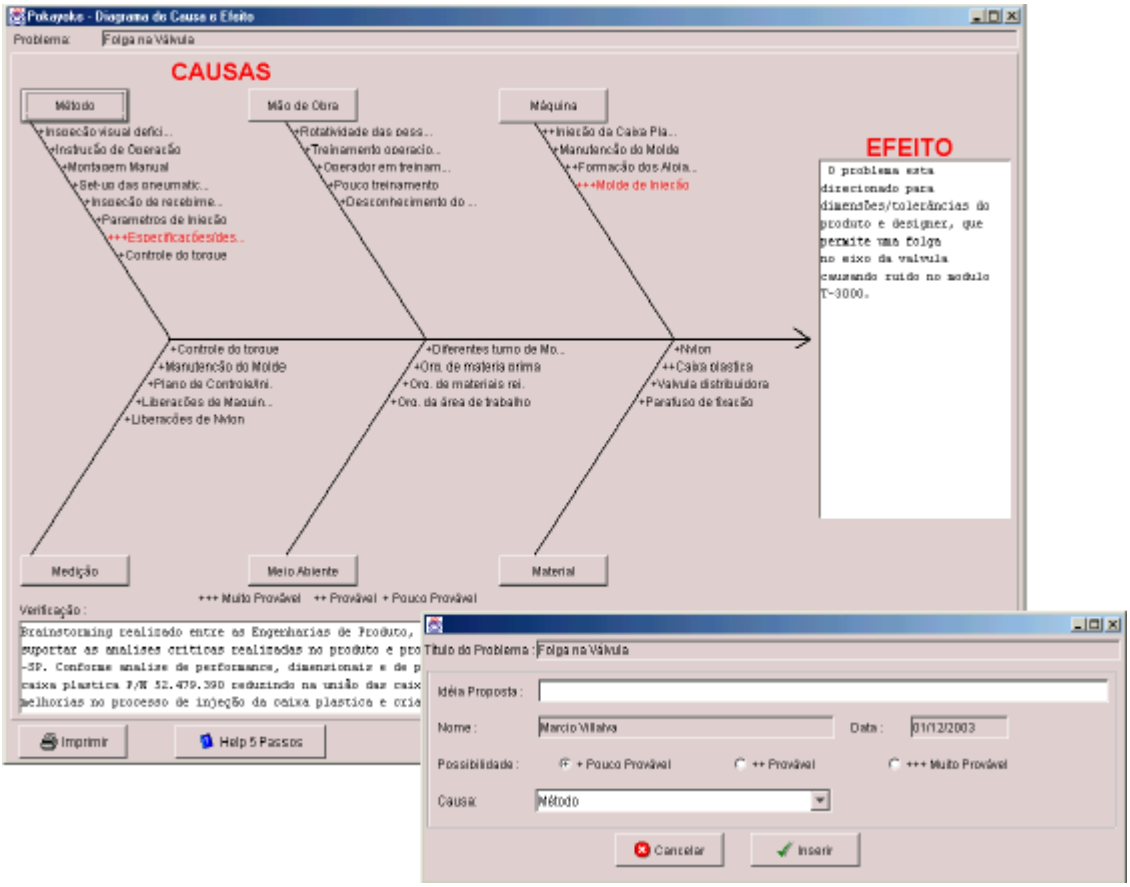


Figura 7.9 - Diagrama de Causa Efeito no sistema Pokayoke

A figura 7.10 exibe a janela do quarto passo do processo de resolução no sistema Pokayoke. Neste passo, o responsável deve descrever o plano para solucionar o problema, visando minimizar as chances de que ele ocorra novamente. O responsável deve verificar também se existe necessidade de alterar documentos e treinamento e, caso exista, ele deverá detalhar quais são estas necessidades. Sempre que possível o

responsável deverá aplicar uma solução *Poka Yoke* (uma solução que inviabilize que o problema ocorra novamente) que deverá ser descrita no passo.

O plano de ação é derivado de um *Brainstorming*, no qual os participantes apresentam idéias de soluções definitivas para o problema em foco. O responsável deve escolher a solução mais adequada entre todas as idéias propostas pelos participantes. Para cada solução ele deverá atribuir ações que levem para elas. O plano de ação descrito na janela da figura 7.10 deverá articular as ações para resultar na solução do problema por completo.

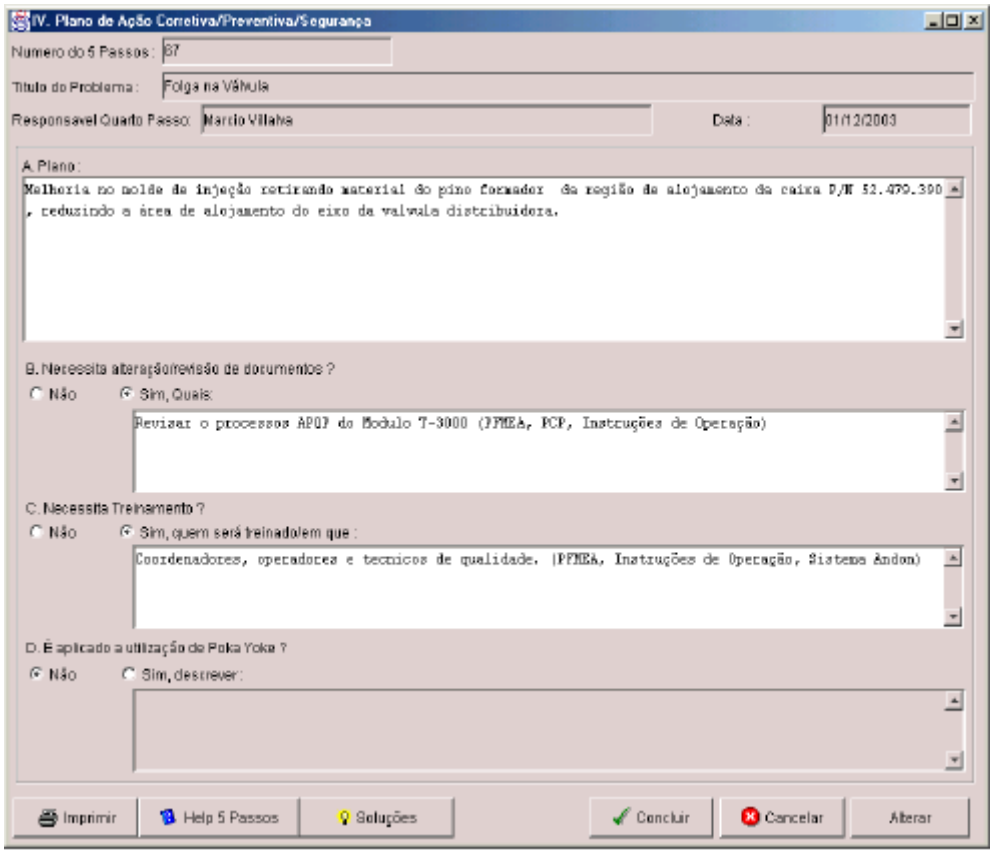


Figura 7.10 - Janela do quarto passo do sistema Pokayoke

A figura 7.11 apresenta as janelas do *Brainstorming* que ocorre durante o quarto passo; a janela no canto superior esquerdo refere-se à lista de idéias propostas pelos participantes. No lado esquerdo de cada idéia existe um espaço para ícones que identificam se a solução não contém ações (botão), contém ações que estão temporariamente suspensas (símbolo de *pause*), contém ações que estão em andamento

(chave de fenda) ou foi concluída (símbolo de verificado "✓"). A janela no canto inferior direito da figura 7.11 é utilizada pelo usuário para propor uma nova solução para o problema.

Normas especificam que os membros do time multifuncional podem propor soluções, entretanto eles não podem atribuir ações para estas soluções, apenas o responsável pelo passo tem a permissão de fazê-lo. O responsável pelo quarto passo só poderá concluí-lo se existirem ações concluídas que realmente solucionem o problema.

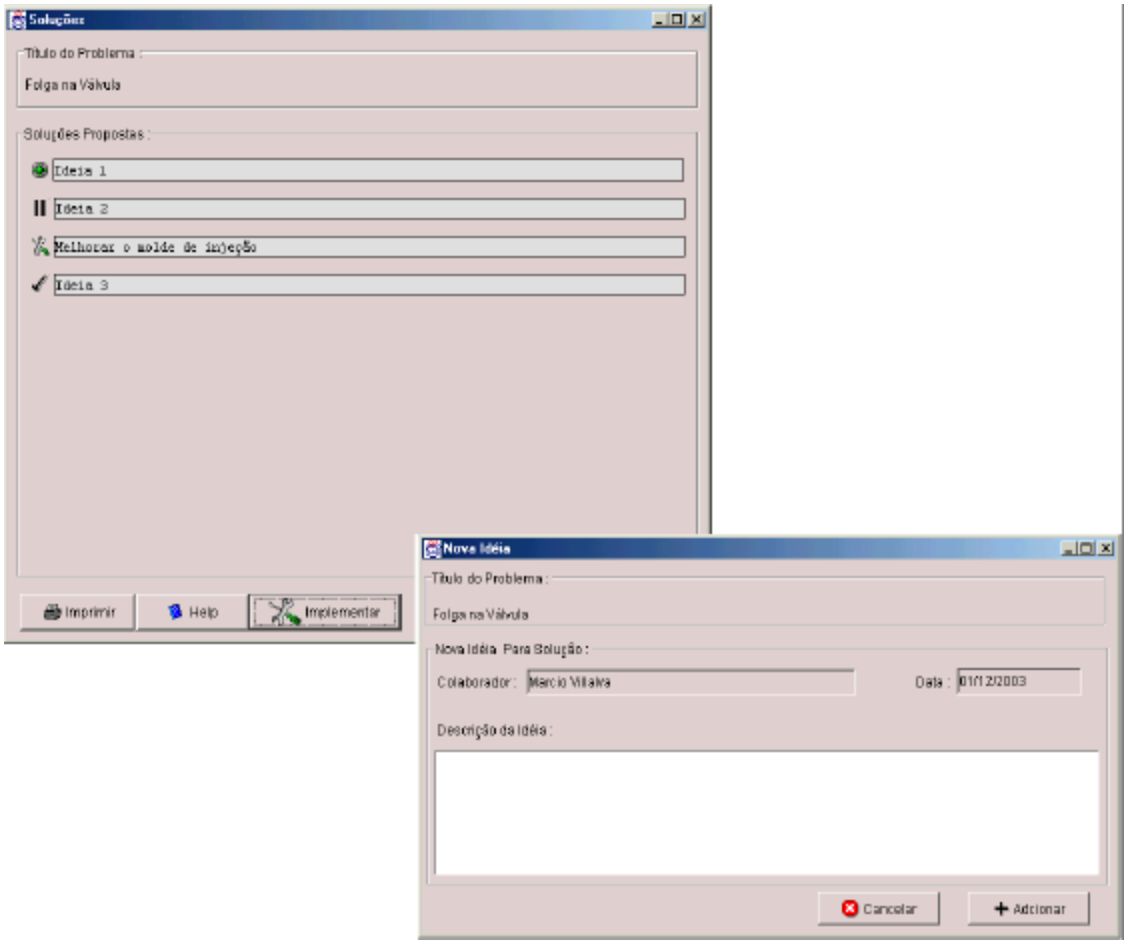


Figura 7.11 - Janelas do *Brainstorming* para propor soluções para o problema

A figura 7.11 apresenta a janela na qual o responsável pelo passo deve incluir ações para todas as idéias de soluções propostas que considerar válidas como solução ao problema em questão. Ele deverá então descrever as ações, definir um responsável para cada uma delas e atribuir uma data limite final para a sua implementação. Após isto, de

acordo com o estado real da ação, o responsável deverá manter o estado da ação no sistema que pode variar de: em andamento, suspensa, concluída e cancelada.

Conforme detalhado em Souza (2003) o responsável pelo passo poderá enviar, através do botão cobrar (canto inferior esquerdo da figura 7.12), *emails* para cada um dos responsáveis pelas ações, com o intuito de avisá-los. O módulo de *workflow* também envia *emails* automáticos para os responsáveis pelas ações, avisando-os com três dias e novamente com um dia de antecedência em relação à data final para a implementação da ação. Após a data final são envidados *emails* automáticos (diários) para cobrar os responsáveis sobre as ações pendentes.

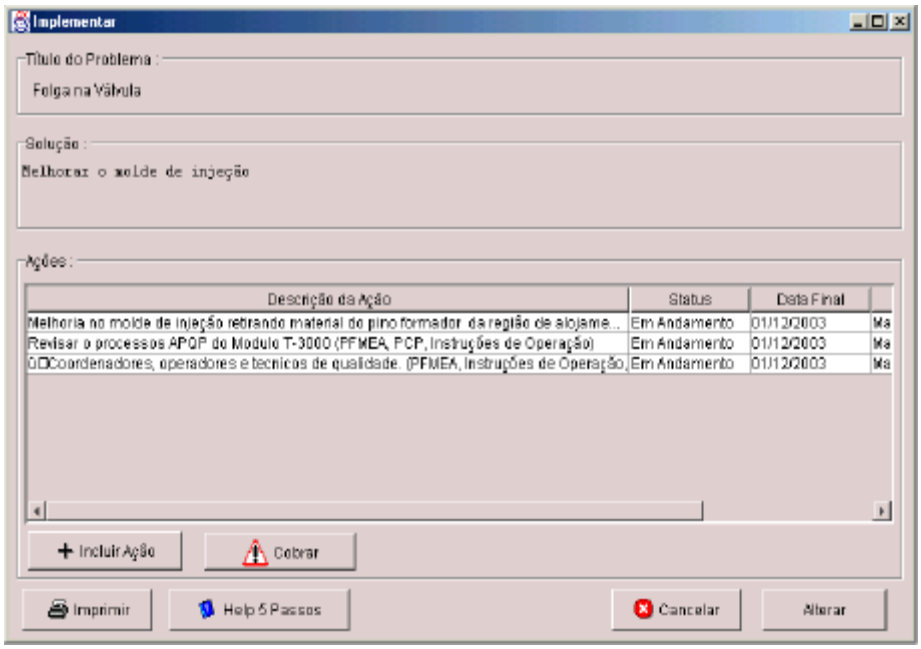


Figura 7.12 - Janela para especificar ações para soluções para um problema

A figura 7.13 exibe o quinto passo no processo de resolução de problemas do sistema Pokayoke. Neste passo o responsável (normas especificam que usualmente é quem emitiu o problema) deve descrever como cada ação foi implementada e também a eficácia destas. O responsável verifica também se a solução atingiu as peças, procedimentos, processos e setores definidos no passo três; caso a resposta seja "não" novas soluções e ações deverão ser propostas para solucionar o problema.

Por último, o responsável responderá se ele considera as ações propostas eficazes. Caso as ações não sejam consideradas eficazes um novo "Cinco Passos" deverá ser

emitido. Se forem consideradas eficazes, o “Cinco Passos” é finalizado e o problema, até então, na pasta de “Problemas em Resolução” passará a integrar a pasta de “Problemas Validados/Solucionados” da tela principal exibida na figura 7.3. O sistema Pokayoke também irá permitir que o time multifuncional discuta os aspectos positivos e negativos das soluções adotadas durante a resolução do problema.

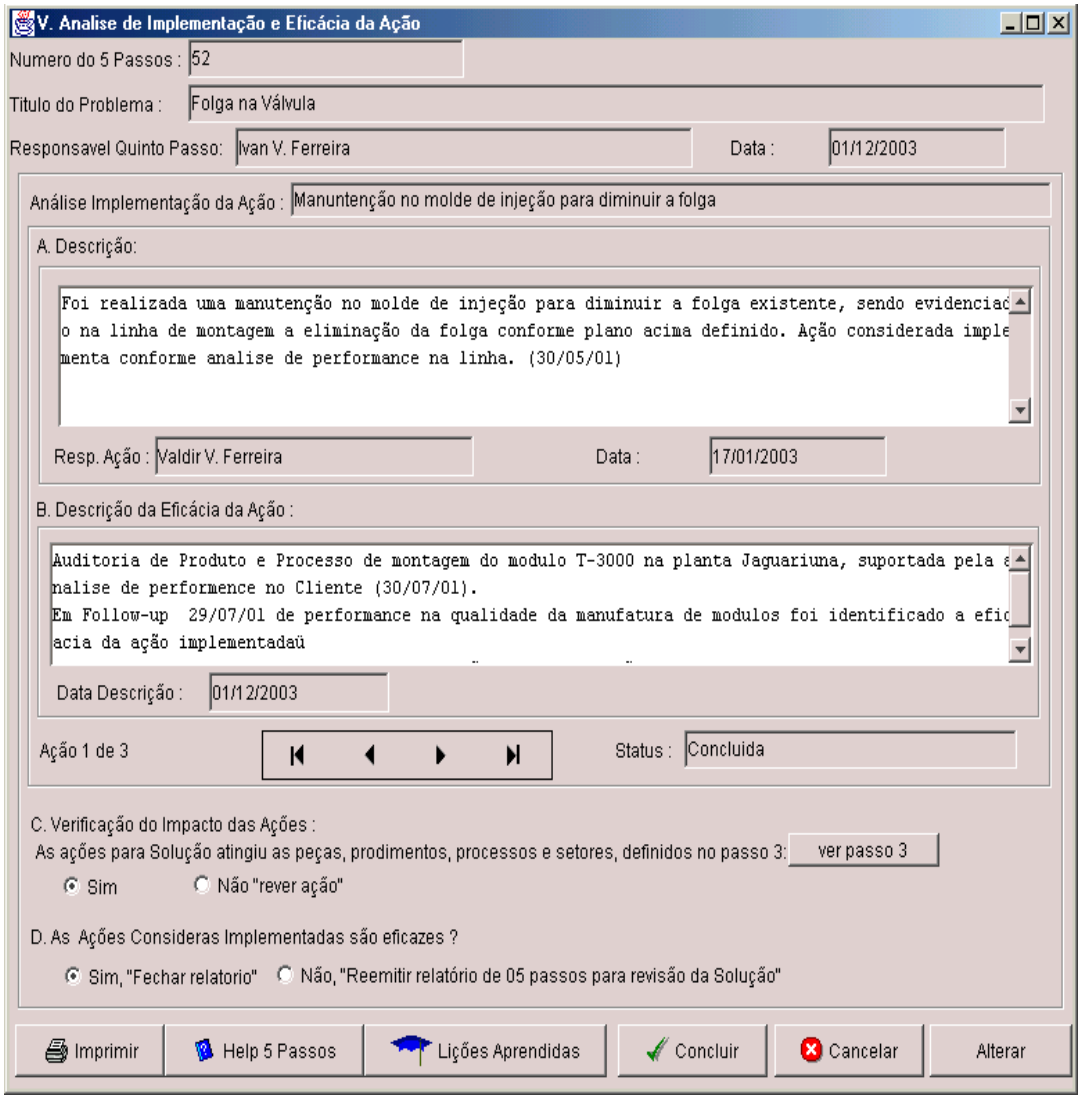


Figura 7.13 - Janela do quinto passo do sistema Pokayoke

Considerando as dificuldades levantadas pelos trabalhadores em relação à versão do “Cinco Passos” anterior ao sistema Pokayoke, exibidas na última seção, podemos dizer que o sistema Pokayoke minimizou a primeira e a segunda dificuldade pela adoção de ferramentas de comunicação assíncronas que possibilitam a participação direta de todos

os funcionários da empresa. Quando um usuário entra no sistema ele pode visualizar todos os problemas da fábrica para os quais tenha permissão ou obrigação de ver e opinar, possibilitando que ele participe da resolução do problema e construa soluções a distância (Intranet ou Internet) seguindo o padrão estabelecido no procedimento de "Cinco Passos". Mecanismos de controle (*follow-up* eletrônico) e cobrança também estão disponíveis e o usuário é chamado para participar da resolução do problema sempre que for responsável por uma determinada tarefa.

Em relação à terceira dificuldade destacada na versão anterior do "Cinco Passos", podemos dizer que ela foi minimizada pois o sistema Pokayoke leva os usuários a preencherem os passos em uma seqüência determinada, por exemplo: normas estabelecem que não é possível encerrar o quarto passo se as ações para resolver o problema não estejam concluídas. Problemas fictícios podem ser facilmente criados para novos usuários e a resolução colaborativa/cooperativa permite que usuários experientes auxiliem os novos usuários. Além disso, houve um treinamento no qual os próprios trabalhadores que participaram do desenvolvimento do sistema foram responsáveis por treinar os demais.

Em relação à quarta dificuldade destacada, no sistema Pokayoke temos um procedimento único para todas as "plantas" da fábrica. Além disso, todos os problemas solucionados estão armazenados no sistema e os usuários podem acessar estes problemas (eles podem ver os problemas para os quais têm permissão conforme especificação de normas) durante a resolução de um outro problema.

O anexo B apresenta os diagramas de ontologia e resultados da análise de normas que descreve o contexto organizacional pretendido, que inclui o sistema Pokayoke.

7.2.2 O SPaM Durante o Design do Sistema Pokayoke

Nesta seção é detalhado o uso do SPaM durante o desenvolvimento do sistema Pokayoke. A subseção 7.2.2.1 apresenta os ciclos de prototipação do sistema Pokayoke e como e quais técnicas de DP foram distribuídas durante estes ciclos. A subseção 7.2.2.2 apresenta uma discussão de como estas técnicas de DP contribuíram para a Análise Semântica e de Normas durante o estudo de caso. A subseção 7.2.2.3 apresenta um exemplo da aplicação da Conferência Semiótica no contexto de desenvolvimento do

sistema Pokayoke. Ao final, a subseção 7.2.2.4 apresenta aspectos do projeto do sistema Pokayoke com enfoque na sua arquitetura e na construção da interface.

7.2.2.1 Ciclos de Prototipação e Aplicação de Técnicas de Design Participativo

O tempo total de desenvolvimento do sistema Pokayoke foi de 14 meses, distribuídos em cinco ciclos de prototipação. O primeiro ciclo teve cerca de 3 meses, ao final deste ciclo tínhamos um primeiro modelo do contexto organizacional pretendido (diagrama de ontologia e normas) e uma primeira versão da interface do sistema composta de telas não funcionais. O segundo ciclo teve cerca 4 meses e resultou em um modelo mais detalhado do contexto organizacional futuro e em uma primeira versão funcional do sistema, porém limitada, incompleta e ainda para uso "mono-usuário". O terceiro ciclo teve cerca de 2 meses e resultou na primeira versão funcional "multi-usuário" do sistema e instalada na fábrica em máquinas de usuários pilotos. Porém, esta era ainda limitada em vários aspectos. O quarto ciclo teve cerca de 2 meses e produziu a primeira versão com todas as funcionalidades do sistema existente atualmente (com exceção do *workflow*), porém alguns aspectos ainda deveriam ser avaliados e refinados pelo usuário. O quinto ciclo teve cerca de 3 meses e produziu o sistema ajustado ao uso em larga escala, implantou o sistema em substituição à versão em papel do "Cinco Passos" e resultou em usuários treinados pelos próprios trabalhadores que participaram do desenvolvimento do sistema.

A versão atual do sistema Pokayoke é resultado de cinco ciclos de prototipação utilizando o SPaM. Na figura 7.14 as setas de cima para baixo representam o uso de técnicas de DP (relacionadas pelas setas tracejadas) durante os ciclos de prototipação do sistema Pokayoke.

Estas técnicas foram escolhidas com base na sua adequação a diferentes fases do desenvolvimento do sistema, a facilidade de aplicá-las no contexto de trabalho, objeto deste estudo de caso, a possibilidade de serem articuladas em conjunto com as técnicas de SO, o potencial para estimular a colaboração e a reflexão durante o design. Seguindo a figura 7.14 temos:

- *Starting Conference* foi aplicada durante o primeiro, terceiro, quarto e quinto ciclos de prototipação. Seus resultados auxiliaram na realização dos passos 1.1, 1.2 e 1.4 do SPaM;
- *Ethnographic practices* foi aplicada durante o primeiro e o quarto ciclo. Seus resultados auxiliaram na realização dos passos 1.1, 1.2 e 1.4 do SPaM;
- *HOOTD(Hierarchical Object-Oriented Task Decomposition)* foi aplicada durante o primeiro ciclo. Seus resultados auxiliaram na realização do passos 1.1, 1.2 e 2.2 do SPaM;
- *Icon Design Game* foi aplicado durante o primeiro ciclo. Seus resultados auxiliaram na realização dos passos 1.1, 1.2, 1.4, 2.2 e 3.1 do SPaM;
- *Artefact Walkthrough* foi aplicado durante o primeiro, terceiro e quarto ciclo. Seus resultados auxiliaram na realização dos passos 1.1, 1.2, 1.4, 2.2 e 3.1 do SPaM;
- *Prototyping* foi aplicada durante os cinco ciclos. A técnicas de *Storyboard Prototyping* e CISP foram utilizadas em substituição às fases 3 e 4 durante o primeiro ciclo. Esta técnica também auxiliou na discussão de novas funcionalidades nos demais ciclos, principalmente durante o passo 3.1 e para viabilizar discussões com o usuário sobre o impacto de aspectos técnicos presentes na fase 4;
- *Participatory Heuristic Evaluation (PHE)* foi aplicado durante o terceiro, quarto e quinto ciclos. Esta técnica, junto com a Conferência Semiótica foi utilizada na fase 1 e 2 no último ciclo de prototipação para dar um enfoque maior na avaliação do sistema em uso no contexto organizacional como alavanca para oportunidades futuras de aprimoramento das práticas de trabalho atuais;
- Conferência Semiótica foi aplicada durante o primeiro, segundo e quinto ciclos de prototipação, auxiliando no passo 1.3 conforme proposto no SPaM.

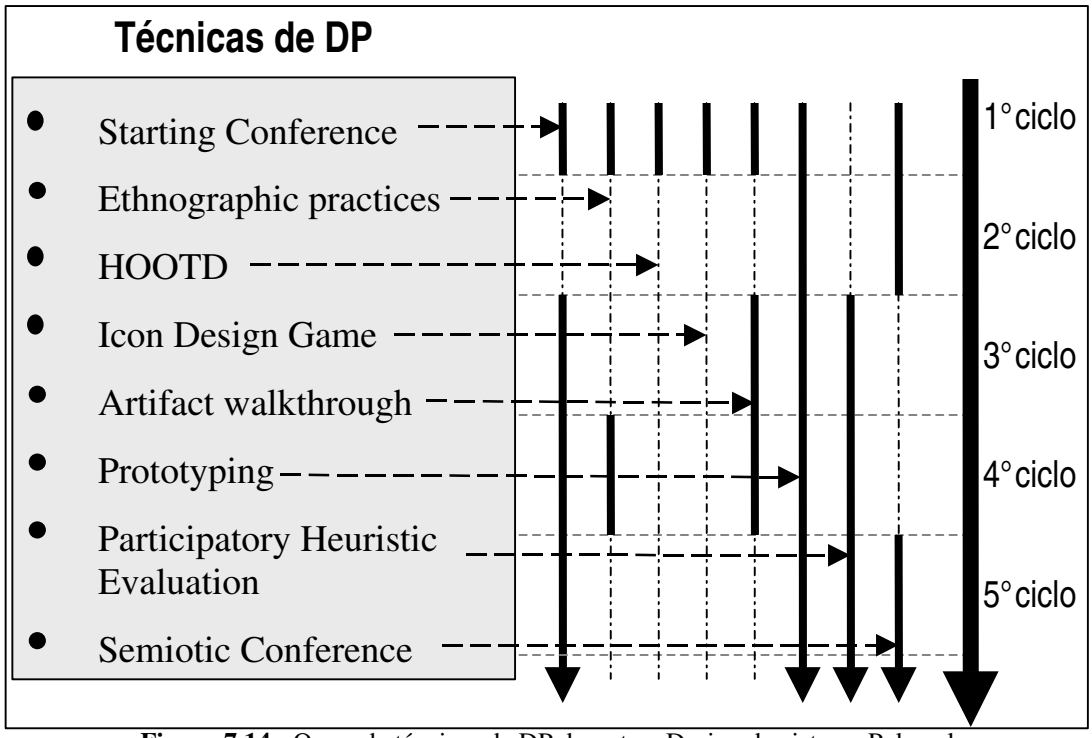


Figura 7.14 - O uso de técnicas de DP durante o Design do sistema Pokayoke

Neste estudo de caso, a primeira versão do diagrama de ontologia (passo 1.2) foi construída com base nos resultados das técnicas de design participativo aplicadas durante a construção do sistema Andon (Silva, 2001; Silva e outros 2001). Desta maneira o passo 1.1 do SPaM foi realizado previamente e continha as seguintes técnicas utilizadas para a construção dos modelos da SO : Starting Conference, Ethnographic practices, HOOTD, Icon Design Game e Artifact Walkthrough. Detalhes sobre a aplicação destas técnicas podem ser encontrados em Silva (2001). Portanto, durante a primeira aplicação da Conferência Semiótica, foi utilizado um modelo semântico derivado dos resultados destas técnicas, entretanto não foram utilizadas telas do sistema Andon como artefato de apoio.

A tabela 7.1 é um resumo dos relatórios de visitas dos designers e pesquisadores à organização durante o desenvolvimento do sistema Pokayoke. Durante as visitas eram realizadas diversas atividades, algumas relacionadas com a aplicação de técnicas de design participativo. Desta maneira, em alguns casos, eram realizadas mais de uma atividade participativa por visita. Cada visita possuía um objetivo principal a ser cumprido, presentes na tabela 7.1. Esta tabela também exibe os participantes de cada uma

delas, a data em que ocorreram, a duração em horas e o ciclo de prototipação em que ocorreram. O anexo A contém estes relatórios completos e alguns documentos que ilustram os artefatos trabalhados durante as visitas.

Tabela 7.1 - Resumo dos relatórios de visitas à organização

N°	Atividades	Objetivo Principal	Particip.	Data	Duração	Ciclo
1	Reunião	Estabelecer objetivos e pontos de contato	5 univer. (pesquisadores) 4 Organ. (gerentes)	08/11/2001	4 horas	1
2	Conferência Semiótica, *Storyboard Prototyping* Avaliação do sistema Andon, Artifact Walkthrough	Discutir o contexto organizacional através da SO, Avaliar o sistema Andon Design de alto nível da interface	2 designers 4 trabalh.	30/11/2001	4 horas	1
3	Avaliação da interface *CISP*	Discutir e evoluir primeiro protótipo em papel	2 designers 4 trabalh.	04/02/2002	3 horas	2
4	Conferência Semiótica (agora com apoio de telas em papel)	Discutir conceitos da organização e como eles são refletidos no sistema	2 designers 2 trabalh.	27/03/2002	4 horas	2
5	Estabelecer novo contato Avaliação do protótipo *Artifact walkthrough*	Estabelecer novo contato Avaliar protótipo (agora funcional) Verificar documentos padrões da fábrica	2 designers 2 trabalh.	03/06/2002	3 horas	3
6	Instalação de máquinas	Instalar máquinas para colocar protótipos do sistema para testes	2 designers	19/06/2002	2 horas	3
7	*Starting Conference* Instalação do Sistema usuários piloto	Discutir implantação do sistema e estratégia de uso na organização	1 designer 3 trabalh.	06/07/2002	4 horas	3
8	PHE Teste com usuários	Avaliação da interface do sistema, teste do sistema em condições de uso e decisões de design	1 designer 2 trabalh.	13/07/2002	4 horas	3
9	Configuração das máquinas para teste *Artifact walkthrough* *Starting Conference* Imersão (Ethnographic practices)	Avaliar e verificar sistema, Ferramentas de follow-up para o Pokayoke-flow e Verificar contexto de uso da ferramenta	3 designers 3 trabalh.	24/07/2002	9 horas	4
10 - 11	Avaliação do Protótipo PHE	Avaliação do sistema, discutir design da interface	3 designers 7 trabalh.	15/09/2002 22/10/2002	4 horas	4
12	Avaliação o sistema – PHE e Conferência Semiótica *Starting Conference*	Avaliar o sistema, discutir alterações na interface e discutir implantação do sistema com o usuário	2 designers 3 trabalh.	26/11/2002	5 horas	5

13 - 14	*Starting Conference* Instalar máquinas Reunião	Definir com o usuário como colocar o Pokayoke em produção, o treinamento e a atividade de suporte	3 designers 4 trabalh.	03/12/2002 17/12/2002	4 horas	5
15	Reunião	Definir com a organização esquema para manutenção	2 designers 3 trabalh.	07/01/2003	2 horas	5
-	-	-	-	-	52 horas	-

A tabela 7.2 apresenta um resumo do relatório de uma atividade em grupo desenvolvida com pesquisadores das áreas de IHC e SO. Esta atividade compreendeu duas atividades menores: (1) a utilização do sistema na resolução de problema fictício extraído do cotidiano do grupo e (2) a aplicação da técnica de PHE para incluir a visão de especialistas em IHC (em adição a dos usuários) na avaliação, conforme proposto por Müller e outros (1998).

Tabela 7.2 - Relatório de teste do sistema com especialistas

N°	**Atividades**	**Objetivo Principal**	**Particip.**	**Data**	**Duração**	**Ciclo**
U1	Uso do sistema na resolução de um problema fictício PHE	Avaliar o sistema e discutir a técnica de PHE	5 Pesquisadores de Interfaces e SO	20/11/2002	4 horas	5

7.2.2.2 Construção dos Modelos da SO a partir dos Resultados das Técnicas de DP

Além da Conferência Semiótica, que trabalha diretamente com o diagrama de ontologia, outras técnicas de DP podem contribuir durante as quatro fases propostas por Liu (2000) para análise semântica: *Definição do problema*, *Geração de affordances candidatos*, *Agrupamento de Candidatos* e *Diagramação Ontológica*. Cada uma das técnicas analisada contribuiu de maneira complementar dando suporte a tarefas em todas as fases da análise semântica e cobrindo aspectos de todos os níveis da "organizational onion" (Stamper, 1992). Com base na análise de como estas técnicas contribuíram durante o desenvolvimento do sistema Pokayoke, é apresentado como elas podem contribuir para análise semântica (a descrição destas técnicas está na seção 3.1.2):

- *Starting Conference.* Esta atividade contribui principalmente na primeira fase da análise semântica (identificação do problema) e na descrição que contém os agentes básicos, *affordances* e dependências ontológicas que são utilizadas em outras fases. Ela também contribui para uma descrição primária dos objetivos dos sistemas. Esta técnica pode ser utilizada para apoiar a modelagem de alguns conceitos nos níveis técnico, formal e informal (organizational onion);
- *Práticas de etnografia*. Estas atividades contribuem para as quatro fases da análise semântica, sendo possível modelar de maneira mais detalhada agentes, *affordances* e dependências. Ela é uma técnica importante para compreender o nível informal da organização. Técnicas de etnografia incluem mecanismos para a compreensão do contexto social pela observação das práticas de trabalho em seu contexto real. Desta maneira, o designer pode identificar o significado real dos conceitos, como estes significados são estabelecidos, o comportamento e intenções dos agentes e normas sociais informais;
- *Artifact walkthrough*. Esta técnica contribui para as quatro fases da análise semântica apoiando o designer na fase de definição do problema, para identificar novos *affordances*, agentes e dependências e, principalmente, para compreender o relacionamento entre as práticas de trabalho e os artefatos. Resultados desta técnica alimentam as últimas fases da análise semântica, onde os conceitos são agrupados e o diagrama de ontologia é construído. Utilizando esta técnica o designer pode compreender melhor o nível formal (burocrático) e técnico da organização através do estudo dos artefatos formais utilizados no local de trabalho (exemplo: manuais, guias, normas, etc);
- *HOOTD (Hierarchical Object-Oriented Task Decomposition)*. Esta técnica é direcionada ao design de interfaces gráficas, mas resultados deste método podem ser utilizados também para compreender a dependência entre agentes e *affordances* e para agrupar estes conceitos, contribuindo para as duas últimas fases da análise semântica. Utilizando esta técnica, o designer pode

compreender as dependências formais e informais, e também o agrupamento hierárquico dos conceitos trabalhados;

- *Icon design game*. Esta técnica está voltada ao design de ícones de interface, mas ela pode ser utilizada também para identificar os significados dos signos para o grupo de trabalho (nível informal). Analisando como estes signos são entendidos pelo grupo de trabalho, resultará durante a análise semântica em uma representação mais consistente com o contexto organizacional. Esta técnica contribui principalmente para a primeira fase (gerar *affordances* candidatos) e para a primeira parte da terceira fase (onde são agrupados e categorizados).

Também foi investigado como algumas técnicas de DP contribuíram para a análise de normas. As técnicas de DP foram fundamentais para capturar as regras que regiam o comportamento dos agentes na organização, uma vez que os trabalhadores puderam compartilhar e prover o *feedback* a respeito da compreensão do ambiente de trabalho. Além disso, os trabalhadores estabeleciam novas regras sobre o contexto organizacional durante a aplicação das técnicas de DP. Com base na análise de como as técnicas de DP contribuíram na condução da análise de normas no desenvolvimento do sistema Pokayoke, a seguir apresentamos como elas podem apoiar a análise de normas:

- *Starting Conference*: podem ser realizadas para discutir normas específicas possibilitando identificar como elas realmente são utilizadas no local de trabalho. Utilizando esta técnica os designers podem identificar as diferentes interpretações de uma mesma norma em diferentes níveis da organização. Esta técnica também pode ser utilizada para discutir possíveis aprimoramentos da organização através do uso de artefatos computacionais e também qual seria o impacto destes artefatos nas normas sociais vigentes;
- *Práticas etnográficas*: As técnicas da etnografia podem contribuir para a análise de normas tornando explícito o conhecimento sobre o contexto social, através da observação dos trabalhadores em seu ambiente de trabalho real. Utilizando estas técnicas designers podem modelar normas informais, ajudando a compreender o comportamento dos agentes quando submetidos a

normas formais (burocráticas), estabelecendo o relacionamento entre o nível formal e informal.

- *Artifact walkthrough*. Esta técnica pode ser utilizada para modelar o comportamento formal, estudando artefatos que contenham descrições formais das práticas de trabalho, regras, leis e padrões. Em uma organização normalmente existem muitos documentos que especificam como as atividades de trabalho deveriam ser feitas, os objetivos nos planos operacional, tático e estratégico e a estrutura da organização. Utilizando esta técnica o designer pode identificar como as normas descritas nestes artefatos podem influenciar o contexto de trabalho.

7.2.2.3 Aplicação da Conferência Semiótica

Conforme apresentado anteriormente, a Conferência Semiótica (descrita na seção 5.4) foi explorada durante o primeiro, terceiro, quarto e quinto ciclos de prototipação do SPaM, durante o desenvolvimento do sistema Pokayoke. Seus resultados foram importantes no início de cada iteração, com o objetivo de revisar conceitos do protótipo construído no ciclo anterior e redirecionar o design do próximo protótipo, e em outros momentos quando necessário para revisar conceitos. A Conferência Semiótica serve como um ponto de partida para revisão e design de protótipos. De certa maneira é a Conferência Semiótica quem direciona os trabalhos desenvolvidos durante as duas primeiras fases do ciclo de prototipação: (1) *O design e revisão do contexto organizacional,* e o (2) *Levantamento de requisitos do sistema e design de alto nível*.

A figura 7.15 exibe um exemplo do uso da Conferência Semiótica durante o design do Pokayoke. A figura 7.15a (esquerda) refere-se a parte de um diagrama de ontologia que representava conceitos do terceiro passo no processo de resolução de problemas (diagramas de ontologia do contexto organizacional do sistema Pokayoke podem ser encontrados no anexo B). Este diagrama (figura 7.15a) foi construído durante a segunda interação no ciclo de prototipação do SPaM. Esta figura exibe desenhos feitos por um participante como resultado de várias discussões durante a Conferência Semiótica, que

sugerem modificações no modelo semântico. Basicamente, neste exemplo, são propostas duas alterações:

- O *Time de ação* é ontologicamente dependente do *Responsável pela Implementação* e não o contrário como estava no diagrama anterior, ou seja, o *Time* só existe durante a existência do responsável, não existe *Time de ação* sem um responsável pela implementação, mas podem existir responsáveis pela implementação sem t*ime de ações;*
- A parte do digrama dentro do ciclo na figura 7.15a deve estar no diagrama do quarto passo e não no terceiro, visto que a solução do problema é discutida durante o quarto passo e não durante o terceiro passo como mostrava o diagrama;

Estas alterações desencadearam a aplicação de outras técnicas de DP e outras aplicações da Conferência Semiótica durante os próximos ciclos de prototipação. As técnicas *Starting Conference* e *Artifact walkthrough* (relatórios 5 e 7 do anexo A) foram aplicadas com o objetivo de identificar como as soluções para os problemas surgem durante o quarto passo e uma nova Conferência Semiótica foi conduzida para discutir o modelo no quarto passo após as alterações. A figura 7.15b (direita) é parte do diagrama no terceiro ciclo de prototipação. Esta parte do digrama modela os conceitos que foram alterados na figura 7.15a, durante o segundo passo. Além das mudanças sugeridas destacadas anteriormente, outras modificações foram introduzidas durante o ciclo, entre elas a inclusão de um *Brainstorming* como gerador de idéias para as soluções, a modificação do nome do *affordance Responsável pela Implementação* para *Responsável pelas Ações*, a inclusão dele como "papel" e a revisão de alguns conceitos de modelagem como a exclusão da dependência causal dos diagramas.

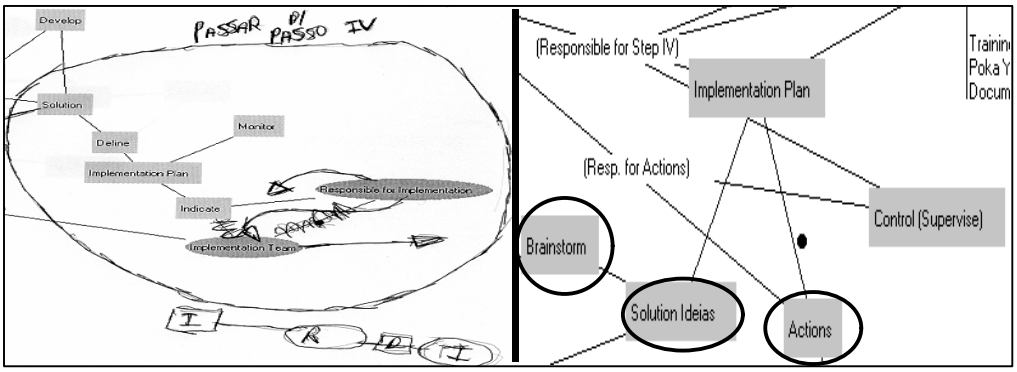


Figura 7.15(a e b) - Alterações nos Diagramas de Ontologia Sugeridas pela Conferência Semiótica

A figura 7.16 exibe alterações na interface do usuário causadas pelas mudanças no diagrama de ontologia apresentado na figura 7.15. Os *affordances* marcados com círculos na figura 7.15b são associados a construções de interface na figura 7.16b. As alterações na interface são inicialmente sugeridas durante a Conferência Semiótica. As soluções que inicialmente eram propostas no passo III conforme modelo da figura 7.15a foram propostas para o passo quatro conforme modelo da figura 7.15b. O desenho exibido na figura 7.16a é resultado de uma discussão (ocorrida no segundo ciclo de prototipação) sobre como gerar soluções de problemas no passo IV. Estes desenhos, então, são transformados em interfaces, conforme é sugerido pela arquitetura que integra o ICE.

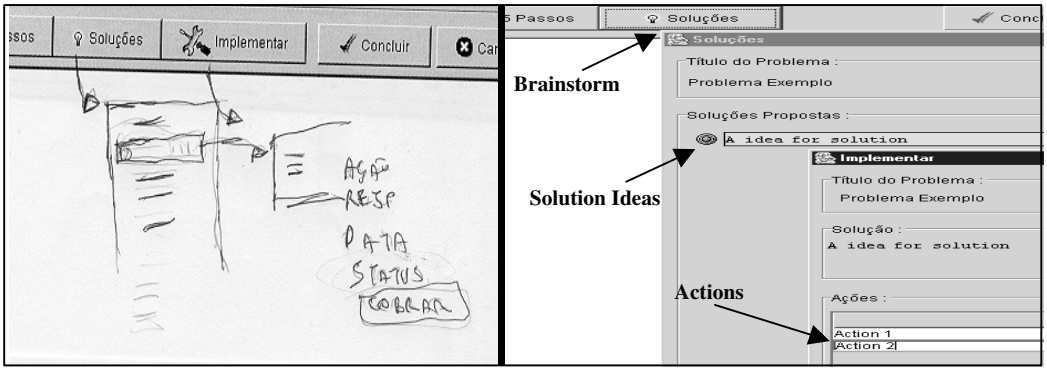


Figura 7.16(a e b) - Alterações na interface do sistema

As dependências ontológicas alteradas durante a Conferência Semiótica (relacionadas aos objetos de interface) são representadas por condições para habilitar/desabilitar funções e pela exibição dos objetos de interface. Como exemplo é

necessário existir um *Brainstorming* para poder incluir idéias para as soluções conforme sugere a arquitetura que integra o NBIC.

As alterações em normas durante a Conferência Semiótica (como por exemplo responsabilidade, deveres e permissões) são refletidas diretamente pelo acesso de algumas funções do Pokayoke (via NBIC), por exemplo: qualquer usuário pode incluir uma idéia em uma sessão de *Brainstorming* mas não pode especificar ações para corrigir o problema. As normas também especificam quem pode ser responsável por cada tarefa em diferentes passos no processo de resolução de problemas.

7.2.2.4 Aspectos do projeto do sistema Pokayoke

Modelos iniciais de projetos foram construídos com base nos procedimentos propostos para serem utilizados durante o design de alto nível (passos 2.2, 2.3 e 2.4 do SPaM). Em relação ao procedimento proposto para o passo 2.3, que resulta em um modelo inicial de projeto para o nível de negócio, podemos dizer que os passos e regras heurísticas que o constituem produziram diagramas consistentes quando aplicados ao contexto de desenvolvimento do sistema Pokayoke. O procedimento possibilitou construir modelos OO a partir dos diagramas de ontologia.

Entretanto em alguns casos a aplicação destas regras heurísticas incluía a escolha de opções de design de acordo com a perspectiva da OO, e também uma análise mais profunda sobre o diagrama de classes produzido. Por exemplo: um *papel* em um diagrama de ontologia pode sugerir uma relação hierárquica ou um papel em uma classe. Estes problemas podem aparecer somente durante as fases 3 e 4 do SPaM, mas foram raros os casos no contexto do sistema Pokayoke.

Durante as fases 3 e 4 do SPaM estes modelos iniciais são alterados e outros fatores ainda inexplorados pelo modelo de ontologia e normas nas fases anteriores passam a ser importantes, como por exemplo a performance e a arquitetura do sistema. Desta maneira estes modelos devem incluir alguns aspectos novos dependentes de implementação. As subseções seguintes apresentam a arquitetura do Pokayoke com seus principais componentes e o desenvolvimento da interface do sistema, baseado na arquitetura proposta nesta tese, para possibilitar mudanças organizacionais a partir de alterações na especificação de normas.

Arquitetura do Pokayoke

O sistema Pokayoke foi construído em Java e utiliza RMI (Remote Method Invocation) como plataforma para comunicação. A figura 7.17 apresenta a arquitetura do sistema Pokayoke, constituída de quatro blocos principais (representados pelos retângulos tracejados): cliente, servidor de aplicação, servidor de dados e estação NBIC.

O cliente é executado na máquina do usuário e contem basicamente a interface do sistema (estática e dinâmica), as classes controladoras e uma máquina de inferência. A interface está dividida conforme a arquitetura de interface proposta no capítulo 6 (ICE). A comunicação entre a interface e qualquer outra parte do sistema ocorre via classes controladoras; estas classes compõem gerenciadores que controlam os diferentes objetos do sistema (por exemplo: gerenciador de Brainstoming, gerenciador de ações, gerenciador de problemas, etc). Na versão em laboratório, foi utilizado o Jess (Java Expert System Shell) como máquina de inferência. Na versão em produção na fábrica foi implementada uma pequena máquina de inferência simplificada, para otimizar a performance do lado do cliente e também facilitar a distribuição.

Conforme mostra a figura 7.17, a comunicação entre o cliente e o servidor de aplicação ocorre via RMI. O servidor de aplicação contém as classes de negócio, uma camada de persistência de objetos e o módulo de envio automático de emails do Pokayoke-flow que funciona como uma aplicação independente, com acesso direto ao banco de dados (sem intermédio da camada de persistência).

Conforme a figura 7.17 exibe, o servidor de dados contém três bancos de dados chamados de: PokaBanco, InterNorms e Normas. O primeiro é o banco de dados do sistema Pokayoke onde os problemas são armazenados, o segundo é o banco de dados que contém as normas já traduzidas para a linguagem da máquina de inferência e o terceiro é o banco de dados com as normas descritas em linguagem pseudo-natural utilizadas pelo NBIC. As comunicações destes bancos de dados com os sistemas ocorrem via JDBC.

O último bloco é a “estação NBIC” que é a máquina aonde o sistema NBIC é executado. Nesta figura, o NBIC é representado como um bloco único. A sua estrutura interna segue a arquitetura apresentada na figura 6.11.

Os modelos (de classes e de banco de dados) que compõem o Pokayoke são apresentados no anexo B. Os detalhes internos do NBIC assim como instruções sobre o seu uso podem ser encontrados no anexo C.

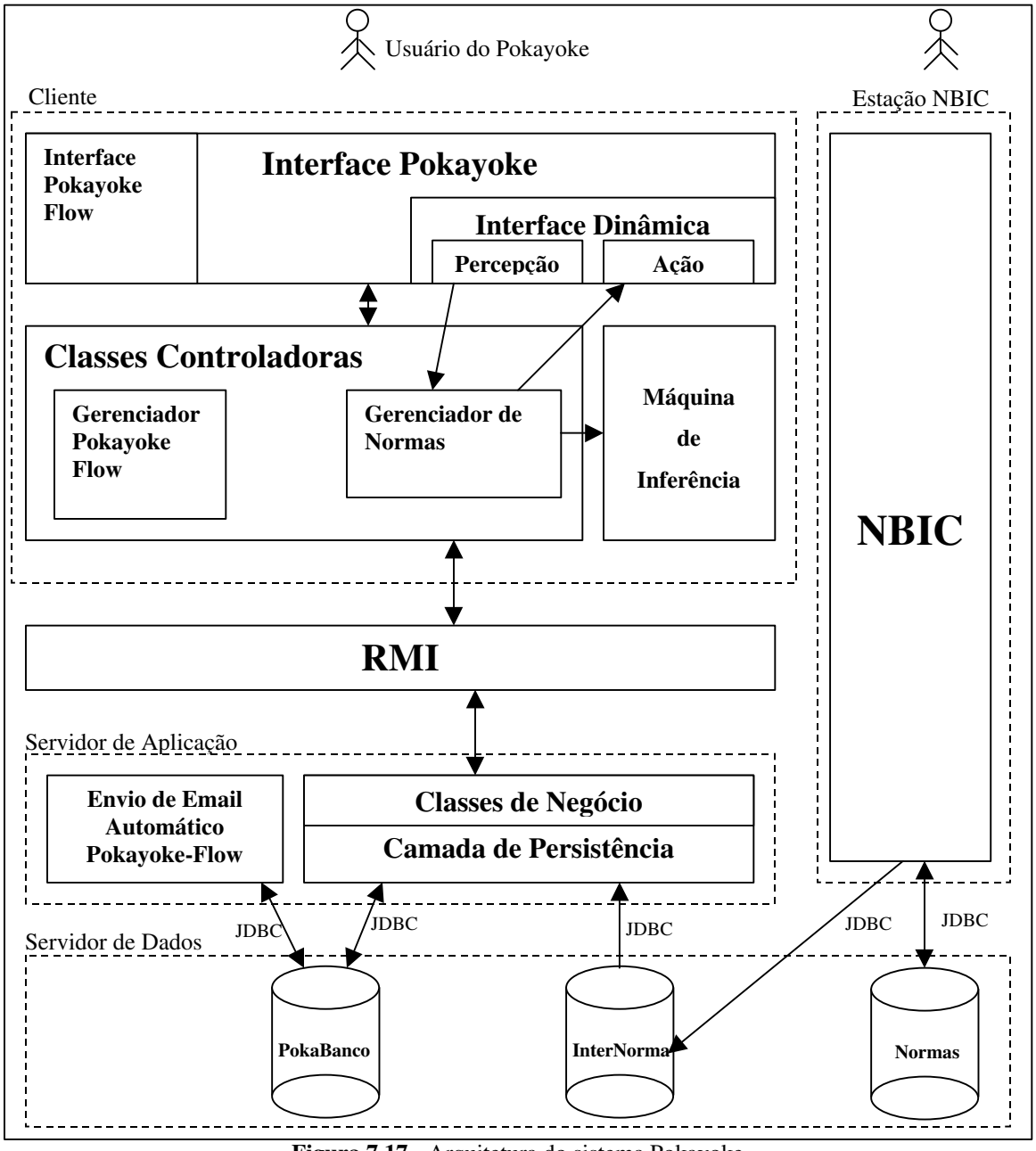


Figura 7.17 - Arquitetura do sistema Pokayoke

O Desenvolvimento da Interface

A figura 7.18 é um diagrama de ontologia (Análise Semântica) do Pokayoke, mais especificamente, um diagrama para o passo II (demais diagramas são apresentados no

anexo B). Ferramentas específicas no sistema Pokayoke dão suporte aos *affordances* humanos deste diagrama. Os *affordances* marcados com círculos (tracejados) na figura 7.18 são associados a funcionalidades do software conforme ilustra a figura 7.19.

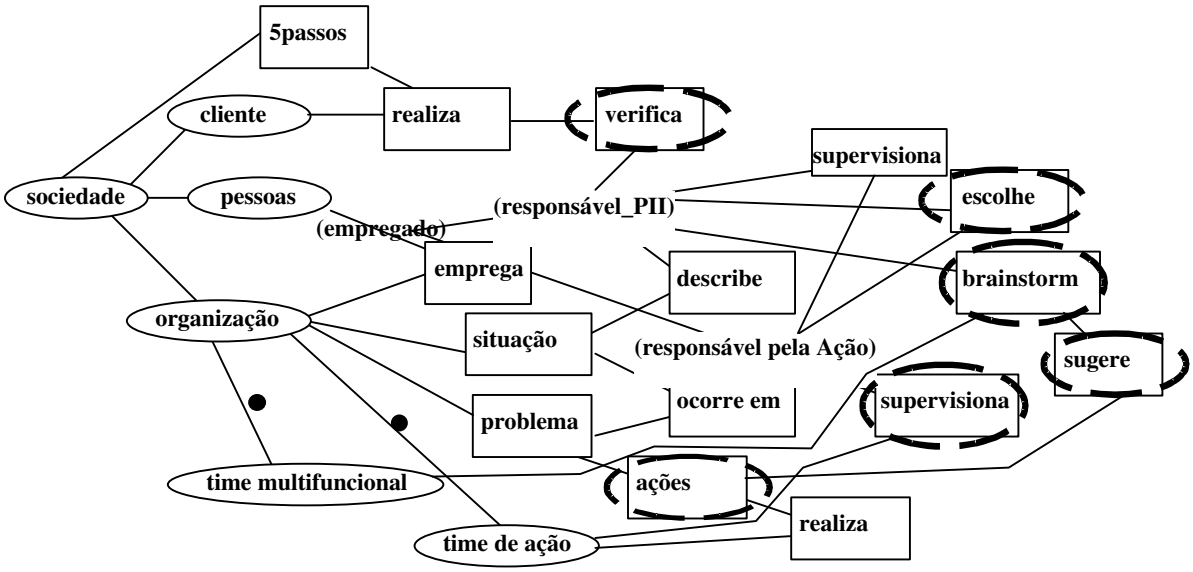


Figure 7.18 - Diagrama de ontologia do passo II

As interfaces foram construídas de acordo com os diagramas de ontologia. Os diagramas de ontologia especificam o que existe (ou o que deverá existir) no mundo e, conforme destacado anteriormente, um sistema de CSCW deve dar suporte as atividades humanas que existem ou supostamente existirão em um modelo organizacional futuro; este é o resultado do passo 2.2 do SPaM. Desta maneira, no design do Pokayoke durante as fases 3 e 4 do SPaM, foram projetadas funcionalidades para dar suporte aos affordances "verifica", "brainstorm", "escolhe", "sugere", "supervisiona" e "ações". Estas funcionalidades estão presentes ou são acessadas através da interface que dá suporte ao passo II do processo de resolução de problemas.

O projeto e implementação da interface (passos 3.1 e 4.1 do SPaM) incluem detalhar como estas funcionalidades darão apoio ao comportamento humano e qual a representação gráfica adequada ao modelo detalhado da interface. Entretanto, estas não são as únicas atividades nestes passos. Também foi necessário codificar as funcionalidades que permitiram a configuração dinâmica da interface via NBIC.

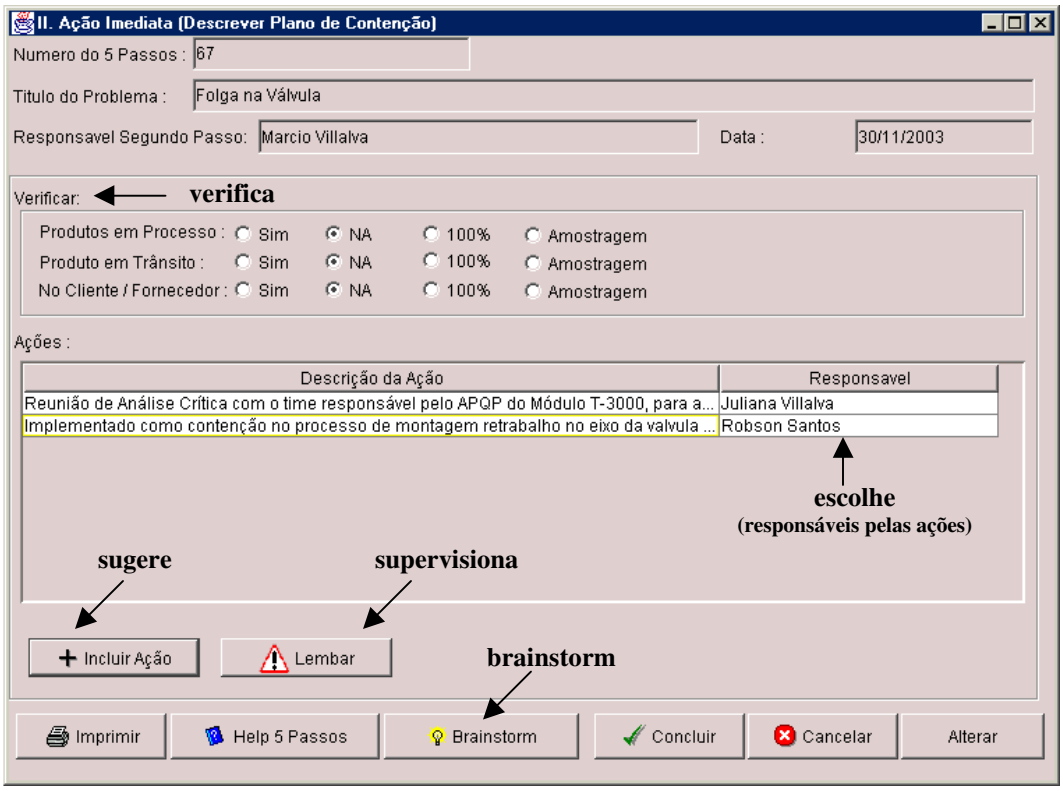


Figura 7.19 - Exemplos de funcionalidades do passo II vindas da definição dos affordances

No Pokayoke cada elemento gráfico possui "padrões" e pode ter seus atributos configurados via mecanismo de ação. Caso não exista norma relacionada ao objeto de interface, um valor "padrão" é assumido. Por exemplo: o mecanismo de ação poderá alterar a cor do objeto, condição de habilitado ou desabilitado, condição de visível ou não e assim por diante. Caso não exista norma, valores-padrão são assumidos para os atributos dos objetos de interface de acordo com a especificação do Pokayoke. Desta maneira, os passos 3.1 e 4.1 do SPaM incluem a especificação de normas e regras no NBIC respectivamente.

No sistema Pokayoke a parte dinâmica está restrita a especificação de normas que modelam responsabilidades, deveres e permissões no processo de resolução de problema, que estão relacionados com o acesso a algumas funções do sistema, a ordem das tarefas a serem realizadas e a especificação de quem poderia ser responsável por cada tarefa em diferentes fases no processo de resolução de problemas.

A Figura 7.20 ilustra como as normas e regras estão relacionadas a elementos da interface do sistema Pokayoke. Esta figura ilustra três situações (da esquerda para a

direita): (1) na primeira delas temos uma norma estabelecendo que é "permitido para os membros da equipe multifuncional participar do *Brainstorming*", uma regra especificando que a norma pode ser traduzida para "o botão de acesso à ferramenta de *Brainstorming* estar habilitado" e o efeito na interface do Pokayoke, (2) na segunda situação temos uma norma estabelecendo que é "proibido para um membro da equipe multifuncional participar do *Brainstorming*", uma regra especificando que o botão de acesso deve estar desabilitado neste caso e o efeito na interface do Pokayoke e (3) na terceira situação temos uma norma estabelecendo que "os membros do time multifuncional são obrigados a participar do *Brainstorming*", uma regra especificando que o botão de acesso à ferramenta de *Brainstorming* deve estar em destaque e o efeito na interface do Pokayoke.

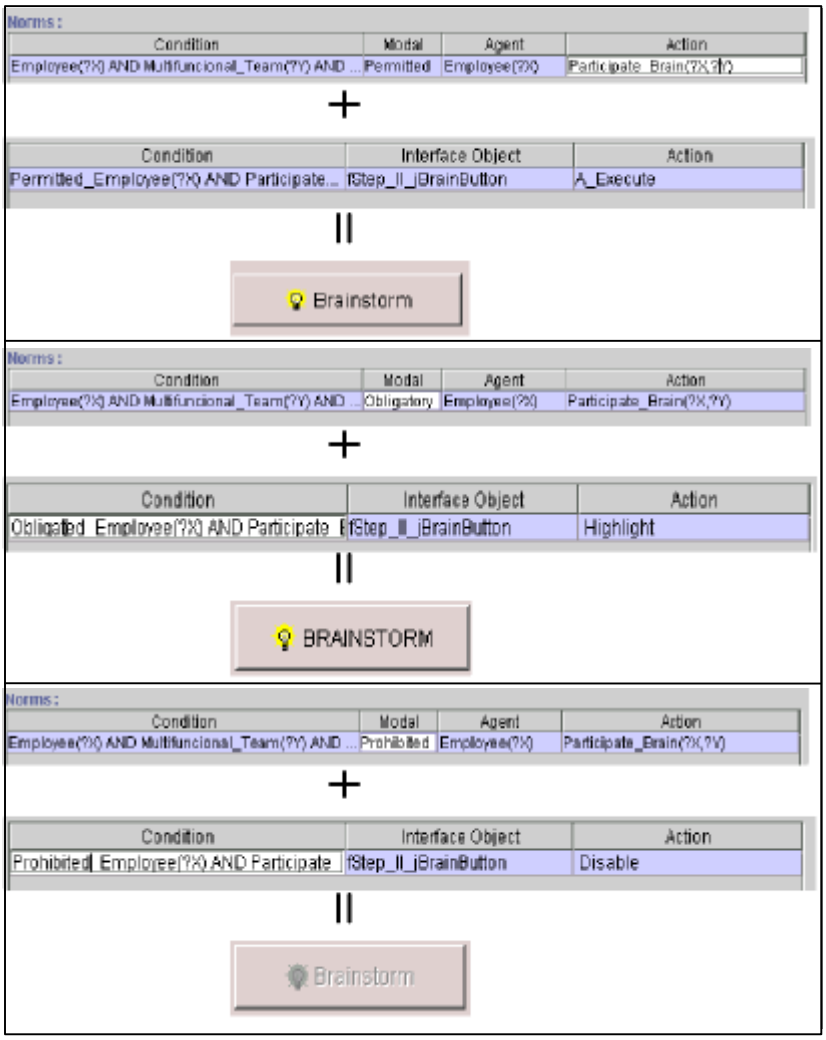


Figura 7.20 - Mudanças de normas e efeitos no sistema Pokayoke

7.2.3 O processo de Significação Durante o Design do Sistema

Os problemas detectados durante a aplicação das técnicas de design participativo PHE (*Participatory Heuristic Evaluation*) e Conferência Semiótica foram classificados em dois grupos: (1) *orientado a produto* que enfoca o artefato de software e (2) *orientado a processo* que enfoca o processo humano para o qual o artefato computacional pretende dar suporte (Floyd, 1987; Mahemoff e Johnston, 1998; Müller e outros, 1995,1998). Embora não seja objetivo desta tese comparar a Conferência Semiótica e o PHE, nós podemos dizer que as duas técnicas têm naturezas diferentes e uma complementa a outra no SPaM. Enquanto o PHE tem seu foco na interface do usuário (contribuindo mais para a segunda fase do SPaM) e indicando principalmente problemas orientados a produto, a Conferência Semiótica é mais orientada a processo, apontando problemas e oportunidades para o aprimoramento no ambiente de trabalho da organização através do design e uso do sistema.

Em uma análise sobre o design do sistema Pokayoke, identificamos que a Conferência Semiótica (considerando todas as modificações propostas para os modelos ou protótipos durante todo o tempo do design) indicou um total de 39 problemas não redundantes. Dezenove problemas tiveram o foco principal no produto, problemas que foram descobertos enquanto os participantes discutiram soluções alternativas nas telas dos protótipos para refletir alterações nos diagramas de ontologia e normas associadas a eles. Por outro lado 20 problemas tiveram como foco principal a visão de processo; estes problemas foram descobertos, principalmente, durante discussões sobre o contexto organizacional e a sua relação com o sistema computacional.

Algumas atividades de avaliação são necessárias para engajar o grupo em um processo de significação sobre os elementos que envolvem o design do sistema com o objetivo de "informar" o design da interface. Desta maneira analisamos os problemas e identificamos um grupo especial de problemas orientados a processo que estavam fortemente relacionados ao processo de significação.

A principal característica destes problemas é que eles viabilizam a aprendizagem mútua, visto que eles resultam em possibilidades de aprimoramento no contexto organizacional, identificadas pelos trabalhadores durante o design do sistema, ou são

resultados da falta de compreensão das práticas de trabalho pelos designers do sistema (design aprendendo sobre o contexto). No primeiro caso os problemas identificados resultaram em modificações no sistema ou no contexto organizacional. Estas modificações são propostas pelos usuários com o objetivo de revisar ou criar novas práticas de trabalho. Esta é a parte do processo de significação pelo qual os usuários aprendem sobre as possibilidades trazidas pela tecnologia, através da avaliação de modelos e protótipos. O segundo caso, no qual os conceitos da prática de trabalho são mal compreendidos pelos designers, também é parte do processo de significação em que os designers aprendem sobre o contexto de trabalho.

Para ilustrar este processo nós apresentamos os problemas identificados durante o design do sistema Pokayoke que resultaram em discussões mais profundas sobre o contexto organizacional futuro e o design do sistema:

- "*Quem pode emitir um cinco passos?"* Somente os trabalhadores do departamento de qualidade até então podiam emitir um novo problema segundo o procedimento de "Cinco Passos". Após discussões, designers e o pessoal do departamento de qualidade chegaram ao consenso de que todos os departamentos deveriam emitir seus próprios "Cinco Passos". Também foi discutida a estratégia adotada para disseminar a cultura da resolução sistemática de problemas utilizando o procedimento de "Cinco Passos";
- "*O formulário de avaliação não é sempre preenchido"* (o formulário de avaliação apresenta detalhes do problema no momento em que ele ocorre). Os designers tinham entendido que os formulários de avaliação eram sempre preenchidos, mas em algumas situações específicas eles não eram preenchidos pelos trabalhadores. Uma discussão a respeito do contexto organizacional clarificou os designers a respeito deste problema;
- "*Os diagramas de Ishikawa, cinco porquês e as ferramentas de Brainstorming são utilizadas somente durante alguns passos específicos"*. Nos modelos e protótipos anteriores, estas ferramentas não foram associadas com passos específicos no processo de resolução de problemas. Os modelos e o sistema foram alterados para dar suporte a idéia de que estas ferramentas deveriam ser

utilizadas somente durante alguns passos específicos. Esta discussão corrigiu alguns aspectos mal interpretados pelos designers;

- *"Nós temos que definir os papéis para a pessoa que abre e para a que coordena o processo de resolução de problemas."* Durante a Conferência Semiótica, os trabalhadores entenderam que eles não tinham regras bem definidas em relação às responsabilidades no procedimento de "Cinco Passos". Eles enfatizaram a necessidade de defini-las o quanto antes. Todos os participantes discutiram as normas de responsabilidades durante alguns encontros. Eles estabeleceram nomes para cada papel desempenhado na resolução de problemas; alguém deveria ser o responsável por abrir o "Cinco Passos" (inclui-lo no sistema) e por conduzir o passo um e cinco, e outra pessoa deveria ser responsável pelas fases de correção (passos II, III, IV). Novas práticas foram adotadas na fábrica como um resultado desta discussão, mesmo antes que o sistema fosse utilizado em larga escala;
- *"O responsável pelo passo não é o responsável pelas ações."* Novamente uma discussão a respeito da responsabilidade, porém agora com foco em quem deveria ser o responsável pelas ações. O conceito de "responsável pelas ações" foi clarificado. Novas práticas foram adotadas na fábrica como resultado desta discussão, mesmo antes que o sistema fosse utilizado em larga escala;
- *"Nós temos diferentes versões de cinco passos para cada parte da fábrica."* A organização em que nós trabalhávamos possui três "plantas" divididas de acordo com a produção. Conforme mencionado anteriormente cada uma delas tinha um procedimento de "Cinco Passos" diferente; algumas diferenças apareceram durante a Conferência Semiótica. Um padrão comum foi definido para todas as "plantas";
- *"Após o diagrama de Ishikawa nós queremos fazer um cinco porquês."* O sistema até então não previa a ferramenta de "Cinco Porquês". Isso resultou em modificações no modelo do sistema que foi resultado de uma discussão a respeito da necessidade da ferramenta conhecida como "cinco porquês". Esta discussão resultou em uma melhor compreensão por parte dos designers do sistema, de como esta ferramenta é utilizada na prática;

- *"O responsável (quem emitiu o passo) é quem escolhe os trabalhadores que irão compor o time multifuncional."* Outra discussão sobre a responsabilidade foi conduzida com o objetivo de estabelecer normas de responsabilidades e deveres para o "responsável" e o "responsável pelo cinco passos" durante o processo de resolução;
- "*A definição do que deveria ser implementado para corrigir o problema ocorre no quarto passo*." Outro aspecto sobre as práticas de trabalho que havia sido mal compreendido pelos designers. Uma discussão a respeito do que deveria ser feito no terceiro e no quarto passos foi conduzida com o objetivo de clarificar as práticas de trabalho;
- *"O gerente de qualidade deverá verificar todos os cinco passos"* Os participantes entraram em uma discussão a respeito do papel do gerente da qualidade; após uma discussão foi detectado que ele deveria ser responsável por verificar todos os "Cinco Passos";
- "*Como fazer auditoria neste novo contexto."* Os trabalhadores perceberam que as regras para auditoria não existiam no modelo do sistema. Os trabalhadores discutiram como fazer esta tarefa utilizando o sistema e foram levantadas novas possibilidades para conduzir esta tarefa da maneira mais simples possível.

Conforme destacado, as discussões que surgiram durante a aplicação do SPaM, particularmente as que ocorreram durante a aplicação da Conferência Semiótica (que contribuiu com oito das doze situações exemplificadas acima), promoveu uma compreensão compartilhada dos signos presentes nas práticas de trabalho. Desta maneira, estas discussões contribuíram para a aprendizagem mútua durante o processo de design.

7.2.4 Uso do Sistema

Após os cinco ciclos de prototipação, o Pokayoke substituiu os procedimentos para "Cinco Passos" até então existentes na fábrica. Os trabalhadores que participaram do processo de design foram responsáveis pelo treinamento de outros usuários, evidenciando que a aprendizagem mútua ocorreu durante o processo de design utilizando o SPaM, uma

vez que eles conheciam o sistema e o ambiente suficientemente bem para treinar os outros, sem a necessidade de intervenções dos designers. A aprendizagem mútua aliada com o sentimento de autoria foi fundamental para a aceitação do sistema pelos usuários. Os trabalhadores expressaram várias vezes o sentimento de autoria do sistema durante o design do sistema e seu uso; eram comuns afirmações do tipo: ". . . nós definimos isto desta maneira para evitar . . .".

Após o sistema Pokayoke ter sido implantado na fábrica, houve um aumento considerável no número de trabalhadores que participam do "Cinco Passos". Contrapondo a versão em papel do "Cinco Passos", o sistema é utilizado por trabalhadores de toda a fábrica em diferentes funções e níveis hierárquicos. O sistema foi instalado em cerca de 40 computadores em diferentes pontos da fábrica. Após 10 meses em uso, o sistema contava com a participação de 59 trabalhadores na resolução de 390 problemas. Deste total 185 eram problemas já solucionados e 205 problemas encontravam-se em processo de resolução.

A avaliação da abordagem utilizada para o design da interface do sistema Pokayoke foi realizada através da análise de modificações requeridas pelos usuários durante seu uso. A implementação da parte dinâmica da interface do sistema Pokayoke foi restringida aos tipos de normas apresentadas anteriormente o que exclui normas que detalham por exemplo a inclusão de tarefas pré-definidas ou de novas tarefas obrigatórias a serem realizadas pelo usuário durante o processo. Desta maneira modificações nestes tipos de normas são consideradas modificações no código do sistema. Durante o período de teste, no qual o sistema foi utilizado por sete usuários finais, foi aplicada ao Pokayoke a técnica PHE, conforme proposta por Müller e outros (1998).

Um grupo de 7 usuários e 5 especialistas em interface propôs um total de 25 modificações no sistema durante a aplicação da PHE. Os usuários finais propuseram 14 modificações, das quais 9 referem-se a modificações de normas na parte dinâmica da interface, 3 referem-se a pequenos problemas (uma frase escrita de maneira errada, tamanho do botão, margem da impressão, etc) que podem ser facilmente corrigidos mas foi necessário modificação do código e 2 referem-se a modificações no *layout* da janela relativo à parte estática; consideramos estas modificações como modificações em código

pois levaram mais de um dia de trabalho para implementá-las (veja distribuição na Figura 7.21).

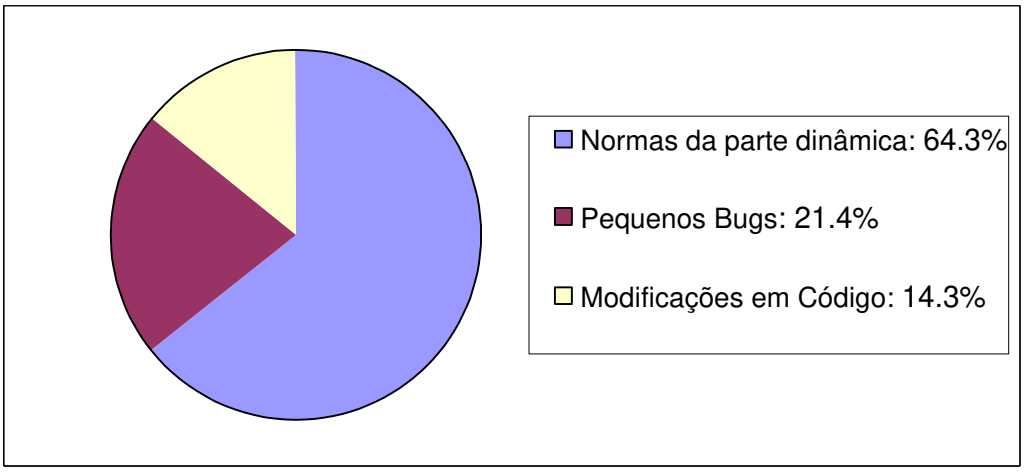


Figura 7.21 - Modificações propostas pelos usuários finais

Os especialistas em interface propuseram 11 modificações, 4 delas referem-se a normas especificadas na parte dinâmica do sistema, 5 a pequenos problemas de fácil correção mas que requerem modificação no código e 2 que requerem modificações na estrutura da tela da parte estática da interface, para estas modificações foi necessário mais de um dia de trabalho de implementação (veja distribuição na Figura 7.22).

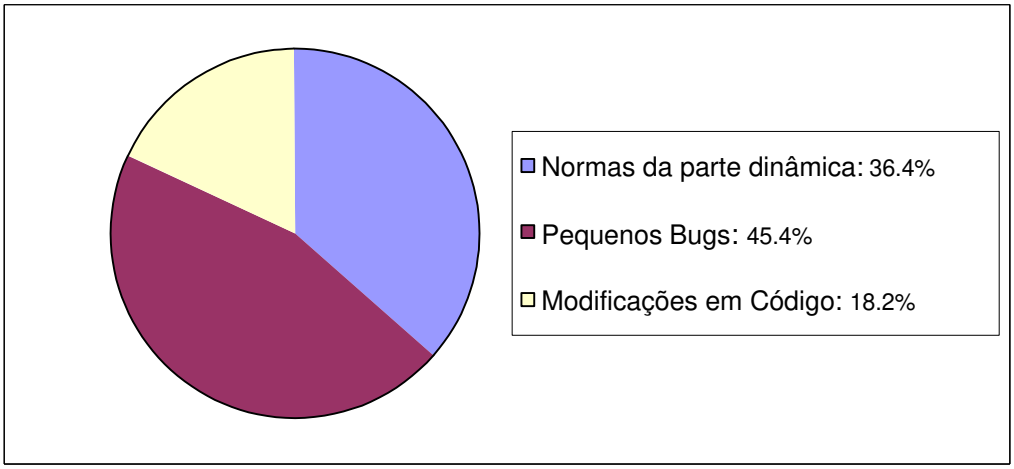


Figura 7.22 - Modificações propostas pelos usuários especialistas

Outras 3 modificações foram propostas durante os 10 meses de uso do Pokayoke em larga escala por 59 usuários, em substituição ao procedimento de “Cinco Passos” em

papel. Duas modificações referem-se à parte dinâmica do sistema e a outra refere a um pequeno problema facilmente corrigido.

Analisando o total de 28 modificações propostas para o sistema Pokayoke, conforme exibe a Figura 7.23, 54% referem-se à parte dinâmica, 32% à pequenas modificações da parte estática e 14% às modificações que requeriam mais de um dia de trabalho de implementação. Temos que considerar também que este número poderia ser mais favorável se tivéssemos especificado outras funcionalidades do sistema como normas, uma vez que o Pokayoke possui apenas uma versão limitada do NIBC. De maneira geral, a abordagem proposta utilizada no design e no uso do sistema Pokayoke mostrou sua capacidade em facilitar estas modificações.

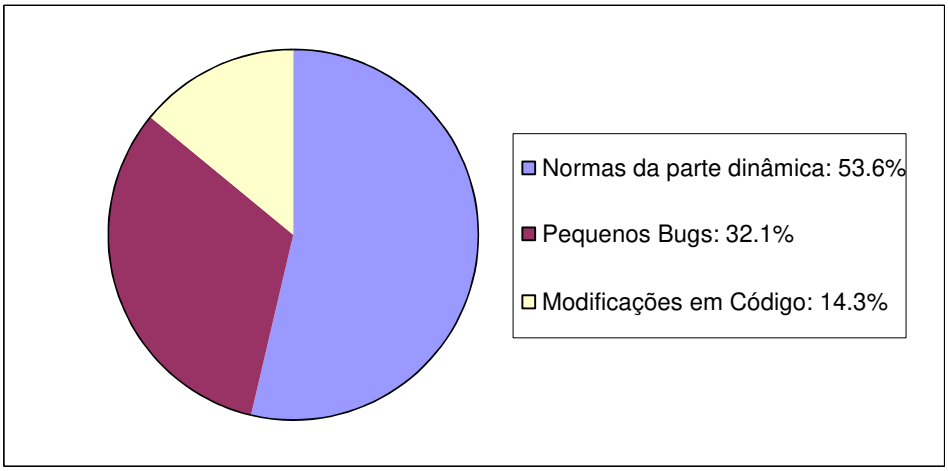


Figura 7.23 - Distribuição de todas as alterações propostas

7.3 Síntese e Considerações Finais do Capítulo

Neste capítulo foi apresentado o estudo de caso conduzido para explorar o uso do SPaM na prática. Foram descritos o contexto organizacional e o sistema Pokayoke que dá suporte a resolução de problemas em uma organização de manufatura que adota o paradigma de "produção enxuta".

O tempo total de desenvolvimento do sistema Pokayoke foi de 14 meses distribuídos em cinco ciclos de prototipação. Em cada ciclo de prototipação foram aplicadas várias técnicas de DP que produziram resultados relevantes a todas as fases do

SPaM. Estas técnicas foram aplicadas durante 15 visitas à organização, que totalizaram 52 horas distribuídas durante todos os ciclos de prototipação do sistema.

Os modelos da Análise Semântica e de Normas foram construídos com base nas técnicas de DP aplicadas. Cada uma das técnicas utilizadas contribuiu de maneira complementar dando suporte a tarefas em todas as fases da análise semântica e cobrindo aspectos de todos os níveis da "organizational onion". As técnicas de DP também foram fundamentais para capturar as regras que regiam o comportamento dos agentes na organização, uma vez que os trabalhadores puderam compartilhar e prover o *feedback* a respeito da compreensão do ambiente de trabalho.

Entre as técnicas aplicadas está a Conferência Semiótica (proposta nesta tese), explorada durante o primeiro, terceiro, quarto e quinto ciclos de prototipação do SPaM. Seus resultados foram importantes no início de cada iteração, com o objetivo de revisar conceitos do protótipo construído no ciclo anterior e redirecionar o design do próximo protótipo. Também em outros momentos quando era necessário revisar conceitos.

Em relação a aspectos do projeto do sistema Pokayoke, podemos dizer que os modelos iniciais de projeto construídos com base nos procedimentos propostos para serem utilizados durante o design de alto nível (passos 2.2, 2.3 e 2.4 do SPaM) foram importantes para as fases 3 e 4 do SPaM. O sistema Pokayoke foi construído em Java e utiliza RMI (Remote Method Invocation). Possui uma arquitetura dividida em quatro blocos principais: cliente, servidor de aplicação, servidor de dados e estação NBIC. O projeto e implementação da interface (passos 3.1 e 4.1 do SPaM) incluíram o detalhamento de como o sistema dará apoio ao comportamento humano, qual a representação gráfica adequada ao modelo detalhado da interface e também codificação das funcionalidades que permitiram a configuração dinâmica da interface via NBIC.

Analisando as atividades de avaliação aplicadas durante o design do sistema Pokayoke concluímos que algumas delas são necessárias para engajar o grupo em um processo de significação sobre os elementos que envolvem o design do sistema, com o objetivo de "informar" o design da interface. Identificamos um grupo especial de problemas orientados a processo que estão fortemente conectados ao processo de significação. A principal característica destes problemas é que eles promovem a aprendizagem mútua, visto que eles resultam em possibilidades de aprimoramento no

contexto organizacional, identificadas pelos trabalhadores durante o design do sistema (trabalhadores aprendendo sobre oportunidades trazidas pela tecnologia), ou são resultados da falta de compreensão das práticas de trabalho pelos designers do sistema (designers aprendendo sobre o contexto).

Após cinco ciclos de prototipação o Pokayoke substituiu os procedimentos para "Cinco Passos" até então existentes na fábrica. Após 10 meses em uso, o sistema contava com a participação de 59 trabalhadores na resolução de 390 problemas. A avaliação da abordagem utilizada para o design da interface do sistema Pokayoke foi realizada através da análise de modificações requeridas pelos usuários durante seu uso e de maneira geral a abordagem proposta utilizada no design e no uso do sistema Pokayoke, e mostrou-se viável e capaz de facilitar a implementação das modificações.

Capítulo 8

Conclusão

A pesquisa relatada nesta tese abordou o desenvolvimento de sistemas no contexto organizacional. Conforme destacado anteriormente, a abordagem utilizada na construção do sistema computacional é fundamental para a sua adequação ao contexto e também para maximizar os benefícios de seu uso na organização. Nesta tese, argumentamos que o sistema em desenvolvimento deve ser parte de um contexto organizacional futuro. Nesta visão devemos não só modelar o sistema, mas também suas interações com o contexto social em que ele estará inserido. O esforço de modelagem deve ser concentrado em maximizar o valor do sistema para a organização e não apenas nas suas características técnicas.

A importância dos aspectos presentes no contexto social para o design de sistemas é ainda mais marcante quando falamos de sistemas para suporte ao trabalho cooperativo. Pesquisadores da área de CSCW constantemente deparam-se com desafios que vão além de aspectos puramente técnicos. Para lidar com estes desafios existem várias propostas, principalmente de caráter interdisciplinar. Nesta tese lidamos com dois aspectos que têm impacto direto no design de sistemas para o contexto organizacional: (1) *A participação do trabalhador no design do sistema e no desenvolvimento de um novo modelo organizacional* e (2) *A representação do sistema e da organização por meio de um mesmo método.*

A solução proposta foi construída pela integração, articulação e expansão de conceitos e técnicas de duas áreas de pesquisa principais: o Design Participativo e a Semiótica Organizacional. Conforme destacado no decorrer deste trabalho, no Design

Participativo o usuário é considerado parte ativa do desenvolvimento do sistema, o que possibilita que designers e trabalhadores discutam e construam juntos um novo contexto organizacional. Por outro lado, a Semiótica Organizacional surge como um veículo facilitador para esta interação, possibilitando o entendimento, a discussão e a modelagem de conceitos da organização através de uma visão subjetivista, que está mais alinhada à compreensão dos aspectos humanos presentes na organização.

Para possibilitar que ambas as áreas contribuam em conjunto ao design cooperativo, foram analisados os seus conceitos e técnicas e identificados pontos em que estas áreas podem complementar-se. Nesta tese, a abordagem de design para a construção de um novo contexto organizacional é delineada pelo conceito de "aprendizagem mútua", no qual os designers aprendem sobre o domínio e os usuários sobre as oportunidades trazidas pela tecnologia.

O SPaM foi proposto para integrar as técnicas de DP e SO em um método de design único. Ele pode ser visto como um método que inclui tanto técnicas de DP quanto de SO e visa promover a "aprendizagem mútua" durante todo o desenvolvimento do sistema. Este método define os ciclos, fases e passos que direcionam o desenvolvimento e auxiliam os designers a articularem as técnicas para produzir o sistema. No SPaM um ciclo de prototipação está dividido em quatro fases. Como ponto de partida temos a discussão e a modelagem da organização futura, que irá incluir o sistema. Em seguida são definidos os requisitos e modelos de alto nível para o sistema. A partir destes modelos são construídos os diagramas de projeto e o protótipo do sistema. A participação do usuário em todas as fases através da interação e comunicação facilitada pelos modelos da SO garante que eles (usuários) tenham a possibilidade de participar das decisões que influenciarão os seus métodos de trabalho em um novo contexto organizacional.

Como parte do SPaM foi proposta a Conferência Semiótica, uma nova técnica de DP que permite que os usuários participem diretamente e democraticamente da construção dos modelos propostos na SO. Ela também possibilita que eles articulem estes modelos em conjunto com os designers para discutir o seu contexto de trabalho, engajados em processo de aprendizagem. A Conferência Semiótica visa balancear os aspectos *orientados a produto* (que focam o artefato de software) e os aspectos *orientados a processo* (que focam o processo de trabalho humano) no SPaM.

No SPaM, diagramas da SO são construídos para representar o sistema e o contexto organizacional no qual ele estará inserido. Para projetar e implementar o sistema a partir destes modelos, o SPaM propõe uma arquitetura em três níveis: dados, negócio (aplicação) e interface. Para o nível de dados foi proposta uma simplificação da abordagem apresentada por Liu; para o nível de negócio foi proposto um procedimento composto de passos e regras heurísticas, como uma maneira quase direta para construir diagramas iniciais de projeto (UML) a partir do modelo de ontologia; e para o nível de interface foi proposta uma arquitetura que explora a característica da análise semântica de representar conceitos mais estáveis do que os presentes na análise de normas. Esta arquitetura possibilita modificar a interface e mantê-la consistente com as mudanças ocorridas na organização após o desenvolvimento do sistema através de alterações na especificação das normas, desta maneira facilitando e barateando a manutenção do sistema.

Para instanciar o SPaM foi conduzido um estudo de caso em uma organização fabril através de uma parceria entre nossa universidade e esta organização. Ela adota o sistema de produção "enxuta" onde os funcionários devem ser flexíveis, suas habilidades devem ir além de suas funções particulares, devem possuir senso crítico e um bom desempenho no trabalho em grupo, sendo assim considerados elementos inteligentes necessários para aprimorar a qualidade do processo de produção. Como resultado deste estudo de caso foi construído o sistema Pokayoke, que é baseado em um procedimento para a resolução de problemas conhecido como "Cinco Passos". A modelagem deste sistema visou diminuir as dificuldades da aplicação do "Cinco Passos" e aprimorar o processo de resolução de problema na fábrica.

O tempo total de desenvolvimento do sistema Pokayoke foi de 14 meses distribuídos em cinco ciclos de prototipação. Atividades participativas (incluindo a Conferência Semiótica) foram desenvolvidas durante todos os ciclos de prototipação em visitas periódicas à organização. Cada uma destas atividades contribuiu de maneira complementar para a construção dos modelos da análise semântica e de normas. Através destas técnicas, pudemos melhor compreender e modelar os signos que representam a organização em função da participação dos trabalhadores que fazem parte dela.

Pela análise do estudo de caso foram identificados aspectos que evidenciam o processo de "aprendizagem mútua" no SPaM. Para ilustrar este processo foram apresentados doze problemas identificados (oito resultantes da aplicação da Conferência Semiótica) durante o design do sistema Pokayoke que resultaram em discussões mais profundas sobre o contexto organizacional futuro e o design do sistema, onde designers e usuários estiveram engajados em um processo de aprendizagem.

O sistema Pokayoke substituiu o procedimento de "Cinco Passos" em papel. A sua aceitação por parte do trabalhador foi facilitada pela participação de seus representantes no design do sistema. Outro elemento que veio a ilustrar a aprendizagem mútua ocorrida refere-se ao treinamento dos usuários para o uso do sistema em larga escala; este foi realizado pelos próprios trabalhadores que possuíam mais conhecimento sobre a estrutura do sistema. Números também evidenciam o decréscimo da necessidade de mudanças no código do sistema, através do uso do NBIC, facilitando assim a sua manutenção.

8.1 Contribuições da Pesquisa

Para responder a principal questão de pesquisa apresentada no início deste trabalho: "Por que e como a Semiótica Organizacional e o Design Participativo poderiam juntos contribuir para o design de sistemas de suporte ao trabalho cooperativo no contexto organizacional?", esta tese lidou com vários desafios presentes no desenvolvimento de sistemas de CSCW. Construir novas abordagens para lidar com estes desafios vem sendo objeto de pesquisa na área de CSCW. O DP e a SO juntos trazem uma nova possibilidade para o design de sistemas de suporte ao trabalho cooperativo ainda inexplorada em outros projetos de pesquisa e contribui diretamente para pesquisa nesta área. Considerando os oito desafios apresentados por Grudin (1994) (destacados no capítulo 2), apresentamos como a abordagem proposta explora com cada um eles:

- *A disparidade entre quem faz o trabalho e quem tem o benefício*. Na abordagem proposta, os futuros usuários participam da decisão sobre as opções de design que possam afetar o trabalho de cada membro do grupo. Ainda que, mesmo com a participação do trabalhador existam disparidades inevitáveis, estas são tornadas visíveis durante o desenvolvimento do sistema

por meio de uma definição clara do contexto organizacional futuro (e não apenas do software). A abordagem proposta nesta tese permite que usuários discutam as normas que definirão as responsabilidades, os deveres e obrigações de cada trabalhador durante o uso do sistema de CSCW;

- *Massa crítica e o Problema do Dilema do Prisioneiro*. Por meio da Conferência Semiótica e demais técnicas de DP, o SPaM traz trabalhadores de diferentes funções e posições hierárquicas para dentro do processo de design do sistema. Desta maneira estes trabalhadores representam visões e interesses da organização, o que facilita obter massa crítica para viabilizar o uso do sistema. Além disso as responsabilidades sobre o interesse coletivo durante o uso do sistema são definidas e modeladas, o que dificulta que os membros do grupo de trabalho possam agir somente em função de seus próprios interesses;
- *Fatores Sociais, Políticos e Motivacionais.* O DP permite trazer para a pauta de discussão durante o desenvolvimento do sistema fatores sociais, políticos e motivacionais que irão influenciar o seu uso. Por outro lado, a SO facilita a interação, comunicação e a modelagem destes fatores;
- *Tratamento de exceções nos grupos de trabalho.* O DP permite que os trabalhadores exponham estas exceções. De maneira complementar a SO permite, por meio da análise semântica, identificar o que é invariante do comportamento do grupo e a análise de normas permite modelar as normas sociais que não necessariamente devem ser obedecidas pelos agentes. Assim, exceções que violam estas normas podem ser especificadas e tratadas adequadamente pelo sistema via especificação de outras normas que abordam estas exceções;
- *O Design para características não freqüentemente utilizadas.* Através do DP e SO é possível identificar as características não freqüentemente utilizadas uma vez que os trabalhadores que as conhecem participam do desenvolvimento. Também é possível discutir em grupo o papel destas características em um contexto organizacional futuro, o que permite definir quando e como elas são utilizadas no processo de trabalho como um todo;

- *A dificuldade subestimada para avaliar groupware.* Usando DP os designers podem avaliar junto com os usuários o dinamismo social, motivacional, econômico e político dos grupos e através da SO é possível representar e compreender melhor estes aspectos;
- *A quebra da tomada de decisão intuitiva.* O DP parte do princípio de que a intuição do grupo trará melhores resultados sobre as decisões de design do que as baseadas na intuição única dos designers. Incluir o trabalhador no processo de design possibilita trazer parâmetros até então não visíveis pelos designers que tinham que tomar decisões sozinhos, com um conhecimento limitado sobre o contexto de trabalho. Decisões sobre aspectos de design de sistemas de CSCW normalmente envolvem discussões sobre o domínio e seus impactos diretos nos métodos de trabalho. A SO junto com o DP permite que o grupo modele e visualize um suposto contexto organizacional futuro delineando as discussões que precedem as decisões de design;
- *O gerenciamento da aceitação.* Na abordagem proposta nesta tese a introdução do sistema no ambiente de trabalho é facilitada, uma vez que a aprendizagem mútua ocorrida durante o desenvolvimento do sistema possibilita que os próprios trabalhadores disseminem o sistema na organização.

A partir da questão principal foram derivadas outras questões não menos importantes para a pesquisa em CSCW, DP e SO. A primeira delas é: "Que aspectos trabalhados pelo DP e SO são relevantes para a construção de sistemas cooperativos?" Esta questão foi abordada no capítulo 2 e o resultado desta análise contribui para criar uma definição mais clara sobre a importância do contexto social no desenvolvimento de sistemas para suporte ao trabalho cooperativo.

A segunda questão é: "Por que é possível integrar conceitos de DP e SO?" Na abordagem proposta DP e SO são complementares e juntas podem lidar com vários aspectos do desenvolvimento de sistemas computacionais em organizações. Esta abordagem representou o primeiro passo necessário para o desenvolvimento de pesquisas futuras que envolvam ambas as áreas.

A terceira questão é: “Como promover o design cooperativo através da participação e significação?” Esta questão levanta um ponto crucial que é a busca pelo entendimento do desenvolvimento de software como um processo. Nesta tese o design cooperativo é entendido através do conceito de “aprendizagem mútua” (Kyng, 1991) onde a participação e a significação entram como elementos necessários para que ela ocorra. Para tanto é estabelecido o papel do DP, SO e do Departamento de Recursos Humanos como agentes facilitadores na promoção da aprendizagem mútua.

A quarta questão é: “Como desenvolver um método de design baseado em DP e SO?” Para responder a esta questão foi criado o SPaM, que traz várias contribuições para a pesquisa em DP e SO. O SPaM define ciclos, fases e passos que possibilitam trabalhar as técnicas de ambas as áreas. Para construí-lo foi necessário pesquisar aspectos referentes ao processo de significação ainda não explorados em DP e pesquisar novos aspectos referentes à participação até então não trabalhados na área de SO. Através deste trabalho criamos bases que possibilitam ampliar a visão de assuntos abordados em ambas as áreas.

A quinta questão é: “Como possibilitar a participação dos usuários na modelagem proposta pela SO?” Para tanto foi proposta a Conferência Semiótica; ela definitivamente traz o usuário para dentro da modelagem proposta na SO. Embora existisse a preocupação em trazer os signos do contexto organizacional para os modelos, ainda não existia uma maneira que possibilitasse que os próprios trabalhadores articulassem e participassem da modelagem destes signos. Os significados modelados são comuns aos designers e trabalhadores, uma vez que são clarificados ou construídos pela participação direta do usuário no processo de design.

A sexta questão é: “Como derivar modelos para implementar o sistema e fazer manutenção a partir dos resultados do uso de DP e SO ?” Criar modelos de projeto a partir dos modelos da SO e transpor os aspectos modelados para a implementação do sistema são desafios para os pesquisadores desta área. Embora já houvesse uma solução para a modelagem de banco de dados baseados nos aspectos presentes na análise semântica e de normas, não havia um processo que permitisse construir modelos da UML informados pela SO e tão pouco especificar interfaces do usuário que incorporassem e aproveitassem os aspectos trabalhados pela SO. Neste sentido esta tese vai um passo além

do proposto inicialmente, pois é apresentado um procedimento composto de passos e regras heurísticas para a construção de um primeiro diagrama de classes baseado nos resultados da análise semântica. Além disso, é proposta uma arquitetura para a construção de interfaces do usuário que propiciam a manutenção mais ágil do sistema, baseada em características da análise semântica e de normas.

Em termos gerais, acreditamos que esta tese contribui para a pesquisa em desenvolvimento de software ao incluir em um só método aspectos referentes à participação e à significação que até então não haviam sido explorados em conjunto.

8.2 Trabalhos Futuros

Neste trabalho foram exploradas principalmente a Análise Semântica e a Análise de Normas em conjunto com sete técnicas de DP escolhidas de acordo com a sua adequação a diferentes fases do desenvolvimento do sistema, a facilidade de aplicá-las no contexto de trabalho objeto do estudo de caso, a possibilidade de serem articuladas em conjunto com as técnicas de SO, seu potencial para estimular a colaboração e a reflexão durante o design. Como trabalho futuro é possível explorar outras técnicas de DP que possam de alguma forma complementar a proposta e se adequarem ao SPaM. Também pode-se investigar a inclusão de outras técnicas da SO no SPaM (do MEASUR ou propostas por outros pesquisadores da Semiótica Organizacional), como por exemplo o PAM (*Problem Articulation Method*).

Embora o SPaM tenha sido construído para ser utilizado em vários escopos, o seu uso na prática ainda não foi explorado além do estudo de caso realizado nesta tese. Como trabalho futuro pode ser investigado o uso de SPaM e da Conferência Semiótica em outros contextos além dos baseados em sistemas de CSCW. Isso permitirá um refinamento da proposta e a sua adequação a um escopo mais amplo de problemas. Outra possibilidade seria explorar como o SPaM (inteiro ou partes dele) poderia ser utilizado em conjunto com processos de desenvolvimento de software, como por exemplo o Processo Unificado ou o Extreme Programming, possibilitando assim abordar um conjunto mais amplo de aspectos ligados ao desenvolvimento de software normalmente explorados na área de Engenharia de Software.

Em relação ao NBIC pode-se investigar como construir uma ferramenta (em adição ao gerenciador de Normas e Ações) que permita que os próprios usuários possam alterar a interface do sistema através de mudanças significativas nas normas que a especificam. Isso incluirá pesquisa na área de *tailoring*.

Em relação ao Pokayoke existe a idéia de transformar este sistema em um sistema de software livre. Isso possibilitará estender os seus benefícios a um número maior de organizações dentro do âmbito nacional. Estas organizações normalmente importam produtos de suporte à comunicação que custam milhares de dólares e são incapazes de auxiliar de maneira eficaz os procedimentos de qualidade específicos (como o "Cinco Passos") das organizações.

Para transformar o sistema Pokayoke em um sistema de software livre é necessário reformular parte de sua estrutura, identificar os aspectos comuns à resolução de problemas em diversas organizações, possibilitar a adaptação do sistema a diferentes contextos e prover mecanismos de distribuição, uma vez que cada organização possui aspectos específicos em seu processo de resolução de problemas. Organizações poderiam obter o software na Internet e implantando-o e adaptando-o para sua realidade utilizando o NBIC.

8.3 Considerações Finais

Em uma análise final posso dizer que este trabalho contribuiu para meu aprimoramento pessoal como pesquisador, educador e profissional de informática. Com este trabalho pude ter uma visão mais clara de que construímos sistemas para pessoas, uma vez que, apesar dos aspectos técnicos serem decisivos no desenvolvimento de sistemas computacionais não podemos esquecer que o seu propósito final é auxiliar as pessoas a desenvolverem e aprimorarem tarefas em seu dia-a-dia. A interação direta com os usuários durante o desenvolvimento do sistema Pokayoke me possibilitou enxergar melhor os efeitos que um sistema computacional pode gerar em um ambiente de trabalho, tanto positivos quanto negativos.

Com este trabalho também pude aprender que o sistema é desenvolvido por pessoas e para pessoas e desta maneira temos que utilizar técnicas construídas com o objetivo de

compreender seu uso em adição às técnicas voltadas aos aspectos técnicos presentes no desenvolvimento do sistema. Ao final, digo de maneira convicta que é um grande desafio para nós, profissionais de sistemas de informação, trazer os aspectos humanos para dentro do design do sistema computacional de modo a satisfazer os anseios dos usuários em relação à tecnologia que construímos.

Referências

Adelson, B. (1999) Developing strategic alliances: A framework for collaborative negotiation in design, *Research in engineering design-theory applications and concurrent engineering*, Springer-Verlang, 233-144.

Andersen, P. B. (2001) What semiotics can and cannot do for HCI, *Knowledge-Based Systems,* 14, 419-424.

Ades, Y. M. (1989) *NAMAT System User Manual*. SIS project, University of Qatar.

Andriole, S. J. (1992) *Rapid application prototyping: the storyboard approach to user requirements analysis* (2nd ed.) Boston: QED.

Areces C., de Rijke, M. & de Nivelle (2001), H. Resolution in Modal, Description and Hybrid Logic. *Journal of Logic and Computation*, Vol 11, Issue 5.

Baranauskas, M. C. C. Borges, M. A. F., Borges, E. L. (1997) Learning by creating models: a computer-based environment for industrial application. Em *Artificial intelligence in education - knowledge and media in learning systems*. B. DuBoulay e R.Mizoguchi (eds.) Frontiers in Artificial Intelligence and Applications. IOS Press, The Netherlands, 426-433.

Baranauskas, M. C. C. (1999) Learning about manufacture process control through the target game. *International Journal of Continuing Engineering Education And Life-long Learning*, UK, v.9, n.3/4, p.210-221.

Baranauskas, M.C.C., Gomes Neto, N., Borges, M.A F. (2000) Gaming at work: a learning environment for synchronized manufacturing. *Computer Applications in Engineering Education Journal*, Wiley & Sons, Inc.

Baranauskas, M.C.C., Gramacho, N., Borges, M.A.F. (2001) Learning at Work through a Multi-User Synchronous Simulation Game. Em *International Journal of Continuing Engineering Education and Life-Long Learning*, UNESCO/Inderscience Enterprises, v. 11, N. 3.

Baranauskas, M. C. C., Bonacin, R. e Liu K., (2002) Participation and Signification: Towards Cooperative System Design. In *Proceedings of V Symposium on Human Factors in Computer Systems*, Fortaleza, 3-14.

Baranauskas, M. C. C., Liu, K. e Chong, S. (2002b) Website Interfaces as Representamen of Organisational Behaviour, Em *Proceedings of 5th International Workshop on Organisational Semiotics*.

Bannon, L. J. e Schmidt, K. (1989) CSCW: Four Characters in Search of a Context. In Proc. First European Conf. on CSCW, Gatwick, UK, Sept. 1989. (Reprinted in J. Bowers & S. Benford (1991) (Eds.) *Studies in Computer Supported Cooperative Work: Theory, Practice and Design,* 3-16.

Bannon, L. J. (1992) Discovering CSCW, in Proceedings of *15th Information Systems Research in Scandinavia*, 507-520.

Bannon, L. J. (1993) CSCW: An Initial Exploration. In *Scandinavian Journal of Information Systems*, 5, 3-24.

Bannon, L. J. e Hughes J. A. (1993) The context of CSCW, In K. Schmidt (Ed.) *Developing CSCW Systems: Design Concepts*, Report of COST14 "CoTech" Working Group4 1991-1992, 9-36.

Barr, P., Biddle, R. e Noble, J. (2003) A semiotic model of User-Interface Metaphor. Em *Proceedings of 6th International Workshop on Organisational Semiotics,* 29-56.

Barthes, R. (1957): *Mythologies*. New York: Hill & Wang

Benyon, D. (1994) A funcional model of interacting systems: a semiotic approach. In Connolly, J. H. and Edmonds, E. A. (eds), *CSCW and Artificial Intelligence*, Springer-Verlag, London.

Berg M, (1998) The politics of technology: On bringing social theory into technological design. *Science Technology & Human Values*, Sage Publications, 456-490.

Boehm B.W. (1975). The High Cost of Software, in Horowitz E., *Practical Strategies For Developing Large Software Systems*, Addison Wesley.

Bonacin, R., (2002) *Um modelo para o desenvolvimento de sistemas para suporte a cooperação e tomada de decisão fundamentado em Design Participativo e Semiótica Organizacional,* Proposta de Tese de doutorado, Instituto de Computação - Unicamp.

Bonacin, R. and Baranauskas, M. C. C. (2003a), Designing Towards Supporting and Improving co-operative Organisational Work Practices, in *5th International Conference on Enterprise Information Systems*, Angers - France, 233-238.

Bonacin, R. e Baranauskas, M. C. C (2003b)., Semiotic Conference: Work Signs and Participatory Design, in *Proceedings of 10th International Conference on Human - Computer Interaction*, Crete - Greece, 38-43.

Bonacin, R., Baranauskas, M. C. C. and Cecilia, R. M. (2003a) Designing and Learning: joining the concepts in work practices. In Paola Forcheri and Alfonso Quarati (eds.), *Journal of International Forum of Educational Technology & Society and IEEE Learning Technology Task Force*, v. 6, n. 1, 2003, 3-8.

Bonacin, R., Baranauskas, M. C. C. e Liu, K. (2003b) Interface Design for the Changing Organisation: an Organisational Semiotics approach, Em *Proceedings of 6th International Workshop on Organisational Semiotics,* Reading - UK, 97-112.

Bonacin, R., Baranauskas, M. C. C. e Liu, K. (2004) Interface Design for the Changing Organisation: an Organisational Semiotics approach, Em K. Liu (ed.), *Virtual, Distributed and Flexible Organisations*, Kluwer Academic Publishers, a ser publicado.

Bonacin, R. e Silva, A. (2004) Design Participativo: o desenvolvimento participativo em projetos de P&D, Em M. Cecilia C. Baranauskas (ed.) *Lidando com a empresa*, A ser publicado.

Booch G. (1999) *Software Development Best Practices*, in The Rational Unified Process: an introduction, P. Kruchten, Addison-Wesley, 3-16.

Borges, E. L. (1997) *Design de um ambiente computacional de modelagem e simulação para formação de pessoal na indústria.* Dissertação (Ciência da Computação) - Universidade Estadual de Campinas.

Borges, M. A. F., Baranauskas, M. C. C. (1998) A user-centred approach to the design of an expert system for training. Em *British Journal of Educational Technology*, u K, v.29, n.1, 25-34.

Borges, M.A.F. (2000) *Repensando o design de ferramentas para o aprendizado colaborativo mediado pelo computador.* Proposta de tese de Doutorado, IC-Unicamp.

Borges, M.A.F., Baranauskas, M.C.C. (2000) Preparing and evaluating a collaborative learning environment: using Dice Game in a manufacturing company, *ICECE2000 International Conference on Engineering and Computer Education*.

Borges, M.A.F., Baranauskas, M.C.C.. (2002a) Analysing Influences of Context in a Game-mediated Collaboration. In *Proceedings of V Symposium on Human Factors in Computer Systems*, Fortaleza, Ceará, 129-140.

Borges, M.A.F., Baranauskas, M.C.C.. (2002b) CollabSS Sistema para Apoio on-line à Colaboração. In *Proceedings of V Symposium on Human Factors in Computer Systems*, Fortaleza, Ceará, 355-358.

Bourges-Waldegg P. e Scrivener S.A.R. (1998) Meaning, the central issue in cross-cultural HCI design, *Interacting with computers*, Elseview Science, 287-309.

Braa, K. (1996) Influencing qualities of informantion systems – Future challenges for participatory design in PDC´96 *Proceedings of the Participatory Design Conference*, 163-172.

Büscher, M., Gill, S., Mogensen, P. e Shapiro, D. (2001) Landscapes of Practice: Bricolage as a Method for Situated Design, *Computer Supported Cooperative Work*, Kluwer Academic Publishers, 10, 1–28.

Chandler, D. (2001) *Semiotics: The Basics*, Brunner-Routledge. Versão online acessada em 17/04/2004: *http://www.aber.ac.uk/media/Documents/S4B/sem06.html*

Clement, A. e Van den Besselar (1993) *A Retrospective Look at PD Projects.* In M. Müller e S. Kuhn (eds.). *Participatory Design: Special Issue of the Communications of the ACM,* vol. 36, no. 4, pp.29-39.

Coleman, D. (1995) *Groupware technology and applications: na overview of groupware*. Em D. Coleman (ed.) Groupware: technology and applications, Pentice Hall, 3-4.

Connolly, J.H., Phillips, I.W. (2001). User-System Interface Design: na Organisational Semiotic Perspective. Em *Proceedings of IFIP WG8.1 Working Conference on Organisational Semiotics*, Montreal, QC.

Chebabi, R. Z. (2002) *A abordagem construtivista no desenvolvimento de sistemas em organizações de trabalho*. Proposta de Dissertação de Mestrado, Instituto de Computação - Unicamp.

Cunningham, D. J. (1992) Beyond educational psychology: steps toward an educational semiotics. *Educational Psychology Review*, vol. 4, p. 164-194.

Deming, W. E. (1992) *Out of Crisis*. The MIT press.

Doerr M. (1997) Reference information acquisition and coordination, *Proceeding of the asis annual meeting*, Information Today, 295-312.

Dourish, P. (1996) *Open implementation and flexibility in CSCW toolkits*. Ph.D. Thesis, University College London, ftp://cs.ucl.ac.uk/darpa/jpd/dourish-thesis.ps.gz.

Eco, U. (1976) *Tratado geral de Semiótica*. Editora Perspectiva.

Ehn, P., e Sanberg, A. (1979) Management Control and Wage Earner Power (Foretagsstyrning och Lontagarmakt). Falkoping: Prisma.

Ehn, P. (1993) Scandinavian Design: On Participation and Skill in Schuler, Douglas and Aki, Namioka (Eds.) *Participatory Design: Principles and Practices*, Hillsdale, NJ: Erlbaum, 41-77.

Fischer, G. , Lindstaedt, S., Ostwald, J., Stolze, M., Sumner, T., Zimmermann, B. (1995) From Domain Modeling to Collaborative Domain Construction. In *ACM Proceedings of DIS'95 Symposium on Designing Interactive Systems: Processes, Practices, Methods & Techniques*, 75-85.

Filipe J. B. L. (2000) *Normative Organisational Modelling using Intelligent Multi-Agent Systems*, Ph.D. Thesis, Staffordshire University.

Finholt T.A., Teasley S.D. (1998) Psychology - The need for psychology in research on computer-supported cooperative work, *Social science computer review*, Sage Publications, 40-52.

Fiske, J. (1990) *Introduction to Communication Studies*. Routledge, London.

Floyd, C. (1987). Outline of a paradigm change in software engineering. In G. Bjerkners, P. Ehn, & M. Kyng (Eds), *Computers and democracy: A Scandinavian challenge.* Brookfield VT: Gower.

Friedman-Hill, E. J. (2001) *Jess, The Expert System Shell for the Java Platform*, SAND98-8206, Sandia National Laboratories, Livermore, CA, http://herzberg.ca.sandia.gov/jess

Gärtner, J. e Wagner, I. (1996) Systems as Intermediaries Political Frameworks of Design & Participation, In *Human-Computer Interaction*. 11: 187-214

Greif, I. (1988) *Computer-Supported Cooperative Work: A Book of Readings.* San Mateo, CA: Morgan Kaufmann.

Gibson,J. J., (1968) *The Ecological Approach to Visual Perception.* Houghton Miffin Company, Boston, Massachusetts, 127-143.

Gonzalez, R. (1997) Hypermedia data modeling, coding and semiotics. *Proceeding of the FEEE*, 85(7), p. 1111-1140.

Goré, R. P. (1992) *Cut-free Sequent and Tableau Systems for Propositional Normal Modal Logics*, Phd. thesis University of Cambridge.

Grudin, J. (1994) Groupware and Social Dynamics: Eight Challenges for Developers in *Communications of the ACM*, 37, 1, 92-105.

Gudwin, R.R. e Gomide, F.AC., An Approach to Computational Semiotics, In *Intelligent Systems and Semiotics - A Learning Perspective*, Gaithersburg, USA, (1997), p. 467-470

Hayes N. e Walsham G. (2000) *Competing interpretations of computer-supported cooperative work in organizational contexts*, Organization, Sage Publications, 49-67.

Heaton L. (1998) Talking heads vs. virtual workspaces: a comparison of design across cultures, *Journal of information technology*, Routledge, 259-272.

Honeycutt L. (1998) Linguistic concepts and methods in CSCW by Connolly, *Computation Linguistic*, MIT press, 324-327.

Jacobson, I., Booch, G. B. e Rumbaugh, J., (1999) *The Unified Software Development Process.* Addison-Wesley.

Kahler, H, Morch, A., Stiemerling, O. e Wulf, V. (2000) Special Issue on Tailorable Systems and Cooperative Work, *Computer Supported Cooperative Work*, Kluwer Academic Publishers, v. 9, Issue 1.

Kensing, F. (1983) The Trade Unions Influence on Technological Change. In *System Design For, With and By the Users*, U. Briefs et al. (eds.). North Holland.

Kensing F., e Blomberg J. (1998) Participapory design: issues and concerns. *Computer Supported Cooperative Work*, Special issue on Participatory Design, Jeanette Blomberg e Finn Kensing (eds.), 7, 167-185.

Klein, H. K. (1996) Preface: the pontential contribution of semiotics and systems theory to the continuing evolution of information systems research. In Holmqvist, B., Andersen, P.B., Klen, H. e Posner, R. (eds). *Signs of Work: Semiosis and Information Processing in Organisations.* Walter de Gruyter, Berlin, p. v-viii.

Kruchten, P., (1999) *The Rational Unified Process*. Addison-Wesley.

Kuhn, S. (1996) Design for People at Work. In T.A. Winograd (ed) *Bringing Design to Software*. Addison Wesley.

Kyng, M. (1991) *Designing for Cooperatin: Cooperating in Design. Commun*. ACM 34,12, 65-73.

Lave, J. Wenger, E. (1990) *Situated Learning: Legitimate Peripheral Participation*. IRL report 90-0013, Palo Alto, Calif.: Institute for Research on Learning.

Liu, K. e Dix, A. (1997) Norm governed agents in CSCW. *The First International Workshop on Computational Semiotics*. Paris. University of De Vince.

Liu, K. e Gao, Z. L. (1997) A problem articulation method for information systems: planning for responsive information management infrastructure. In Avison, D. (ed). *Key Issues in Information Systems.* McGraw-Hill, Maidenhead, Berkshire, p. 353-362.

Liu, K., Stamper R. K., and Huang, K. (1997) A morphology for re-engineering competitive business. *Legacy to client/server*, A. Booth (ed.), Unicom Press.

Liu K. (2000) *Semiotics in information systems engineering*, Cambridge University Press.

Liu, K. and Xie, Z., (2003) *Applying Semantic Analysis: a user's view*, Research report EPSRC-SEDITA project.

Lyytinen K. (1989) *Computer-Supported Co-operative Work (CSCW) - Issues and challenges.* Technical Report, Department of Computer Science, University of Jyväskylä, Finland.

Manderson R. and Layzell P. (1999) Software change control for maintenance: some experiences of moving from black art to black box, *In Fifth Workshop on Empirical Studies of Software Maintenance*.

Mahemoff, M. J. & Johnston, L. J. (1998). Principles for a Usability-Oriented Pattern Language In Calder, P. & Thomas, B. (Eds.), *OZCHI '98 Proceedings*, 32-139.

Mazzone, J. (1993) O Sistema Enxuto e a Educação no Brasil. Em *Computadores e Conhecimento: Repensando a Educação*, J.A.Valente (ed.), Gráfica Central da Unicamp, 247-312.

Morch A. (1995) Three Levels of End-User Tailoring: Customization, Integration and Extension. *Third Decennial Aarhus Conference*, 41-51.

Morch A. e Mehandjiev N. (2000) Tailoring as Collaboration: the Mediating Role of Multiple Representations and Application Units. Kahler, H, Morch, A., Stiemerling, O., Wulf, V. (Eds), Special Issue on Tailorable Systems and Cooperative Work, *Computer Supported Cooperative Work*, Kluwer, v. 9, Issue 1.

Morris, C. W. (1938) Foundations of the theory of signs, *International Encyclopedia of Unified Science*, University of Chicago Press, 1 (2).

Monplaisir, L. (1999) An integrated CSCW architecture for integrated product process design and development. *Robotics and computer-integrated manufacturing*. Pergamon-elsevier science, 145-153.

Müller, M. J., Wildman, D. M., e White, E. A. (1994) Participatory design through games and other exercises. *Tutorial at CHI'94 conference*, Boston.

Müller, M. J., McClard, A., Bell B., Dooley S., Meiskey L., Meskill J. A., Sparks R., and Tellam, D. (1995) Validating an Extension to Participatory Heuristic Evaluation: Quality of Work and Quality of Work Life, In *Proceedings of CHI ´95 Conference companion*, 115-116.

Müller, M. J., Haslwanter, J. H., e Dayton, T. (1997) Participatory Pratices in the Software Lifecycle in *Handbook of Human-Computer Interaction*, M. Helander, T. K. Landauer, P. Prabhu (eds.), Elsevier Science, 2 ed., 255-297.

Müller, J. M., Matheson, L., Page, C. & Gallup, R. (1998) Participatory Heuristic Evaluation. *Interactions*. September + October 1998. V. 5, Issue 5, ACM Press, 13-18.

Nied (1999) *Projeto de dinamização da formação e da aprendizagem nas empresas*. II Relatório Parcial de Atividades (01/03/1997 a 01/07/1998). Projeto apoiado pela Fapesp (processo número 96/9810-8).

Nied (2002) *O uso da Internet na formação colaborativa e descentralizada de funcionários de fábricas enxutas.* Relatório final de Atividades (01/09/2000 a 31/08/2002). Projeto apoiado pela Fapesp (processo número 2000/05460-0).

Norman, D. (1990). *The Design of Everyday Things*. New York: Doubleday

Norman, D. (1999) Affordances, Conventions, and Design, *Interactions*, May 1999, 38-43.

Norman, D. (2002) *Affordances and Design*, Don Norman & The Nielsen Norman Group, Referencia online, http://www.jnd.org (11/07/2004)

Oliveira, OL e Baranauskas M.C.C. (1999) Communicanting Entities: a Semiotic-Based Methodology for Interface Design. In *Human-Computer Interaction – Ergonomics and User Interface*, H. J. Bullinger e J. Ziegler (eds), London: Lawrence Erlbaum Associates Publishers, vol.1.

OMG, (2003) *OMG Unified Modeling Language Specification, Version 1.5*. www.omg.org/uml

OSW (1995) *The circulation document in the Organizational Semiotic Workshop*, Enschede.

Palshaugen, O. (1986) Method of designing a starting conference. *Oslo: Work Research Institute*.

Peirce, C.S. (1931-1958) *Collected Papers*, Cambridge, Mass: Harvard University Press.

Pressman, R. S. (1995) *Engenharia de Software.* Makron Books, São Paulo.

Ronald, M. B., Jonathan, G., William, A. S. B. and Saul, G. (1995) From Customizable Systems to Intelligent Agents. *Readings in Human-Computer Interaction*, second edition, Morgan Kaufmann Publishers, Chapter 12, 783-831.

Robinson M. (1993) Design for unanticipated use..., *In Proceedings of The Third European Conference on Computer-Supported Cooperative Work*, edited by G. DeMichelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, 187-202.

Rocha, H.V. e Baranauskas, M.C.C. (2000) *Design e avaliação de interfaces humano-computador*. Escola de computação:IME-USO, São Paulo, v1.

Rose, A., Shneiderman, B., e Plaisant, C. (1995) An applied ethnographic method for redesigning user interfaces in *ACM proceedings of DIS'95 Symposium on Designing Interactive Systems: Processes*, Practices, Methods & Techniques , 115-122

Schlichter J., Koch M. e Burger M. (1998) Workspace awareness for distributed teams. *Coordination thechnology for collaborative applications*, Springer-Verlag Berlin, 199-218.

Schuler, D. e Namioka, A. (1993) *Participatory design: Principles and Practices*. Lawrence Erlbaum Associates, Inc., Publishers, New Jersey, USA.

Schuluzen, K. (2000) *A criação de um ambiente de aprendizado contextualizado baseado no computador para formação de recursos humanos em empresas enxutas.* Tese de Doutorado, FEE-Unicamp.

Sebeok, T. A. (1994) *Signs – An Introduction to Semiotics.* Toronto: University of Toronto Press Incorporated.

Simoni, C. A. C e Baranauskas, M. C. C. (2003) Da Análise de Requisitos ao Projeto de Interface: uma Abordagem Subjetivista para Sistemas de Informação. *Congresso Latino-americano de Interação Humano-Computador*, Rio de Janeiro, Brasil, Agosto, 17-20.

Simonsen, J. e Kensing (1997) Using Ethonography in Contextual Design. Em *Communications of the ACM*, vol. 40, no. 7, Pg. 82-88.

Silva, A. M. (2001) *Uma Abordagem Participativa ao Design de Ambiente Computacional de Apoio a Discussão de Problemas no Contexto de Trabalho.* Dissertação de Mestrado apresentada ao Instituto de Computação - Unicamp.

Silva, A. M. e Baranauskas, M. C. C. (2000a) The Andon System: Designing a CSCW Environment in a Lean Organization. Em Proc. of *Sixth International Conference on Groupware (CRIWG'2000)*. IEEE Computer Society Press.

Silva, A. M. e Baranauskas, M. C. C. (2000b) Partipatory Techniques in the Design of Andon: A Computer-Supported Environment for Collaborative Work. Em *Proceedings International Conference on Engineering and Computer Education - ICECE'2000*.

Silva, A. M., Bonacin, R., and Baranauskas, M. C. C., (2001) Collaborative Discussion and Decision-Support in a Manufacturing Organization, Em *Proceedings 6th International Conference on Computer Supported Cooperative Work in Design*. NRC of Canada Research Press. London, Ontario - Canada, 117-122.

Sjöström, J. e Goldkuhl, G. (2003) The semiotics of user interfaces - a sócio-pragmatic perspective. Em *Proceedings of 6th International Workshop on Organisational Semiotics,* 57-72.

Souza, C.S. (1993) The Semiotic Engineering of User Interface Languages, *International Journal of Man-Machine Studies*, 39, 733-753.

Souza, S. M. (2003) *Estendendo Ambientes de Suporte a Trabalho Cooperativo com base no conceito de workflow*. Dissertação de Mestrado apresentada ao Instituto de Computação - Unicamp.

Souza, S. M., Bonacin, R. e Baranaukas, M. C. C. (2003) Design de um Gerenciador de Fluxo de Tarefas no Contexto de Aprendizado em Manufatura. Proceedings of *3rd International Conference on Engineering and Computer Education*.

Sommerville, I (2000) *Software Engineering*, Addison-Wesley Pub Co, 6th edition.

Stamper, R. K. (1973) *Information in Business and Administrative Systems*, John Wiley & Sons (ed.).

Stamper, R. K. (1979) Towards a Semantic Normal Form. In G. Bracchi and G. Nijssen (Eds.) *Database Architecture*, North-Holland, Amsterdam.

Stamper, R. K., Althaus, K. e Backhouse, J. (1988) MEASUR: Method for Eliciting, Analyzing and Specifying User Requirements. In Olle, T. W., Verrijn-Stuart, A. A. e Bhabuts, L. (eds.) *Computerized Assistance During the Information Systems Life Cycle*. Elsevier Science, Amsterdam, 67-116.

Stamper, R. K. (1992) Language and computer in organized behaviour, in Riet, R. P. v .d. and Meersman, R. A., (eds.), *Linguistic Instruments in Knowledge Engineering*, Elsevier Science, 143-163.

Stamper, R., (1993), Social Norms in Requirements Analysis – an outline of MEASUR. In Jirotka, M., Goguen, J. and Bickerton, M. (eds.), *Requirements Engineering, Technical and Social Aspects,* Academic Press, New York.

Stamper, R., (1996), Signs, Information and Systems, in B. Holmqnist, et. Al. (Eds) *Signs of Work Semiotics Information Processing in Organisations*, Walter de Gruyter, N. Y.

Stamper, R. K. (2000) Information Systems as a Social Science: An Alternative to the FRISCO Formalism. In E. D. Falkenberg, K. Lyytien, A. A. Verrijn-Stuart (eds) *Information System Concepts: an Integrated Discipline Emerging*, Kluwer Academic Publishers, USA, p.1-51.

Stamper, R., Liu, K., Hafkamp, M. and Ades, Y., (2000) Understanding the roles of signs and norms in organizations – a semiotic approach to information system design, *Behaviour & Information Technology,* 19, 1, 15-27

Suchman, L. (1989) *Notes on Computer Support for Cooperative Work*. Working Paper WP-12, Dept. of Computer Science, University of Jyvaskyla, SF-40100, Jyvaskyla, Finland.

Teege, G., Kahler, H. and Stiemerling, O. (1999) Implementing Tailorability in Groupware, In *SIGGROUP Bulletin*, 20, 2, 57-59.

Thönissen, K. (1990), *Semantic Analysis: a Study and Comparison.* MSc Thesis, University of Twente, Enschede.

Trepess D. e Stockman T. (1999) A classification and analysis of erroneous actions in computer supported co-operative work environment, *Interacting with computers*, Elsevier Science, 611-622.

Womack, J. P., Jones, D.T. and Roos, D. (1990) *The Machine that Changed the World*, Harper Collins Publishers.

Wixon, D. R., e Raven, M. B. (1994) Contextual inquiry: Grounding your design in user work. *Tutorial at CHI'94 conference*, Boston.

Wulf V., Stiemerling O. and Pfeifer A. (1999) Tailoring groupware for different scopes of validity. *Behaviour & information technology*, Taylor & Francis, 199-212.

Xie, Z., Liu, K., e Emmitt, D. (2003) Improving Business Modelling with Organisational Semiotics, In H.W.M. Gazendam, R. J. Jorna, R. S. Cijsonw (eds.) *Dynamics and change in Organizations - Studies in Organizational Semiotics*, Kluwer Academic Publishers, 85-102.

Anexo A: O Resultado da Aplicação das Técnicas de Design Participativo

Anexo B: Modelos do Pokayoke

B.1 Modelos da Semiótica Organizacional

B.2 Modelos Entidade Relacionamento

B.3 Modelos UML

Anexo C: Documentação do NBIC

Anexo A: O Resultado da Aplicação das Técnicas de Design Participativo

Número do Relatório: 01	**Data:** 08/11/2001
Atividades: reunião	**Duração:** 4h
Objetivo: Estabelecer objetivos e pontos de contatos	**Ciclo:** 1°
Participantes: 5 universidade (pesquisadores) 4 Organização (gerentes)	

Fomos à fábrica ontem e os resultados foram melhores do que esperava, eles demonstraram estar realmente interessados nos projetos (tomara que não seja só no início).

Quanto ao Andon, esperam implantar o quanto antes, mas vamos ter que refazer o sistema e alterar a maneira como foi implemento, pois ele funciona bem para mono usuário mas tem defeitos para multi-usuário (como por exemplo quando insere algo novo na base isto não reflete imediatamente nos outros computadores, que só atualizam os dados quando é reiniciado, o problema é que não tem um servidor para manter a consistência). Eles têm rede NT e desejam colocar o sistema na intranet, eu não sei se mais para frente eles vão querer também por na Internet. Eles demonstraram interesse em testar o Andon em Janeiro, desta maneira vamos poder iniciar nossos trabalhos.

O Amadeu ficou de estabelecer "Owners" para cada projeto dentro da empresa. Para o Andon já foi estabelecido um (o do Jogo da Fábrica e do Alvo ainda não), o nome dele é Rafael. Ele é um funcionário do Recusos Humanos e servirá como facilitador para interagirmos com os outros funcionários. Ele pareceu bastante interessado no projeto, uma vez que foi ele mesmo quem pediu para o Amadeu indicá-lo para trabalhar com o Andon. Na verdade, eles ficaram bem interessados no sistema porque têm o objetivo de estimular o pessoal a utilizar os "5 passos" para todos os problemas que ocorrem na fábrica e a idéia de "gerencia de conhecimento" foi muito bem vinda para eles, inclusive o Amadeu falou que ele tem guardado em papel uns 400 problemas resolvidos com o 5passos e que nós poderíamos alimentar o sistema para fazer testes e que seria muito interessante "resgatar" estes problemas (só tem que achar alguém para digitar).

Aparentemente gostaram muito quando falei que estes problemas não só poderiam ser apresentados ao usuário mas também que eles poderiam servir para propor soluções para problemas novos. Eles também gostaram muito do projeto de Workflow, inclusive propuseram que o sistema mandasse e-mail quando houvesse algum problema em atraso.

O Rafael ficou de mandar um e-mail marcando uma reunião para a próxima semana ou no máximo daqui duas semanas (pois semana que vem tem um feriado prolongado). Agora no início vou apresentar o Andon para o Rafael e para o pessoal da qualidade que utilizarão o sistema (espero que estes gostem da idéia). Pensei em propor alguma técnica de design participativo que trabalhe o modelo de semiótica (que construí pelos relatórios do André), para a gente trabalhar quais mudanças seriam necessárias.

Aparentemente eles demonstraram interesse em modificar duas coisas principais no sistema e de incluir um monte. A primeira coisa é alterar o nome, antes mesmo de começar a apresentação o Amadeu já pediu para mudar o nome do sistema, porque Andon significa outra coisa para eles (é o semáforo) e pelo que eu entendi quando eles falam "sistema Andon", significa o sistema de semáforos. Parece que isto já gerou dúvidas quando eles conversaram com o Valente e o Mazzone, pois falaram que eles já tinham o sistema Andon (o do semáforo). O nome sugerido por eles foi Pokayoke, que significa na qualidade "a prova de erros". Outra mudança que deve ser feita (que até já tinha discutido com o André anteriormente) é incluir os cinco passos de forma explícita no "Pokayoke" (inclusive o DCE e o Brainstorming estão invertidos na interface).

Mesmo que não seja possível trabalhar com muita intensidade devido à problemas de tempo por parte deles, eles demonstraram ter uma mentalidade complemente diferente da fábrica de xxxxxx. O Amadeu também levou a gente para dar uma volta na fábrica e ela é realmente interessante.

Número do Relatório: 02	**Data:** 30/11/2001
Atividades: Conferencia semiótica *Storyboard Prototyping* Avaliação do sistema Andon Artifact walkthrough	**Duração:** 4h
Objetivo: discutir o contexto organizacional através da SO, avaliar o sistema Andon e Design de alto nível da interface	**Ciclo:** 1°
Participantes: 2 designers 4 trabalhadores (de três plantas diferentes)	

Alguns Resultados desta visita:

Pokayoke parece ser o nome adequado mesmo. Perguntamos se este seria um nome adequado e todos eles concordaram. Durante a descrição de seus trabalhos, eles apresentaram o manual dos 5 passos e tem uma parte que diz o seguinte:

> "O objetivo deste procedimento (cinco passos) é definir um método disciplinado para sempre que uma situação de não conformidade for identificada, uma Ação seja tomada para corrigi-la e impedi-la que ocorra novamente. Sempre que a possibilidade de uma situação de não conformidade for indicada,

uma ação de caráter irreversível deve ser adotada (error proofing /Poka Yoke)"

Desta maneira o objetivo do processo de resolução de problemas é chegar em uma solução Poka Yoke, além disso falaram que é um termo conhecido na qualidade e é sempre relacionado a processo de resolução de problemas. Portanto o objetivo do sistema é chegar sempre em soluções PokaYoke.

Algumas coisas sobre o processo de resolução de problemas não estavam definidas ainda, por isso participaram do encontro os responsáveis pela qualidade de todas das plantas. Com isso foi possível trabalhar juntos para definir como iria ser o 5 passos comum a todas as plantas (no sistema e mesmo antes do sistema).

Ficou estabelecido o objetivo de construir mecanismos para estimular que qualquer pessoa da organização possa abrir o 5 passos (atualmente só a qualidade abre). Segundo o Rafael é um objetivo do RH estimular todas as pessoas a utilizarem os 5 passos.

Após pedir para que eles apresentassem as ferramentas e explicassem como eles trabalhavam com elas (Artifact walkthrough), tentei trabalhar com os diagramas da Semiótica Organizacional. Apresentei o diagrama de ontologias e discutimos em cima dos conceitos descritos (expliquei como eu tinha entendido o contexto através do diagrama), e eles falaram o que estava certo ou errado (não deu para discutir tudo, tinha muita coisa). Isso levou a uma discussão sobre responsabilidade: quem deveria ser cobrado no "follow-up", quem deveria ser responsável por cada fase no processo de resolução de problemas, etc.

Desta discussão (sobre o diagrama de ontologia) saíram algumas alterações no diagrama de ontologia, e também definições para o sistema e o módulo de Workflow. Outras alterações são (ainda sem ver como refletir no sistema): Inserir o 5 Porquês na causa Raiz (identificar a causa raiz normalmente é o principal), o Brainstorming é algo que serve para identificar a solução e algumas das propostas apresentadas que devem ser implementadas. Ficou decidido também que o Ivan iria me mandar cópia eletrônica dos documentos que eles mostraram.

Apresentei o sistema (Andon) e eles sugeriram uma série de alterações, e propuseram a inclusão do "cinco passos" de maneira explícita no Pokayoke. Desenhamos algumas telas em papel (de acordo com a *Storyboard Prototyping*) e tentamos expressar a idéia de que o usuário deve seguir os 5passos e durante o processo possa utilizar as ferramentas para discussão (colaborativa). Isso está de acordo com o que foi modelado no diagrama de ontologia.

Ficou acertado que a gente construiria uma versão das telas do sistema que contenha os pontos levantados por eles (baseado nos resultados deste encontro e na análise das ferramentas em papel). A estratégia adotada foi modelar e implementar um sistema completamente novo (Pokayoke), e reaproveitar a "estrutura" e a "idéia" por traz de algumas ferramentas do Andon.

A impressão geral é que o pessoal está realmente empolgado com o sistema, eles querem colocar para funcionar logo e ir testando. Outro fato que achei interessante é que o Rafael recebeu elogios da chefia do RH sobre o andamento do trabalho, na minha visão isto demonstra o apoio deles.

Foi marcada uma data para próxima visita na qual iremos apresentar as telas (prototipação rápida).

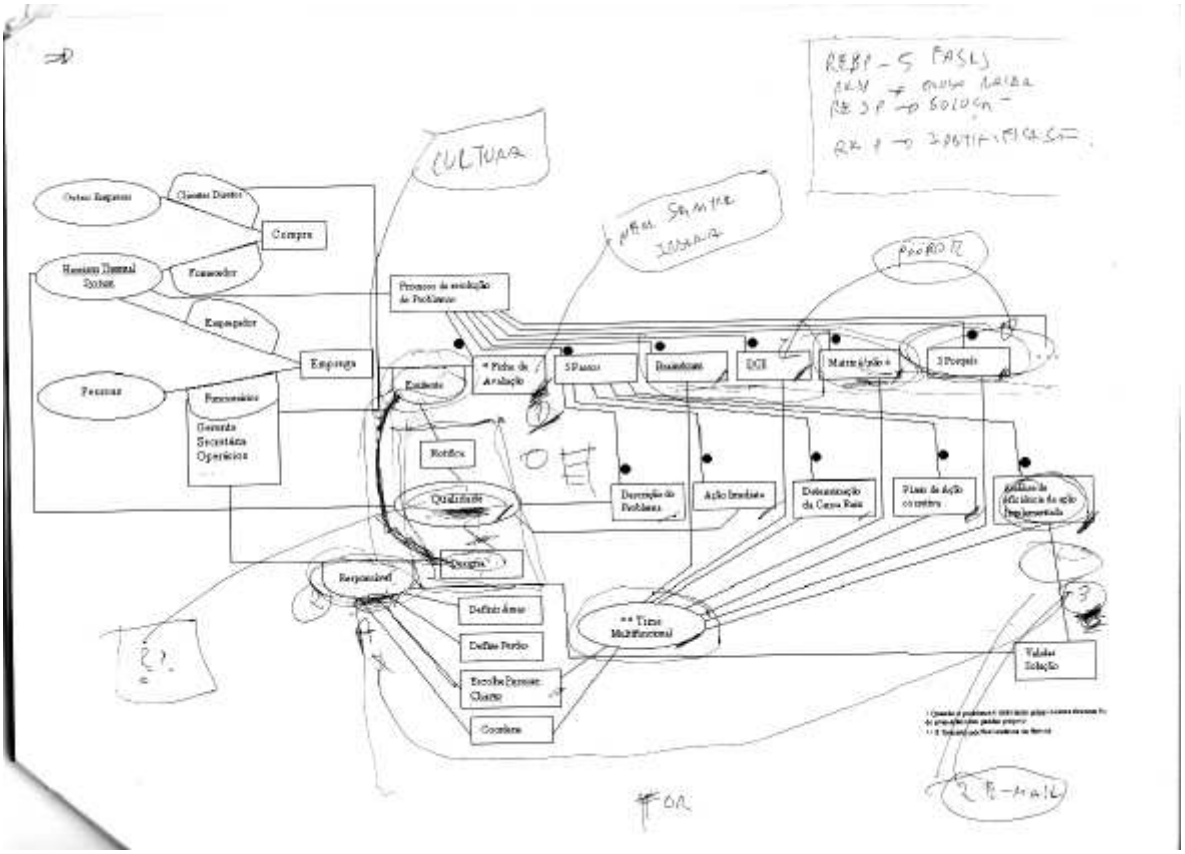


Figura A.1 – Exemplo de discussão durante a Conferência Semiótica

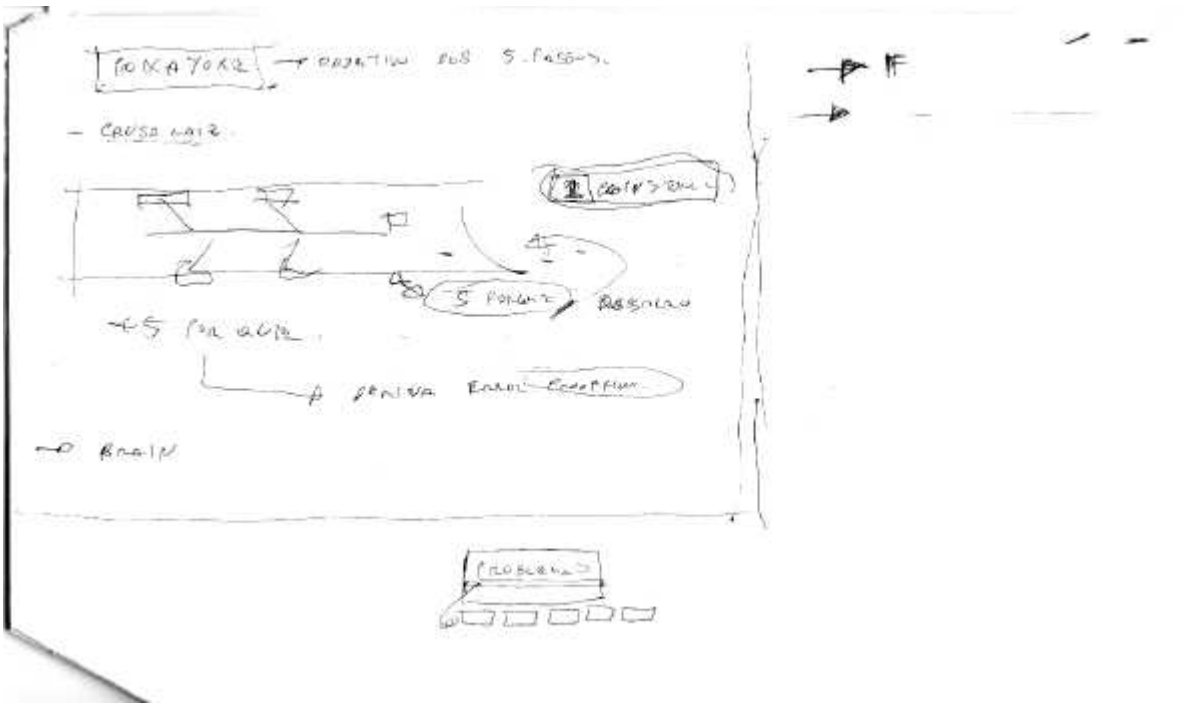


Figura A.2 – Exemplo de desenhos durante a técnica de *Storyboard Prototyping*

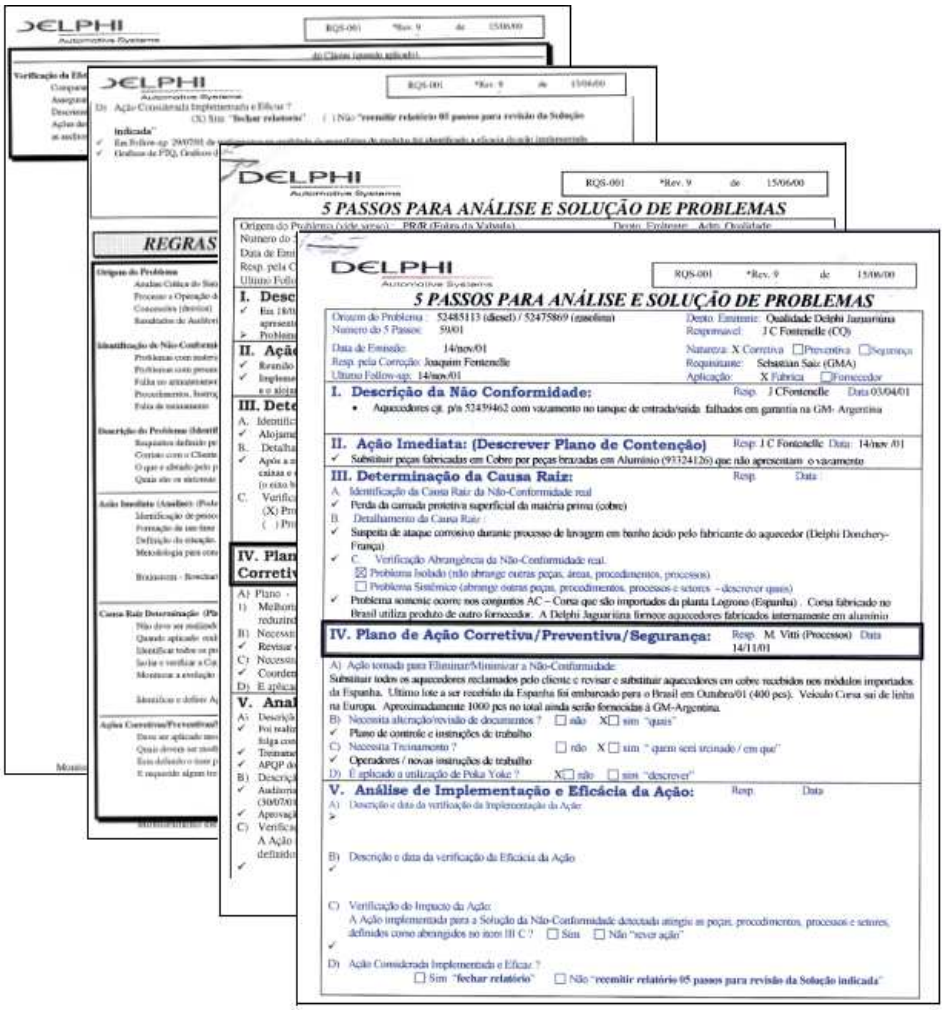


Figura A.3 – Exemplo de artefatos trabalhados durante a aplicação da técnica de *Artifact walkthrough*

Número do Relatório: 03	**Data:** 04/02/2002
Atividades: Avaliação da interface *Storyboard Prototyping CISP*	**Duração:** 3h
Objetivo: Discutir e evoluir o primeiro portótipo	**Ciclo:** 2°
Participantes: 2 designers 4 trabalhadores	

Resultado das discussões:

1.1 Tela Pokayoke:

Alterar Problemas em Processo de Resolução para Problemas Não Solucionados.
Tirar Ver 5 Passos (ficaria uma tela muito carregada).
Ainda será definido o que vai no botão Pesquisar.

1.2 Tela Inserir Novo Problema

Trocar "Responsável pela Correção" por "Responsável 5 passos".
Importante: Requisitante é quem encontra o problema (cliente) e comunica alguém da Delphi. Responsável é quem abre o 5 Passos. Entre os dois pode haver alguém que atendeu o Requisitante, e comunicou a pessoa mais apropriada para ser o Responsável. O Resp. 5 Passos é quem iniciará a correção. Este definirá, mais à frente, os responsáveis pelas ações.
Em Regras Básicas deve haver um help para preenchimento dos dados (ensinar como preencher os 5 passos).
Foi solicitado o preenchimento automático da máscara em Word que irá para o Fornecedor.
Deve ser pensada a tela para Definir Grupo.

1.3 Tela Inserir Não Conformidade

O campo Responsável deverá ser preenchido automaticamente, e ser protegido (devemos repensar esta proteção, para os casos de mudanças de cargos, férias, etc.).
Proteger também o campo Data.
O botão Regras Básicas deve chamar "Help".
Tornar disponível a inclusão de foto digital da peça com problema para facilitar identificação do mesmo.
Retirar o campo Observações.

1.4 Tela Ação Imediata

Reunir na mesma linha Ação Imediata, Resp. Ação e Data (repensar com eles).
Desabilitar Responsável e Data (repensar com eles).
Brainstorm conterá discussão sobre quais ações deverão ser implantadas.
Concluir = concluir passo. OK = inserir.
Tirar Observações.

1.5 Tela Determinação da Causa Raiz

DCE e 5 Porquês = discussão sobre a causa raiz
Passar Soluções para a próxima tela.

1.6 Tela Plano de Ação Corretiva/ Preventiva/Segurança

Desabilitar Responsável e Data (repensar).
Tirar Observações.
Implementar = implementar a ação
Receber Soluções.

1.7 Tela Análise de Implementação e Eficácia da Ação

Definir tela para Lições Aprendidas.
Controlar para que este passo não seja disponibilizado antes da conclusão do passo anterior.

2. Passos futuros: o Rodrigo fará as alterações para que o sistema possa ser utilizado e avaliado para alterações futuras.

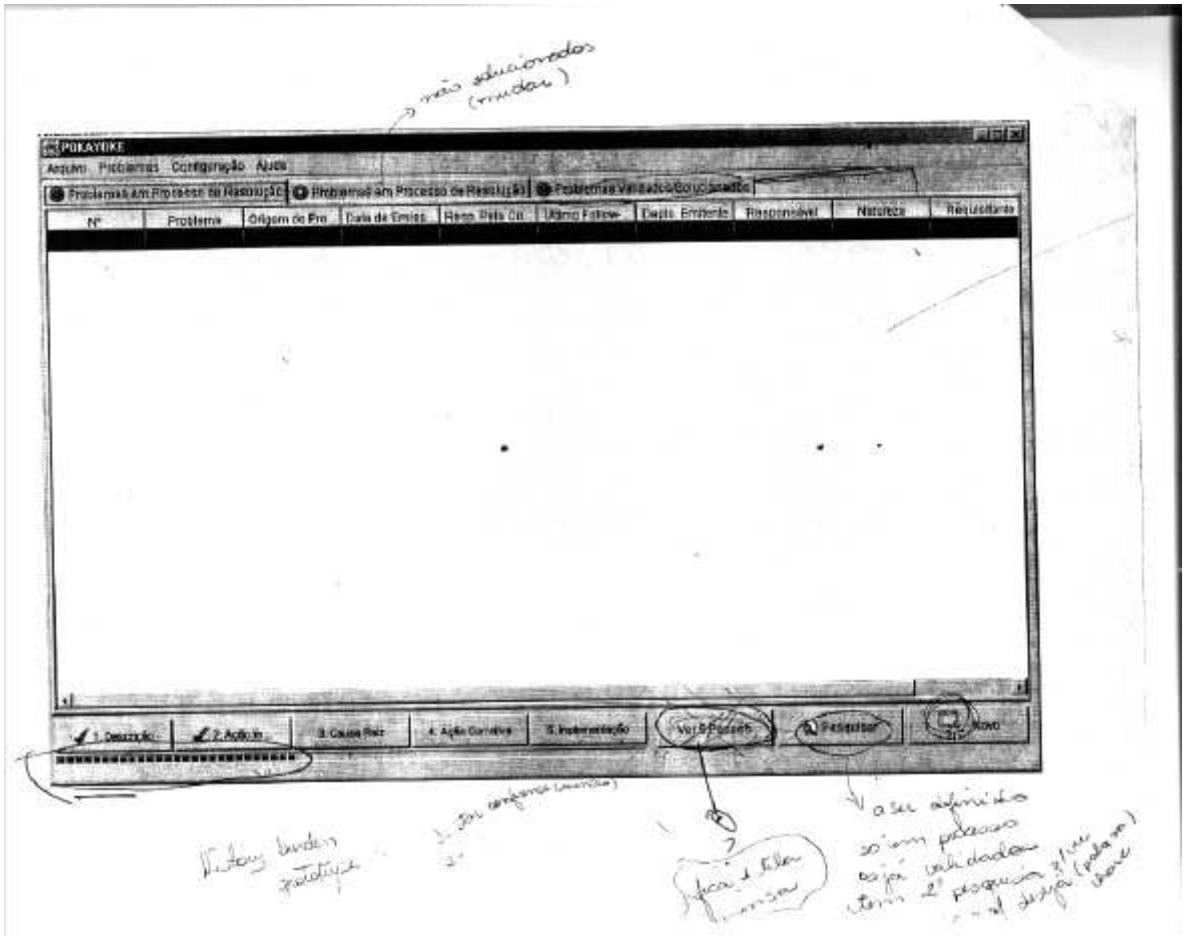


Figura A.4– Tela principal trabalhada durante a técnica de *Storyboard Prototyping-CISP*

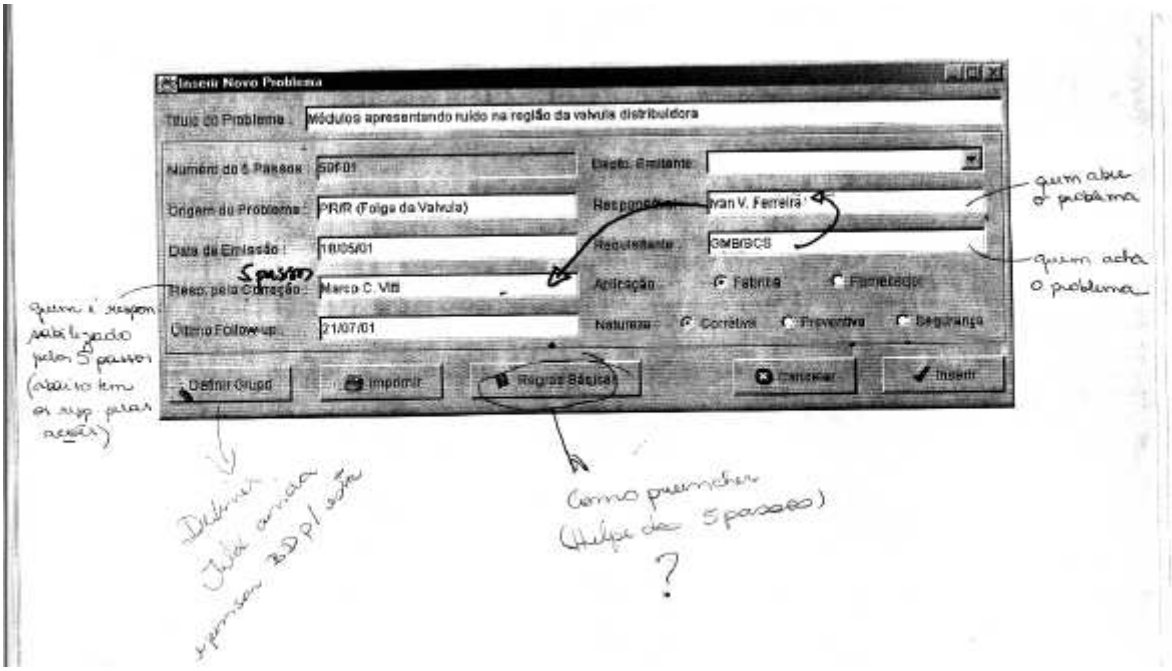


Figura A.5– Tela para inserir problemas trabalhada durante a técnica de *Storyboard Prototyping-CISP*

Figura A.6– Tela para o primeiro passo trabalhada durante a técnica de *Storyboard Prototyping-CISP*

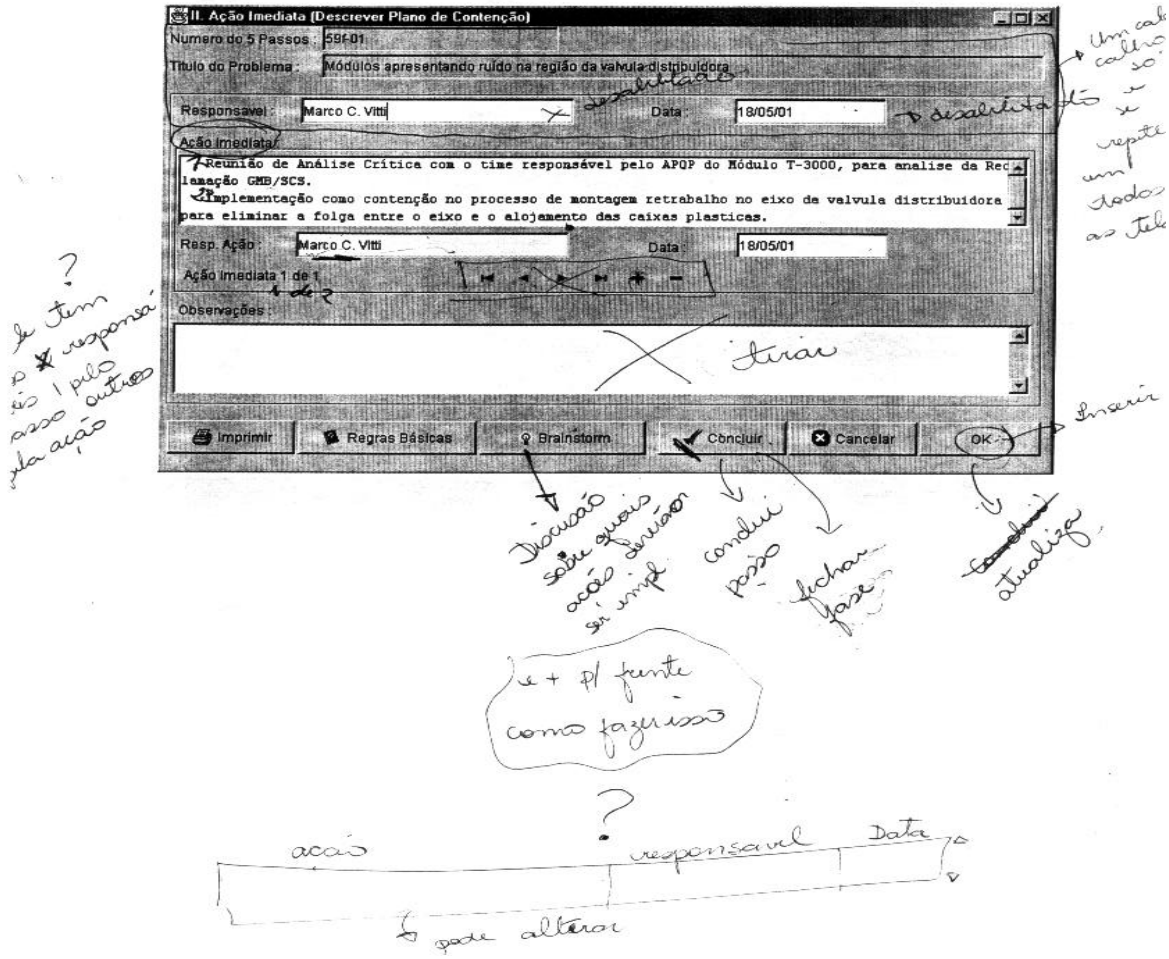


Figura A.7– Tela para o segundo passo trabalhada durante a técnica de *Storyboard Prototyping-CISP*

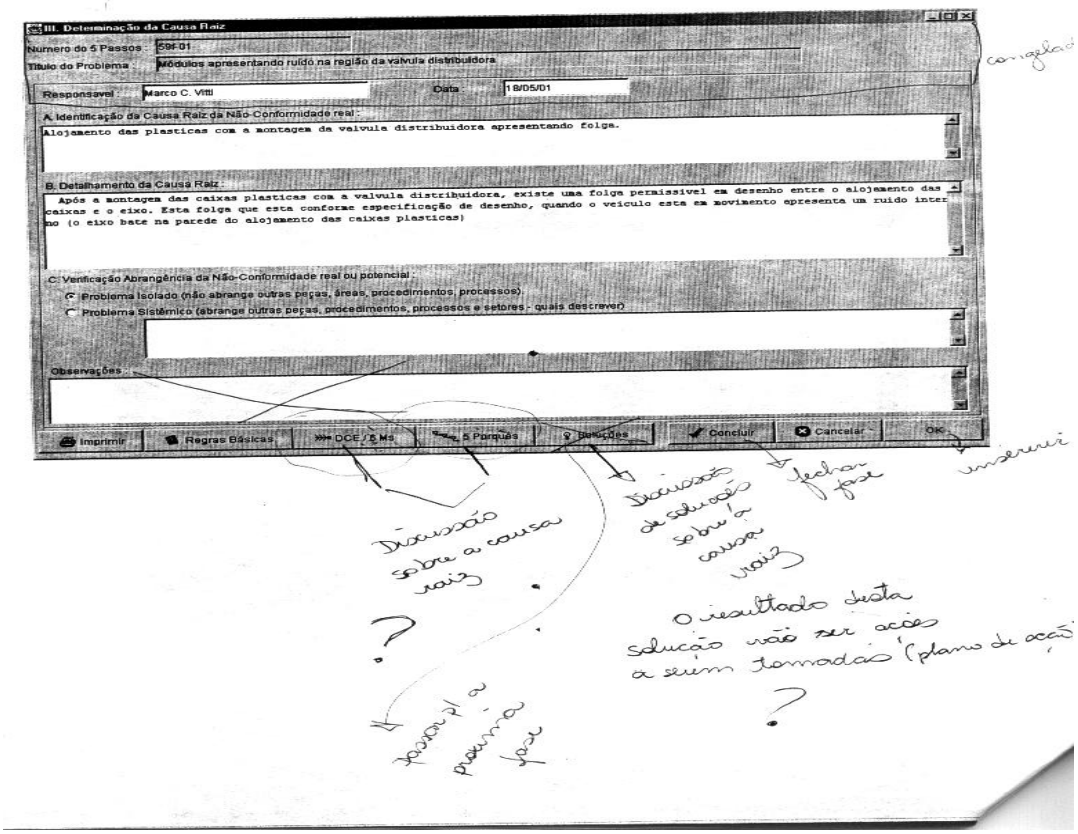


Figura A.8– Tela para o terceiro passo trabalhada durante a técnica de *Storyboard Prototyping-CISP*

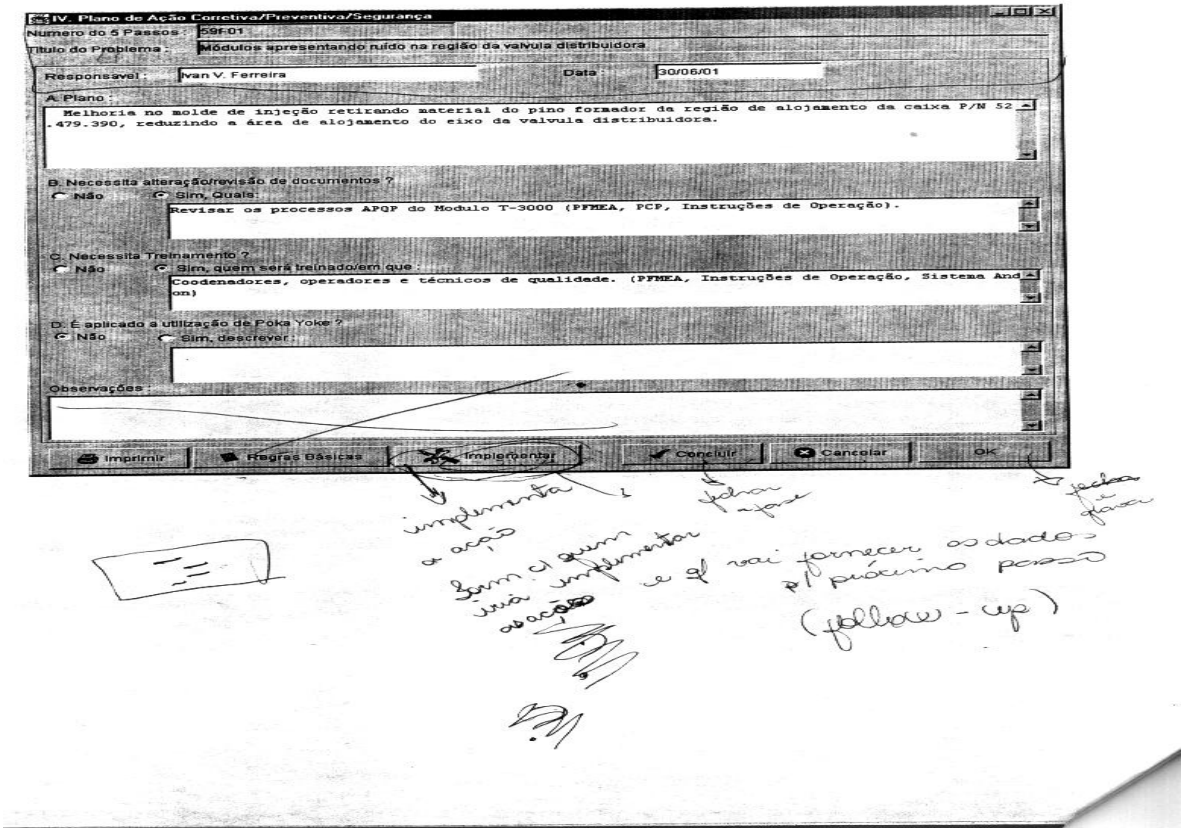


Figura A.9– Tela para o quarto passo trabalhada durante a técnica de *Storyboard Prototyping-CISP*

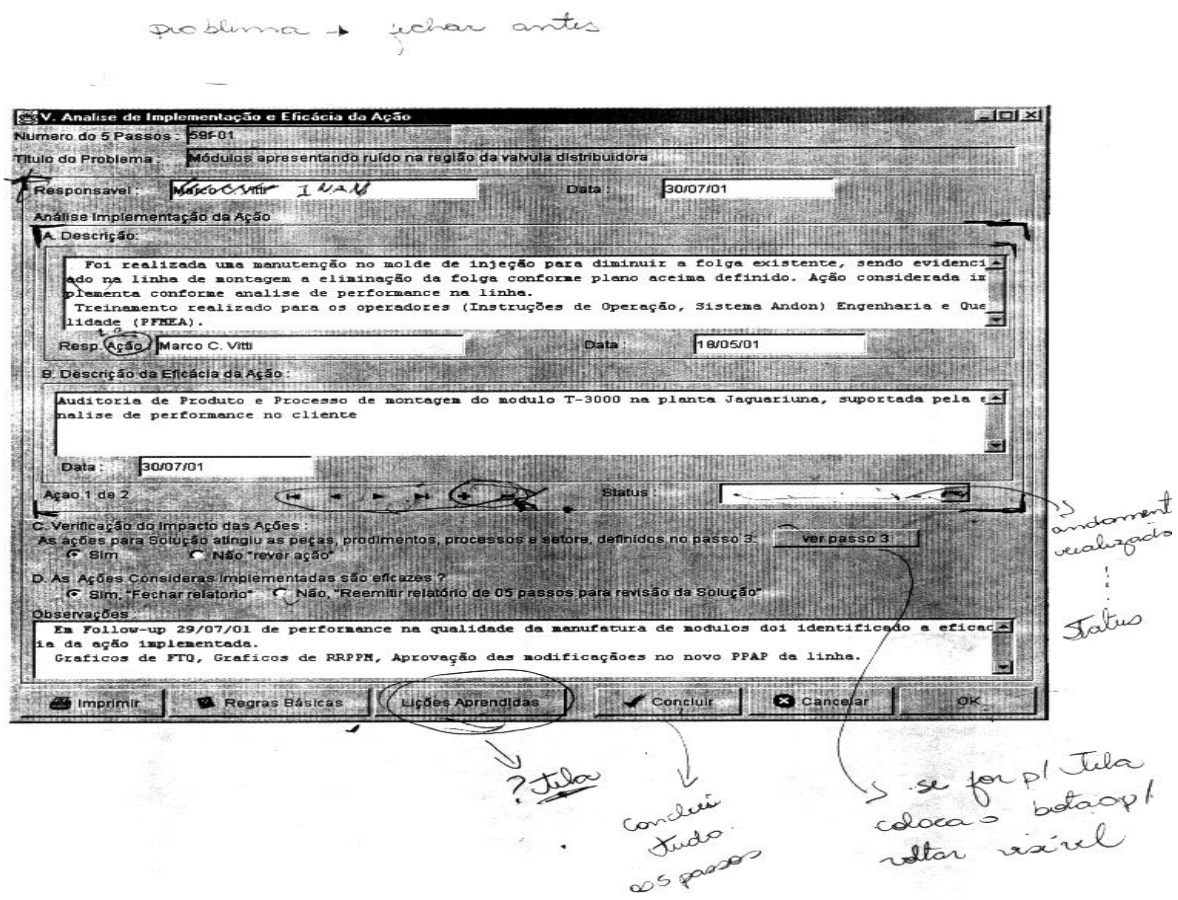


Figura A.10– Tela para o quinto passo trabalhada durante a técnica de *Storyboard Prototyping-CISP*

Número do Relatório: 04	**Data:** 27/03/2002
Atividades: Conferência Semiótica	**Duração:** 4h
Objetivo: Discutir conceitos da organização e como eles são refletidos no sistema	**Ciclo:** 2°
Participantes: 2 designers 2 trabalhadores	

Fomos ontem à Deplhi, participamos de uma reunião somente com o Rafael e o Ivan. Apresentei os novos diagramas (de ontologia) e também as novas telas do sistema. Foi discutido diagrama principal e todos os outros diagramas, acho que o modelo está representando razoavelmente o contexto deles.

Outras mudanças são:

- o time multifuncional também participa do (5porquês);
- o responsável pelo 5passos é quem supervisiona o time multifuncional, e não o responsável pela emissão;
- (Affordance - participante)

- Foi discutido o que cada pessoa pode ver no sistema (todo mundo pode ver todos os problemas da empresa?)
- Todo mundo pode ser requisitante e/ou participar de um time multifuncional

- (Affordance - emitir)
  - Quem pode abrir um 5 passos? Todo mundo pode ser requisitante e participar de um time multifuncional, mas nem todo mundo pode abrir um 5 passos pois ele passa a ser responsável (emitente) por ele. Ficaram de definir quem pode ser. Atualmente só o pessoal da qualidade pode ser responsável, mas a intenção é mudar quando o sistema estiver rodando. A princípio, disseram que cada pessoa poderá ser responsável somente por problemas do seu departamento.

- (Affordance "role-name" - Responsável pelo 5 passos)
  - Quem pode ser responsável pelo 5 passos (antigo responsável pela correção), são poucas pessoas. Segundo eles, um funcionário de chão de fábrica pode participar e sugerir alterações, mas quem responde por estas alterações são poucas pessoas. Ficaram de conversar entre eles e definir exatamente quem poderá ser.

- (Affordance "role-name" - Responsável pelas ações)
  - Quem pode ser responsável por uma ação? (ficaram de discutir entre eles, mas parece que todo mundo pode ser)

- (Affordance - Re-emite)
  - Quando uma solução não acabou com o problema, o cinco passos deve ser re-emitido. A idéia é incluir (o que ainda não existe) um relatório para descrever o que falhou

Vou associar estas funcionalidades com normas nas ferramentas, pois se mudarem dá para alterar no sistema facilmente.

As mudanças no sistema são muitas, a mais significativa é no passo II onde o problema não precisa ter "Status", pois ação imediata é para ser rápida e atualizar o "status" no sistema é algo que vai mais "atrapalhar" do que ajudar, uma vez que a ação é realizada na mesma hora em que foi definida. O botão de cobrar também deve ter ação restrita neste passo (apenas lembrar), ao contrário do passo IV. O diagrama de ontologia está certo, existe a cobrança no passo II, mas no passo IV ela é feita de forma mais complexa e a ferramenta tem mais que auxiliar lá do que aqui.

Foi identificado que atualmente (no procedimento em papel) existe um problema de interpretação quando as pessoas preenchem o item A e B do passo III (identificação da causa raiz da não-conformidade real e detalhamento da causa raiz). Para evitar este problema eles ficaram de criar uma lista com os possíveis valores para o item A (que devem ser poucos) uma vez que a descrição vai mesmo para o item B. Eles falaram que vão alterar também o procedimento em papel.

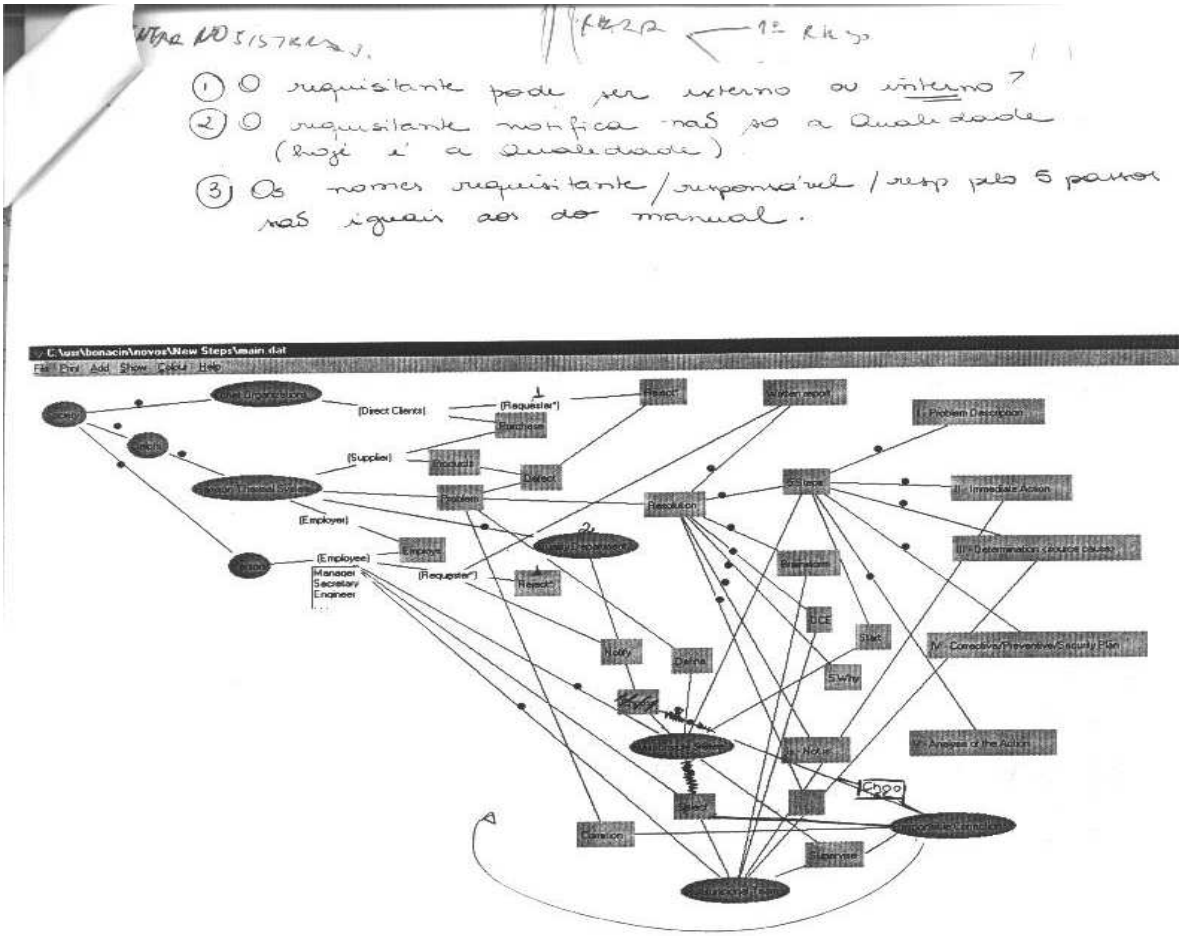


Figura A.11– Exemplo de discussão durante a Conferência Semiótica (diagrama principal)

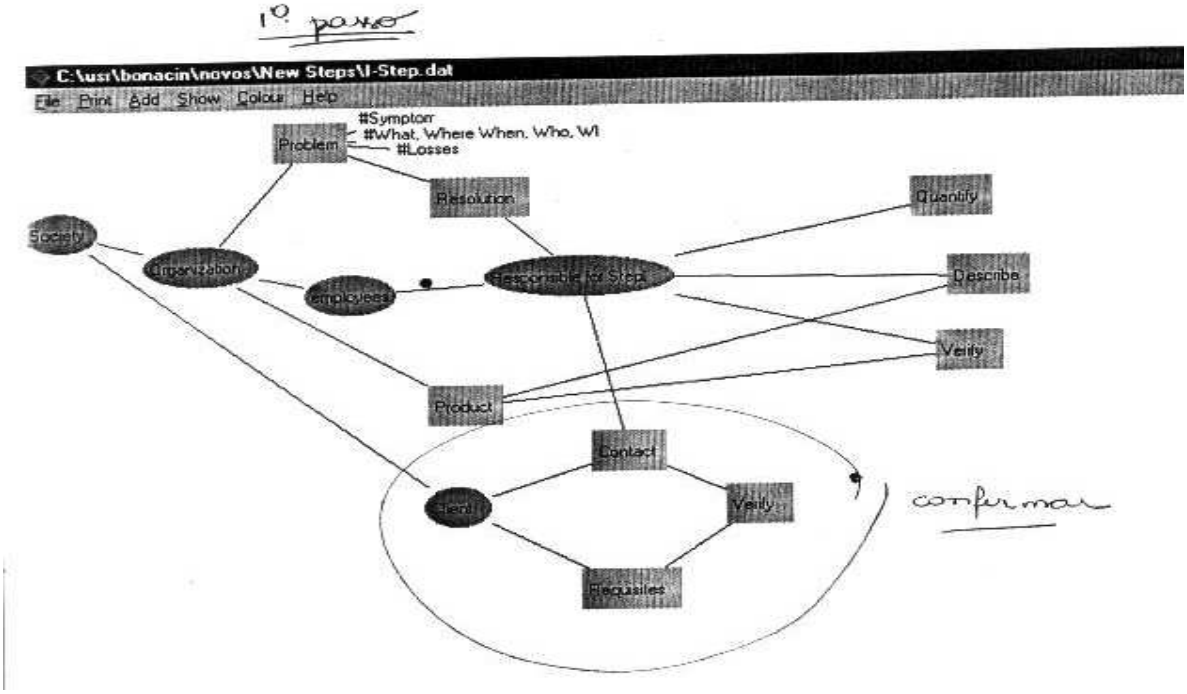


Figura A.12– Exemplo de discussão durante a Conferência Semiótica (diagrama do primeiro passo)

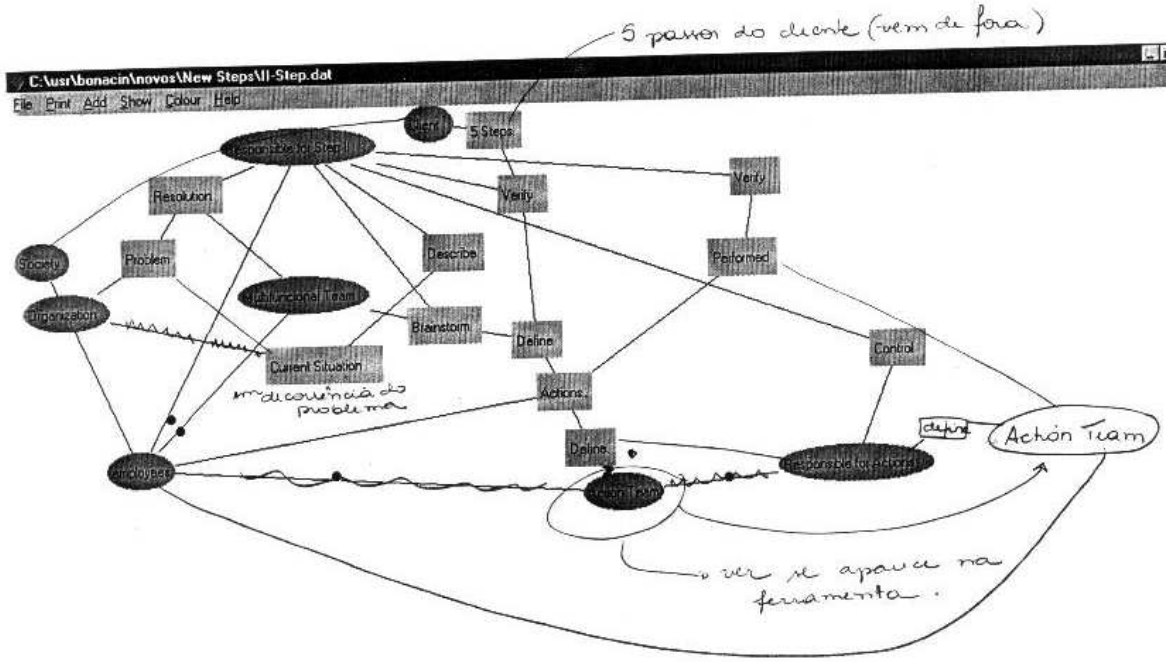


Figura A.13– Exemplo de discussão durante a Conferência Semiótica (diagrama do segundo passo)

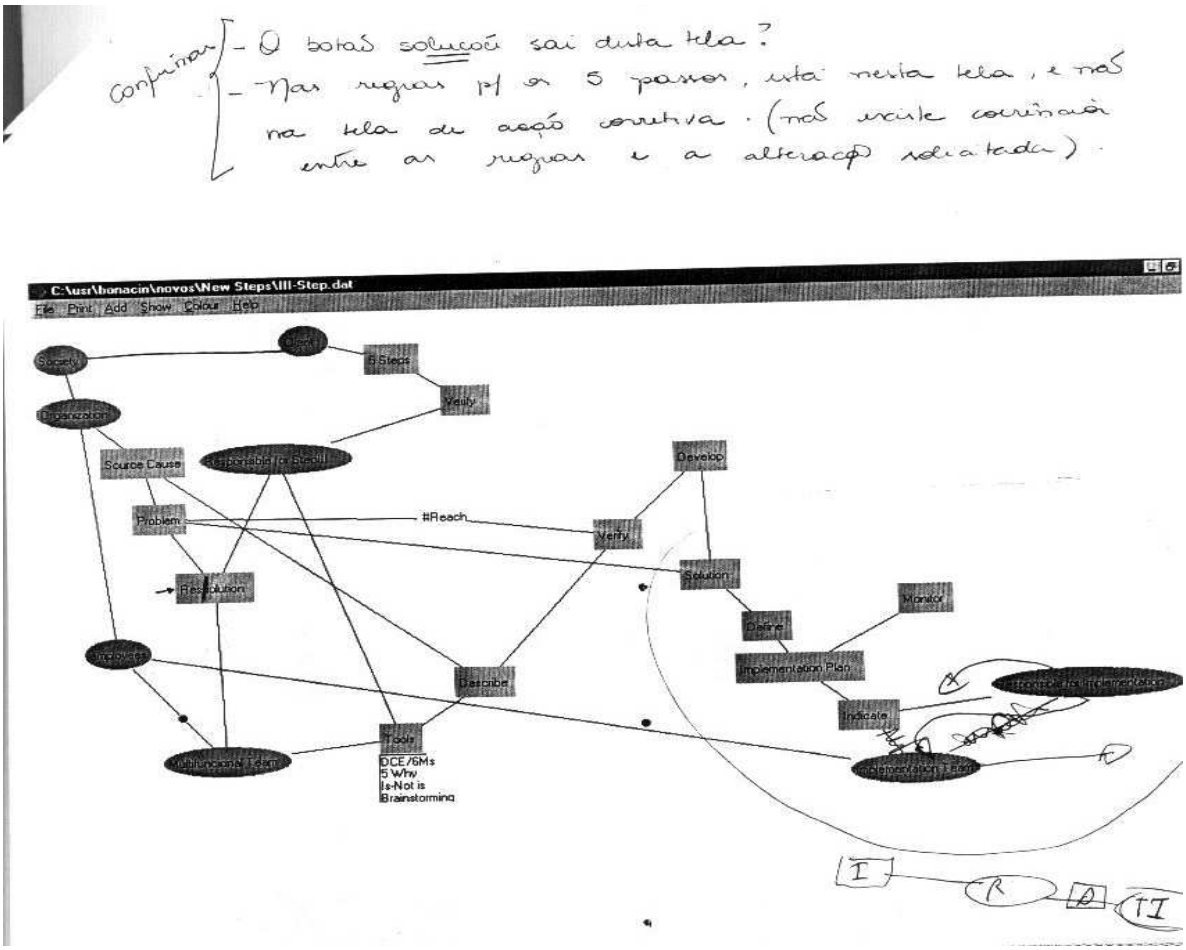


Figura A.14– Exemplo de discussão durante a Conferência Semiótica (diagrama do terceiro passo)

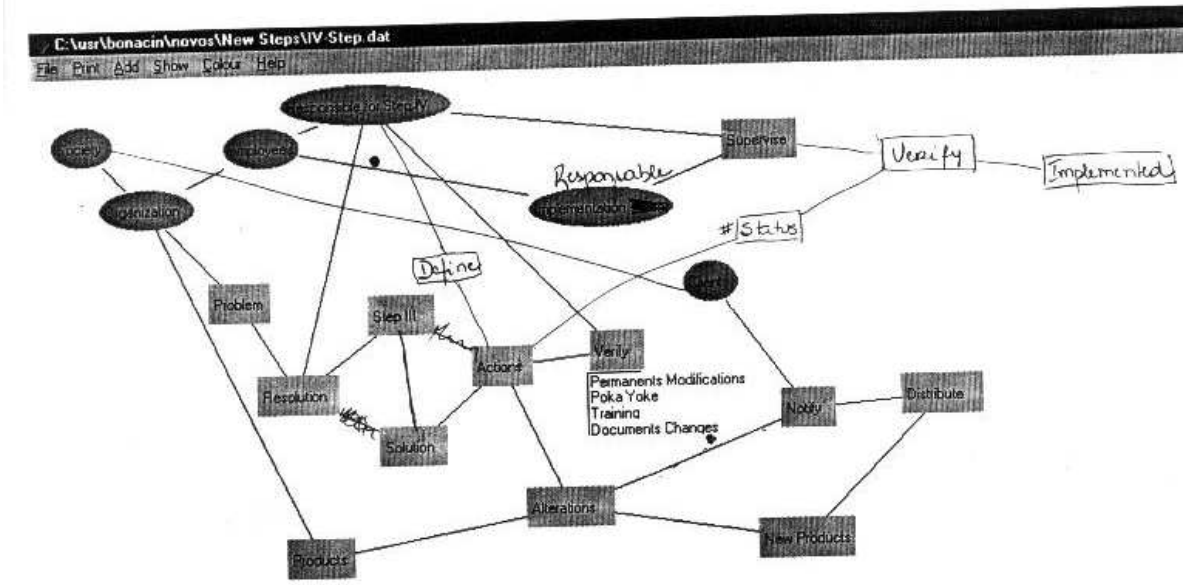


Figura A.15– Exemplo de discussão durante a Conferência Semiótica (diagrama do quarto passo)

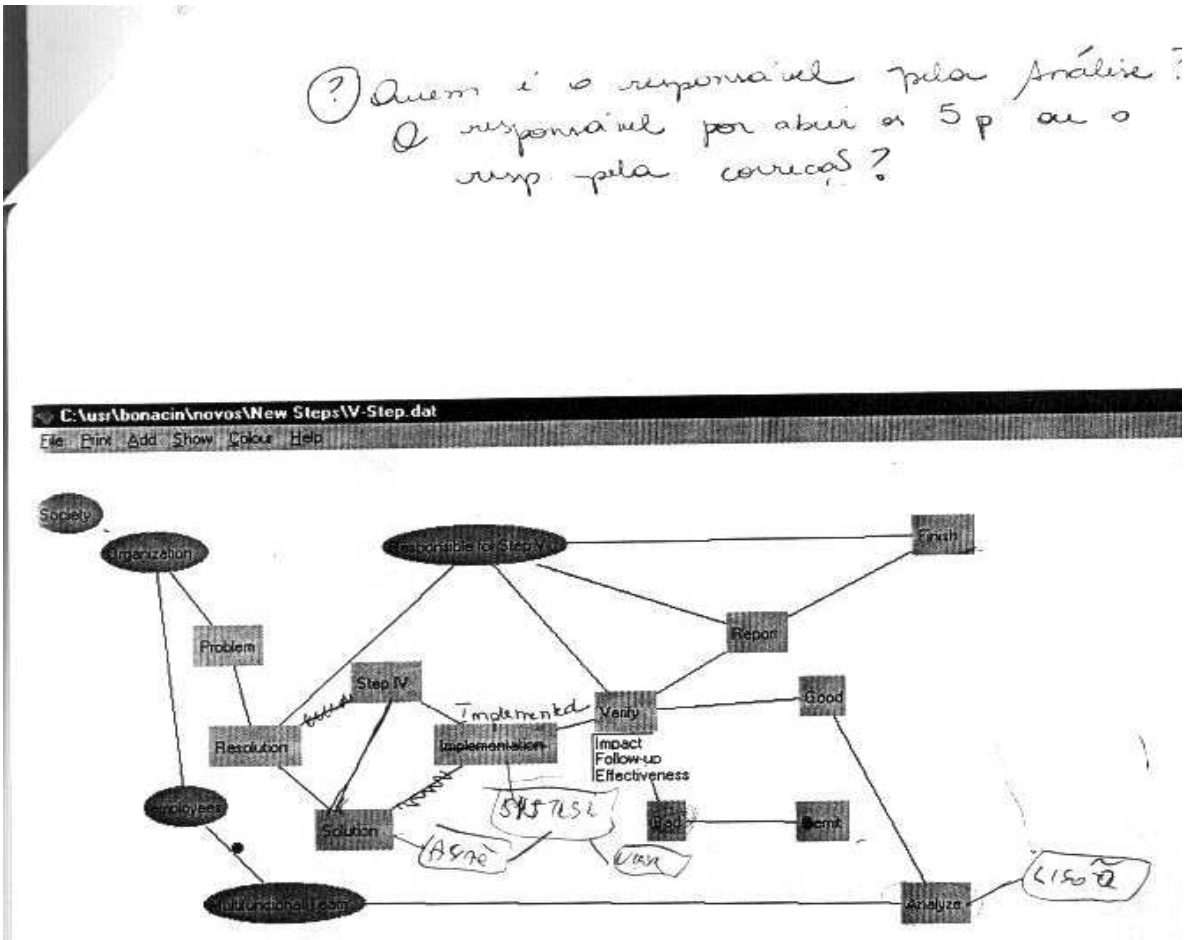


Figura A.16– Exemplo de discussão durante a Conferência Semiótica (diagrama do quinto passo)

Número do Relatório: 05	**Data:** 03/06/2002
Atividades: Estabelecer novo contato Avaliação do protótipo Artifact walkthrough	**Duração:** 3h
Objetivo: Estabelecer novo contato, Avaliar protótipo (agora Funcional) , Verificar documentos padrões da fábrica	**Ciclo:** 3°
Participantes: 2 designers 2 trabalhadores	

Fomos hoje a Delphi. Nosso contato direto e facilitador mudou, é o "Robson da qualidade". O Ivan também vai acompanhar de perto, mas ele falou que é interessante incluir o Robson, pois ele tem mais tempo e usa o cinco passos de forma mais intensiva no dia a dia. Também teremos o acompanhamento de um cara do RH, o Cleiton.

O Robson é quem faz o 5 passos com os fornecedores quando acontece de alguma peça vir com defeito para a Delphi. Ele é responsável pelo primeiro e último passo, e os fornecedores para os passos 2, 3 e 4. Ele tem interesse em colocar o Pokayoke na Internet. Discuti com ele como incluir os fornecedores no processo (temos que revisar o diagrama de ontologia).

Instalei uma versão do Pokayoke agora funcional, na máquina dele para que ele possa avaliar. Durante o encontro, o Robson e o Ivan inseriram um 5 passos para testar. Eles falaram que é mais ou menos isso que esperavam do sistema e mudaram um pouco a sistemática do 5 passos,por causa de um problema ocorrido na fábrica. Vai ter que fazer algumas alterações no programa, mas nada significativo pois o sistema já possuía estas alterações conforme discutido nas reuniões anteriores. Inclusive falaram que se o programa estivesse rodando, seria bem provável que o problema não tivesse ocorrido. O que aconteceu foi o seguinte: parece que dois engenheiros pulavam o passo IV, o que não dava para fazer no sistema e também teve casos que não escreviam corretamente o item A do passo III. Com o sistema, este caso, provavelmente, não teria ocorrido, porque conforme discutido na visita anterior incluí um "Combo" com todas as opções possíveis.

Embora estes problemas, não ocorreriam no Pokayoke, tentamos achar mais mecanismos para evita-los. Como resultado desta discussão, devemos fazer algumas alterações no Pokayoke. Um resumo destas alterações é:

- Incluir um checklist no passo II (não está representado na figura A18)
- Aprovação de um responsável para verificar se tudo foi preenchido corretamente no passo IV. Para mim, isto é meio burocrático, mas o Ivam falou que é importante pelo menos no começo até o pessoal estar habituado a preencher corretamente o 5 passos

Ficamos também de alterar as máquinas do Nied que estão lá, pois são 486. A idéia é colocar uma máquina de servidor Pokayoke e instalar clientes em outras máquinas para o pessoal utilizar/testar. O jogo do Alvo também não roda nestas máquinas. O Manoel falou que tem 3 máquinas aqui no Nied que podem ser levadas para lá, só que tem que trazer de volta as outras máquinas. São três PII 333 (uma delas é a que estava com o Lotus Notes), não são máquinas ótimas, mas acho que é suficiente.

A Lia ficou de entrar em contato com o Ivan para agendar um dia para fazer esta troca. A princípio, vou instalar o Pokayoke e colocar versões clientes para mais pessoas avaliarem. Por enquanto, ele só tem na máquina do Robson (com servidor local). Em discussão com o pessoal da fábrica, chegamos a conclusão que a melhor idéia é que após esta fase de avaliação devemos incluir algumas opções a mais no sistema, como por exemplo: pesquisa e relatórios. Após isso, pretendemos colocar pelo menos uma versão instalada em cada departamento da empresa e depois instalar na máquina de todos os usuários, que é o objetivo final. O Cleiton, do RH, falou que vai apoiar a disseminação da cultura de utilizar o 5passos. Deixei bem claro para eles que o programa não faz nada sozinho, é preciso que as pessoas estejam convencidas quanto à importância de utilizar os 5 passos antes disso.

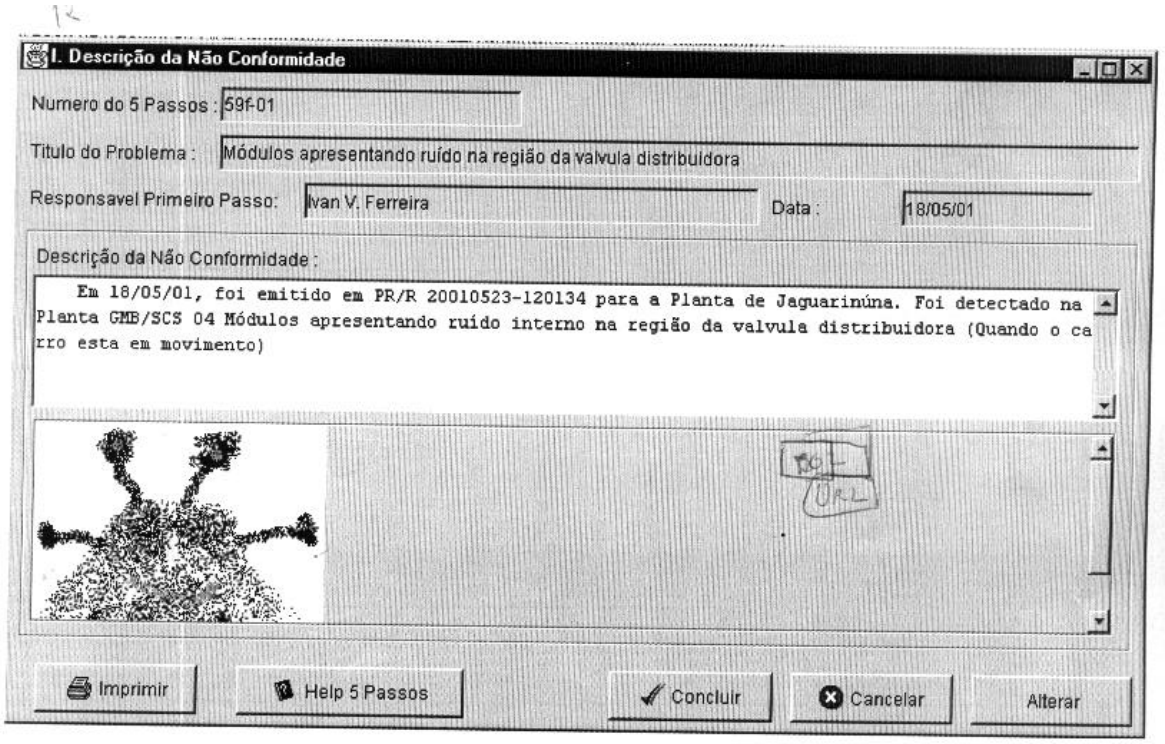


Figura A.17– Alterações propostas para o primeiro passo

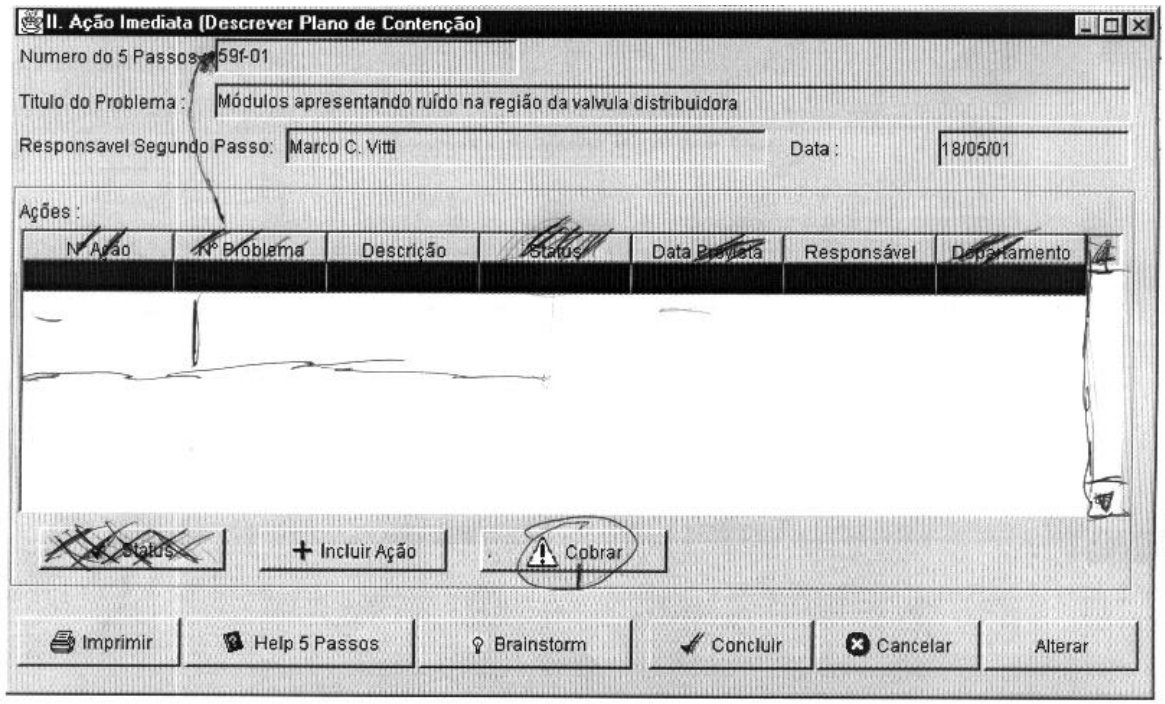


Figura A.18– Alterações propostas para o segundo passo

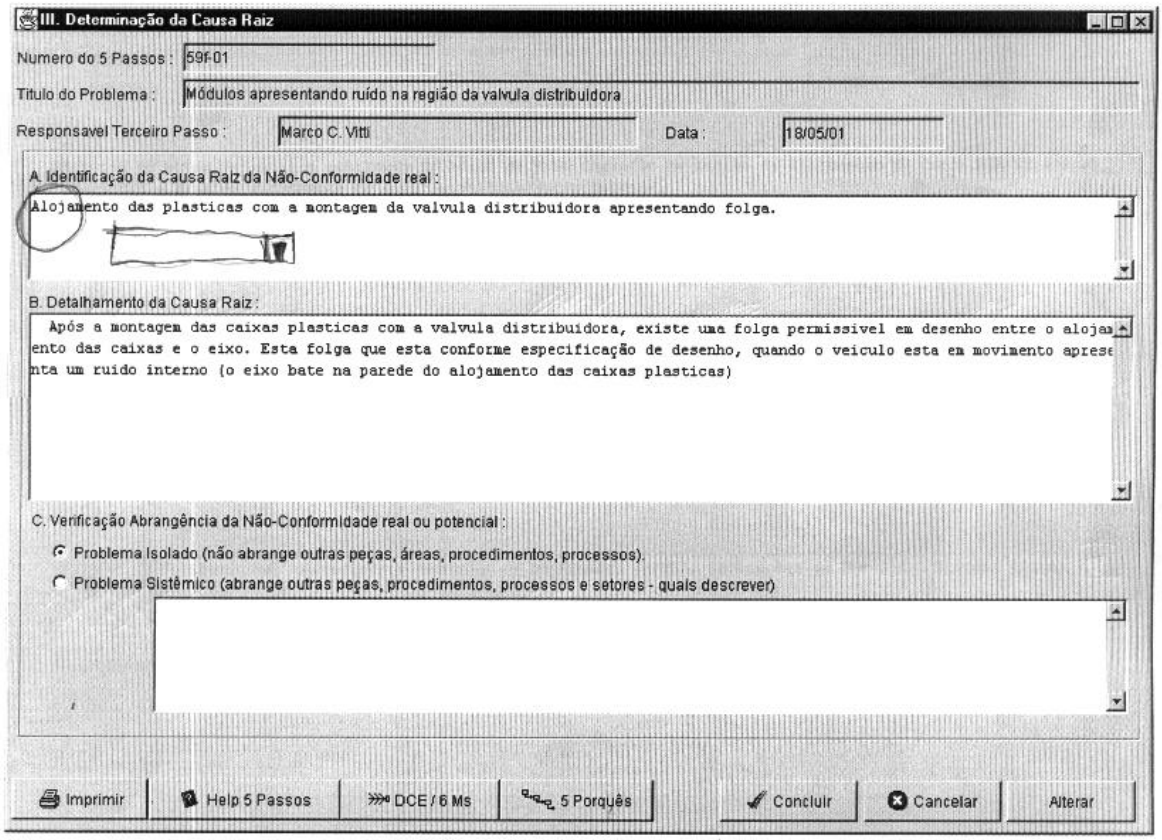


Figura A.19– Alterações propostas para o terceiro passo

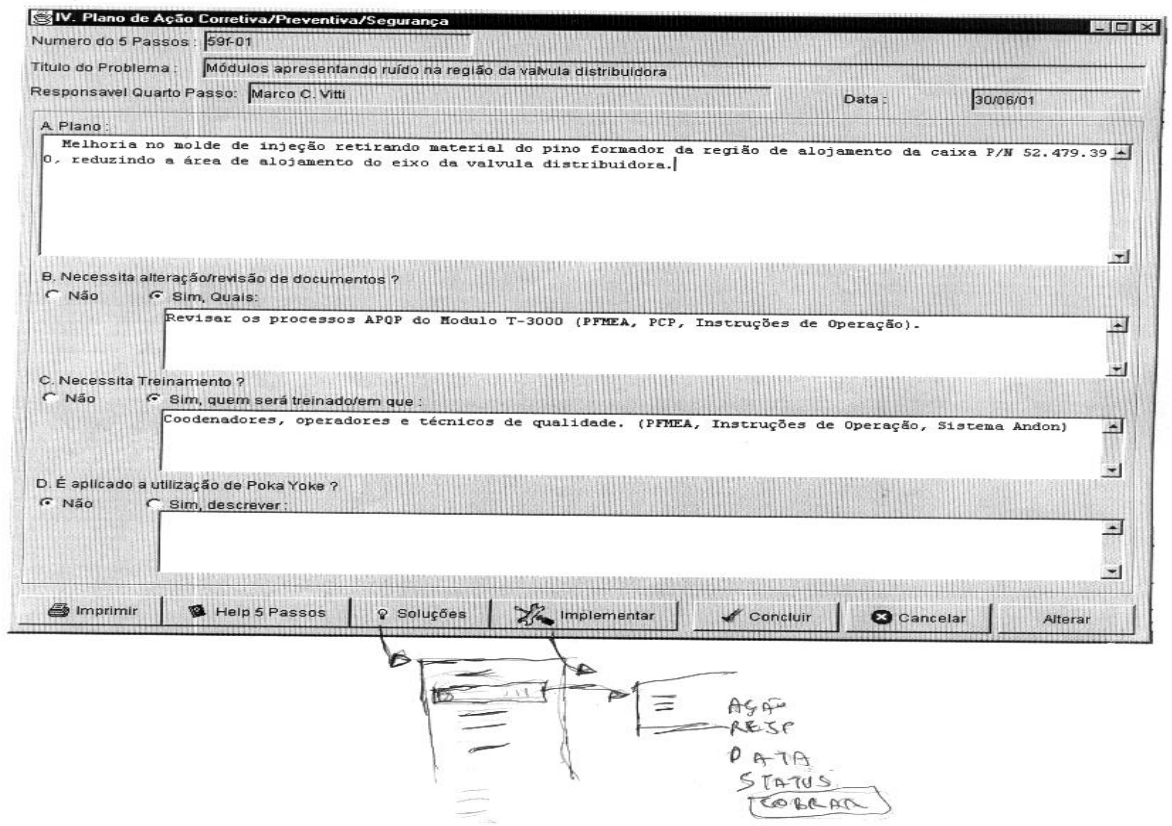


Figura A.20 – Alterações propostas para o quarto passo

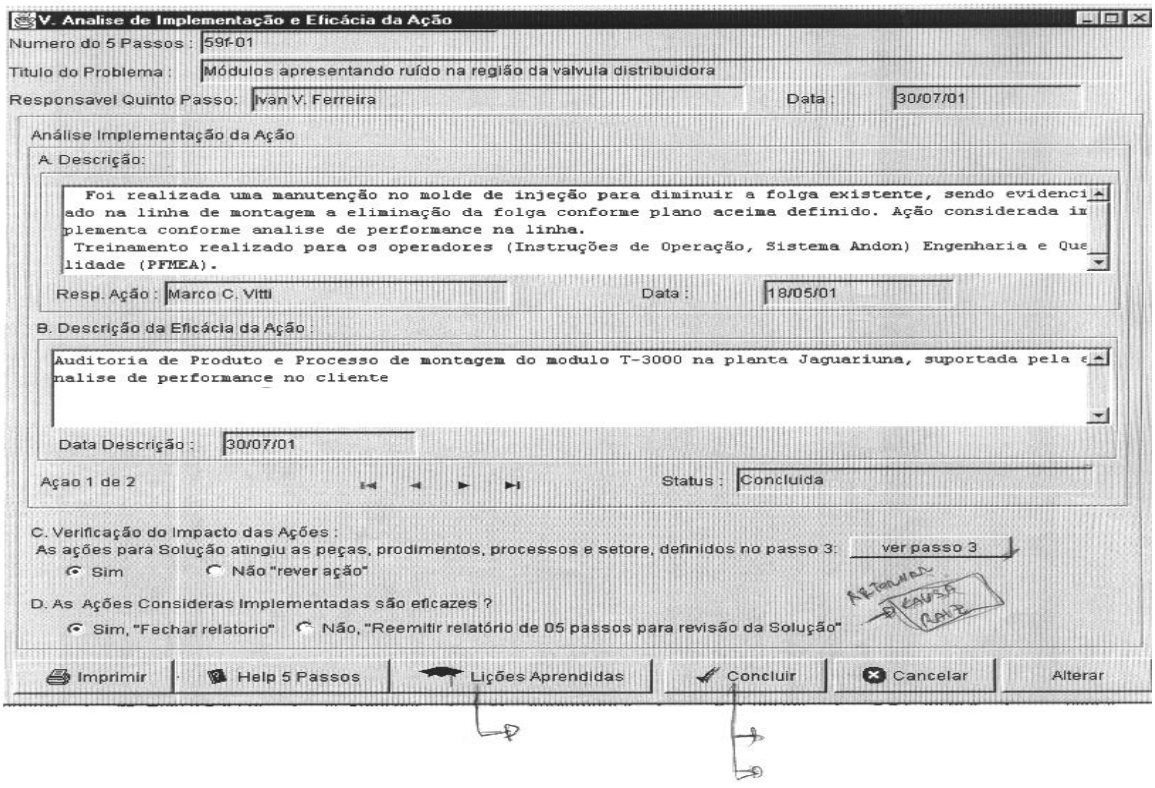


Figura A.21 – Alterações propostas para o quinto passo

Número do Relatório: 06	**Data:** 19/06/2002
Atividades: Instalação de máquinas	**Duração:** 2h
Objetivo: Instalar máquinas para colocar protótipos do sistema em Teste	**Ciclo:** 3°
Participantes: 2 designers	

Fomos ontem na Delphi levar as máquinas. Após andar pela fábrica inteira, conseguimos entrar com elas. Eles só entregaram 2 máquinas (e não três), mais 3 monitores e 3 teclados. Em resumo não entregaram a "obra de arte" e nem os "mouses" que fugiram. Telefonamos para o Manuel e ele falou para pegarmos o que tinha que ser pego e trazer uma declaração da Delphi que as coisas que não foram entregues continuavam sob a responsabilidade deles.

Colocamos as máquinas, 2 couberam na mesa, a outra, ficaram de providenciar uma outra mesa. Falta configurar o IPs das máquinas. Temos que conversar com o pessoal de sistemas de lá. Faltou ainda um adaptador do teclado para PS2 em uma das máquinas.

Ficou marcado para instalarmos o servidor Pokayoke na semana que vem.

Número do Relatório: 07	**Data:** 06/07/2002
Atividades: Starting Conference Instalação do Sistema em máquinas de usuários piloto	**Duração:** 4h
Objetivo: Discutir implantação do sistema e estratégia de uso na Organização	**Ciclo:** 3°
Participantes: 1 designers 3 trabalhadores	

Foi instalado o sistema e as máquinas foram incluídas na rede da empresa. Discutimos (Starting Conference) e estabelecemos quais seriam os passos para a implantação do sistema:

- Primeiro para um grupo de teste da qualidade, que é composto de 5 pessoas (Ivan, Agnaldo, Flavio, Robson, Joaquim). Este grupo utilizará e avaliará a ferramenta, que posteriormente passará a ser utilizada por toda a fábrica
- O Robson fica como contato mais direto e também como um “multiplicador” de como utilizar o sistema na fábrica

Número do Relatório: 08	**Data:** 13/07/2002
Atividades: PHE Teste com usuários	**Duração:** 4h

Objetivo: Avaliação da interface do sistema, teste do sistema em condições de uso e decisões de design	**Ciclo:** 3°
Participantes: 1 designer 2 trabalhadores	

Foi instalada uma nova versão do sistema em duas máquinas e foram realizados testes do sistema. O teste foi feito da seguinte maneira: não passei instruções de como utilizar a ferramenta e também não foi utilizado "help" do sistema (mesmo porque nesta versão não existe). A minha participação se limitou a explicar que foi incluído um Brainstorm para propor soluções, visto que isto não existe na versão em papel. O usuário foi alguém treinado para realizar o cinco passos na versão em papel, porém não participou das atividades de desenvolvimento do sistema computacional.

Com dados sobre um "5 passos" realizado anteriormente, o usuário tentou simular como seria este 5 passos utilizando o sistema Pokayoke. Todas as etapas foram realizadas com êxito. Por se tratar de uma simulação com um único usuário, este assumiu o papel de várias pessoas do processo de resolução de problemas (responsável, responsável 5 passos e pessoas do grupo multifuncional, etc). Entre as tarefas realizadas, as que mais ocuparam tempo para assimilar como poderia fazê-las no sistema, foram as ligadas funcionalidades do sistema que não estavam presentes na versão em papel, como: definir o time funcional e propor idéias no Brainstorming de soluções.

Dificuldades encontradas para manipular a interface:

- O tamanho da figura com a descrição do problema. A foto da peça com problema normalmente é muito grande e o sistema não a manipula de maneira adequada. No relatório, ela é cortada toda vez que excede o tamanho da página;
- A necessidade de dar duplo "click" do mouse para inserir elementos em listas (definição do time multifuncional, ações imediatas e ações);
- O sistema mudava o problema selecionado quando um passo era encerrado: sistema não mantinha o mesmo problema que o usuário estava editando, logo ele entrava em um próximo passo de outro problema
- O programa apagava a caixa de texto quando o usuário pressionava no título de uma coluna no "listbox"
- Seqüência de foco utilizando "Tab" nem sempre era a seqüência esperada pelo usuário (foram anotadas as seqüências certas)

Também encontramos o seguinte Bug de execução: o sistema travou quando o usuário tentou imprimir um relatório em uma máquina que não possuía impressoras instaladas.

Em reunião em que foram abordados assuntos sugeridos pela PHE eles apresentaram as seguintes sugestões: possibilidade de incluir várias fotos para um único problema, incluir nomes dos fornecedores na lista de possíveis responsáveis pela correção, quando (e somente se) o problema for com fornecedor. Foram distribuídos 5 formulários para pessoas que estarão avaliando o sistema, foi instalado em 2 máquinas e ficamos de marcar uma próxima visita para instalar em mais 3 máquinas.

Número do Relatório: 09	**Data:** 24/07/2002
Atividades: Configuração das maquinas para teste Artifact walkthrough Starting Conference Imersão (Ethnographic practices)	**Duração:** 9h
Objetivo: avaliar e verificar o sistema, ferramentas de follow-up para o Pokayoke-flow e verificar contexto de uso do sistema	**Ciclo:** 4°
Participantes: 3 designers 3 trabalhadores	

Foi instalada uma nova versão do Pokayoke no servidor, e também foram solucionados os problemas com as Máquinas, Software e Rede da última reunião. Para tanto, duas máquinas foram formatadas e foram instalados novamente o Windows 98 e Office 97, por causa de erro no ODBC e excesso de software nestas máquinas.

Estas máquinas "supostamente" foram revisadas pelo suporte do Nied, entretanto duas apresentaram defeitos de software (pelos motivos citados acima) e a outra apresentava um defeito de Hardware na placa de rede. Esta máquina que já havia apresentado um problema de "mau contato" parou de funcionar quando fomos à fábrica na última vez.

Por sorte existiam placas da mesma marca e modelo na Delphi, contando com apoio do suporte na Delphi (Gisele) o problema foi solucionado. Outro problema solucionado foi um dos cabos de rede que não estava funcionando.

Após solucionados os problemas técnicos foi instalado o Pokayoke na máquina de mais um funcionário da qualidade (Joaquim). Este incluiu um problema na ferramenta, e a avaliou como boa. Tudo ocorreu bem, a tarefa foi realizada rapidamente, as perguntas se limitaram a novas funcionalidades do sistema que não existiam na versão em papel e também sobre os possíveis relatórios a serem gerados pelo sistema.

O sistema também foi apresentado a outros funcionários da Delphi e perguntaram sobre um possível treinamento. Eles se mostraram interessados pelo software e entenderam seu propósito, entretanto falaram sobre a necessidade de treinamento não só no software mas principalmente na metodologia do 5 passos. A possibilidade sobre treinamento já havia sido discutida anteriormente com o Cunha (RH) que aparentemente demonstrou interesse em viabilizá-lo e também acompanhar uma possível mudança de "cultura" na organização com o uso do sistema computacional.

Em uma conversa com o Robson, discutimos como o controle de tarefas é feito atualmente na Delphi, este passou uma cópia da planilha original que é usada por ele neste controle, e foi definido que será elaborado um protótipo baseado nesta planilha, e em uma próxima reunião iremos (Sofia+Robson) discutir e "lapidar" este protótipo.

Passamos o dia na fábrica, deu para acompanhar o trabalho deles. Almoçamos no restaurante e conversamos diretamente com várias pessoas.

Número do Relatório: 10-11	**Data:** 15/09/2002 22/10/2002
Atividades: Avaliação do Protótipo (10) PHE (11)	**Duração:** 4h
Objetivo: Avaliação do sistema Discutir Design da Interface	**Ciclo:** 4°
Participantes: 3 designers 7 trabalhadores	

Na primeira visita foram apresentadas as telas em papel e os usuários utilizaram o sistema. Aparentemente ele foi compreendido pelos novos usuários, que não participaram do desenvolvimento da ferramenta. Eles demonstraram também interesse em utilizar a ferramenta logo. As principais dúvidas estão relacionadas como ela poderia ser utilizada em conjunto com os fornecedores e clientes.

A princípio ficou estabelecido que em uma primeira fase eles poderiam utilizar o sistema para solucionar problemas internos pela ferramenta e externos de maneira "off-line" (via email), por ser necessária uma infra-estrutura para a instalação e manutenção do software junto aos fornecedores.

Na outra visita (11) foi aplicada avaliação heurística participativa e o formulário foi respondido por quatro participantes. Este formulário enfocava os pontos destacados na proposta de avaliação heurística participativa no que diz respeito aos aspectos do uso do software no ambiente de trabalho e também alguns aspectos técnicos. Como resultado do uso do formulário e da aplicação da PHE, destacamos:

- a necessidade de treinamento (o que já havia sido levantada anteriormente por alguns funcionários). Eles deixaram claro que as principais dificuldades não eram no manuseio do software mas sim do preenchimento do formulário independente que este seja em papel ou no sistema. Para minimizar estas dificuldades foi destacada a necessidade de desenvolver um item já previsto na ferramenta: o "help 5 passos", onde o usuário teria não um "help" de como utilizar o software mas uma descrição da metodologia dos 5 passos e também exemplos;
- dois dos usuários demonstraram a importância de desenvolver o sistema de apoio à decisão e recuperação de conhecimento;
- alguns bugs foram identificados e devidamente documentados.

Os formulários foram arquivados e serão analisados posteriormente de forma crítica. Conforme identificado na metodologia de Müller, a avaliação heurística participativa deve ser complementada por um grupo de especialistas em IHC. Espero poder fazer isto na universidade.

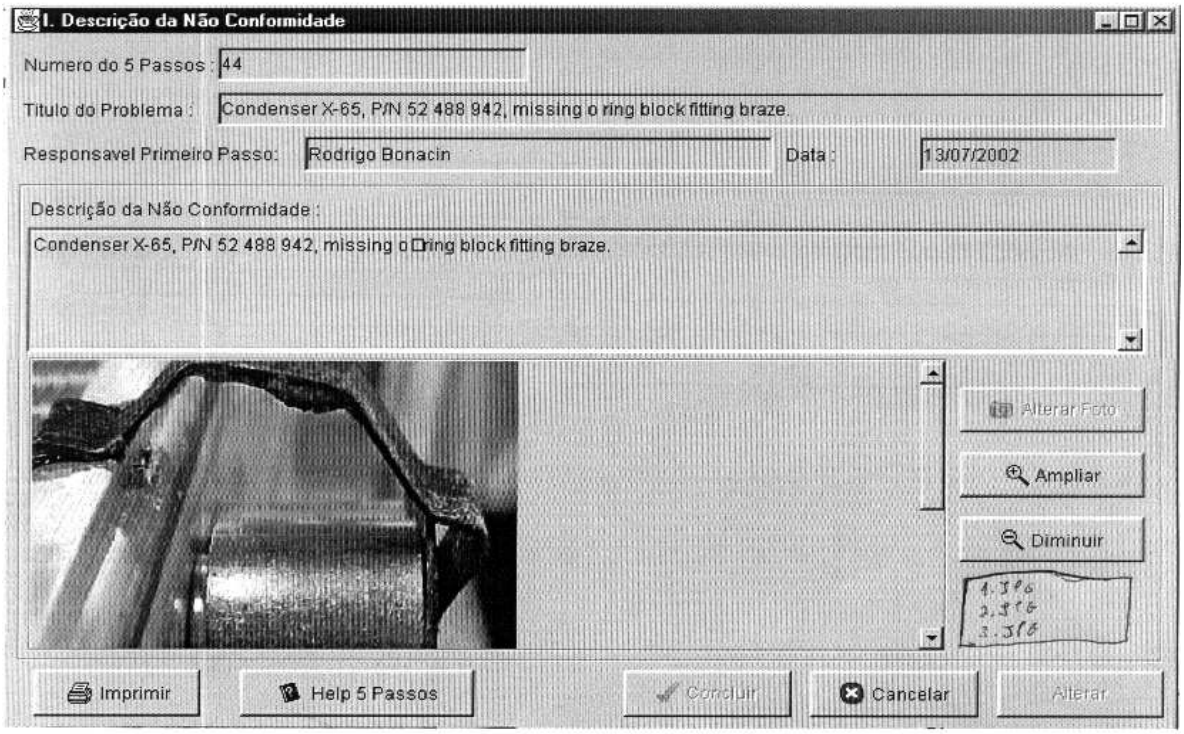


Figura A.22 – Alterações propostas para o primeiro passo

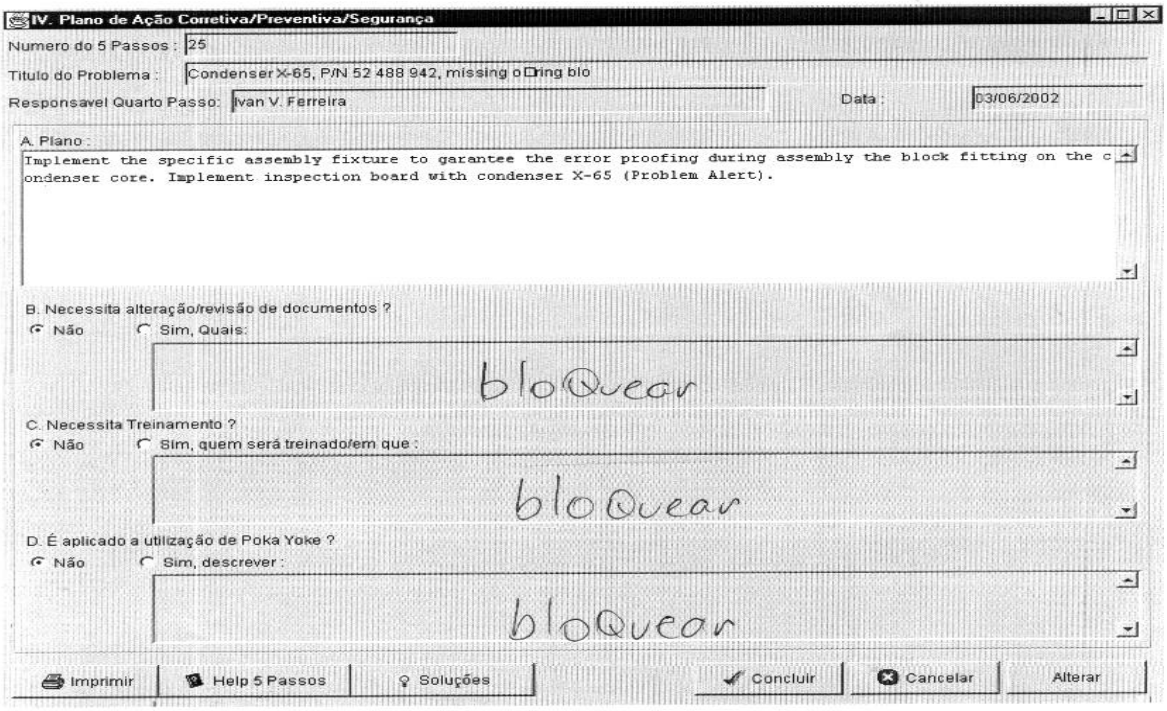


Figura A.22 – Alterações propostas para o quarto passo

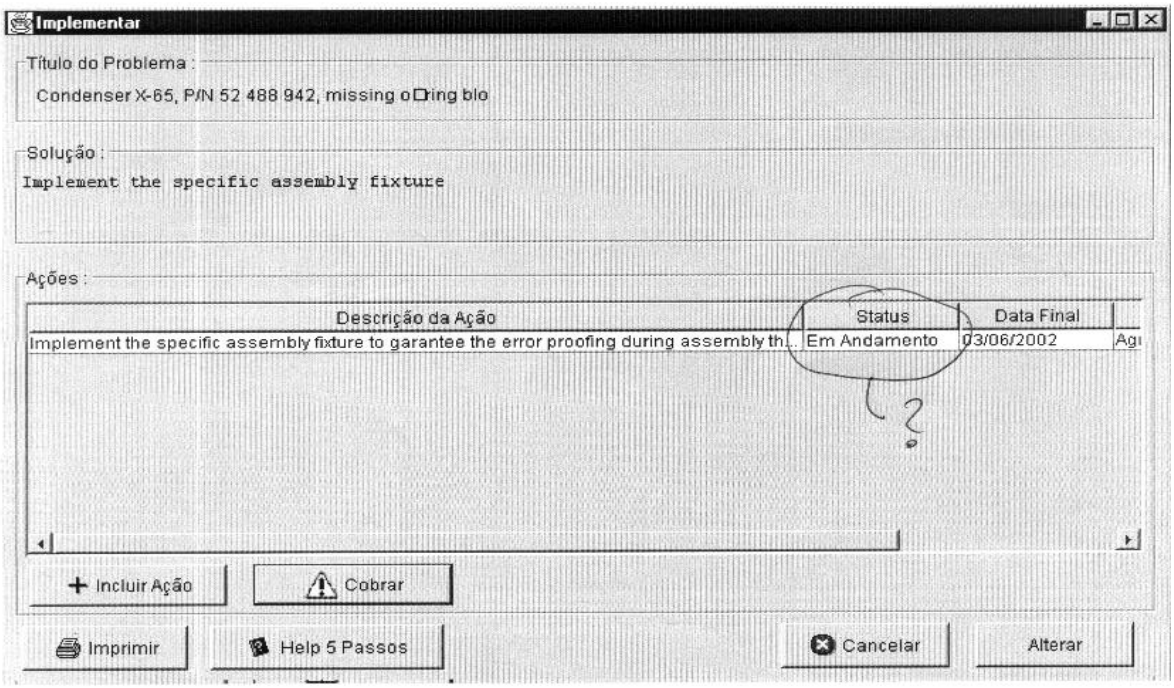


Figura A.23 – Avaliação para a tela de implementação de ações

Número do Relatório: 12	**Data:** 26/11/2002
Atividades: Avaliação o sistema - PHE ,Conferência Semiótica e Starting Conference	**Duração:** 5h
Objetivo: Avaliar o sistema, discutir alterações na interface e discutir implantação do sistema com o usuário	**Ciclo:** 5°
Participantes: 2 designers 3 trabalhadores	

O nosso novo ponto de contato na empresa agora é o Márcio (também da qualidade). O Ivan também ficou de acompanhar mais de perto a implantação do sistema na fábrica.

Como resultado desta visita destacamos:

- A necessidade de implementar formulário completo para 5 passos (imprimir todos os passos mesmo os não completos para que possam ser copiados em uma ferramenta de email, para enviar a clientes e fornecedores)
- Possibilitar que estes formulários sejam enviados a fornecedores através de emails
- Máquinas em que serão instalados o Pokayoke:
  - Ivan
  - Joaquim
  - Flávio
  - Márcio
  - Amadeu
  - André

  - Agnaldo
  - Valdei
  - Fabio
  - Marco
  - Juliana
  - Aloísio

- Definir lista de departamentos, cliente e fornecedor (Ivan)
- Limitações do Pokayoke (foi apresentado o que ainda não foi implementado)
- Versão em Inglês (o quanto antes)
- Alteração para voltar no passos (basta alterar normas)
- Colocar padrão no número do 5 passos conforme dito por eles
- Ver só os problemas relacionados com quem "logou" (falta implementar, já previsto)
- Lista de prováveis responsáveis pelas ações (Ivan)
- Opção para alguns problemas sem foto
- Imprimir cada fase separada (falta implementar)
- Help do 5 passos (Ivan ficou de passar texto)
- BUG DA DATA (URGENTE)
- Só cobrar responsável se estiver em andamento e data já estourou
- Não retorna do 5 passo
- Colocar tela em construção (para funções não implementadas)
- Data da discussão no passo 5 também está errada (30/07/01) outro bug (comportamento diferente)
- Ação status (sempre mostra 1 de n)
- Lições aprendidas (discutir mais para frente o que exatamente vai ser)
- Fazer plano para próxima reunião ...
- Não está imprimindo o campo "necessidade de treinamento" no relatório na impressora
- Ver função para (Pesquisar) quando o volume for maior
- Colocar cores nos passos (verde, amarelo, azul, vermelho ...) no follow-up
- Configura para acessar a nova máquina servidora, agora administrada pela fábrica (JagWks046)
- Comunicação entre máquinas ver com "sistemas Delphi"
- Máquinas instaladas (Ivan, Márcio)
- PokaYokeFlow (Lembrar)
  - Cópia para chefe
  - Enviar Email Automático
  - Mensagem padrão "atenção você tem uma ação imediata a ser implementada, caso já implementou favor desconsiderar"
- PokaYokeFlow (Cobrar)
  - Cobrar (dois dias antes e um dia depois)
  - 2 dias antes "Atenção, o prazo para implementação vencerá em 2 dias"
  - 1 dia depois "Atenção o prazo para implementação já venceu. O superior imediato deve entrar em contato com a qualidade"

- PokaYokeFlow
  - Alterações propostas no “gerenciador de Ações Corretivas”

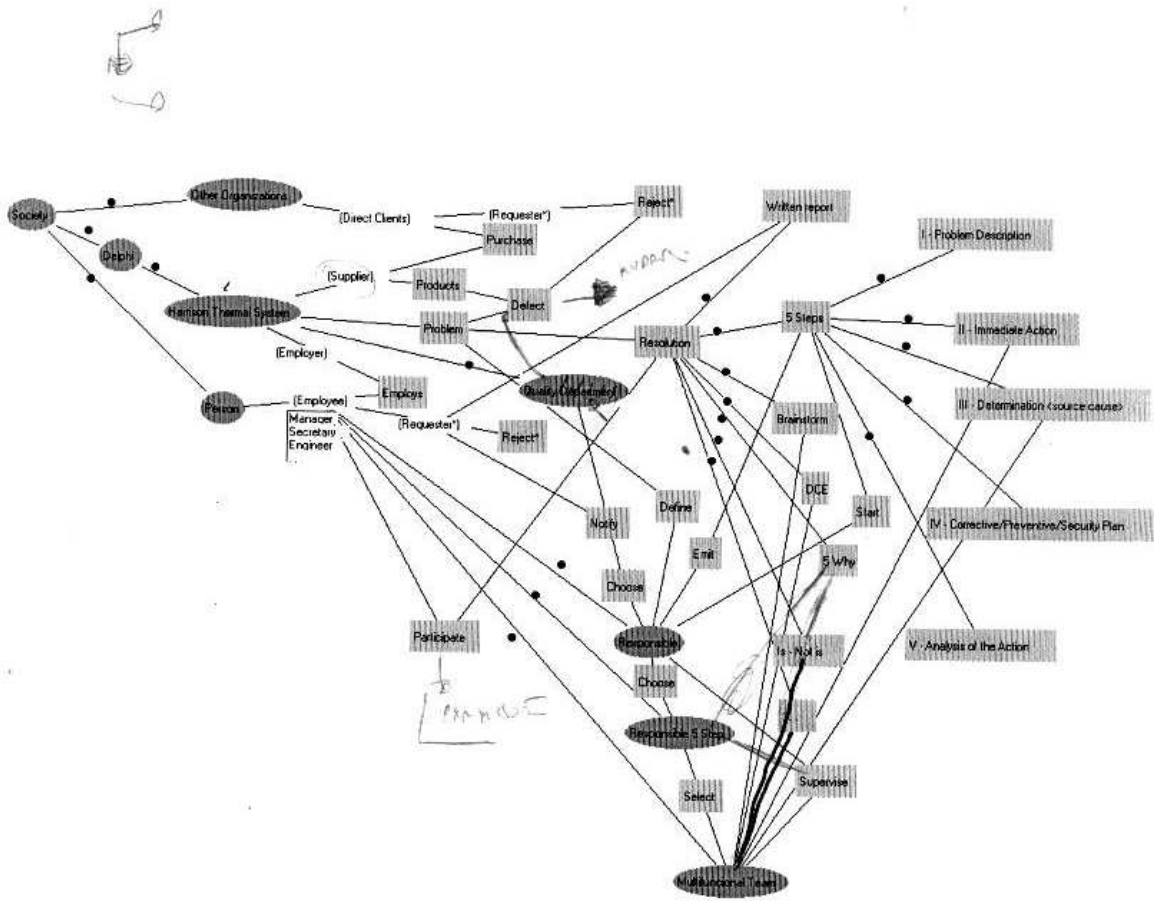


Figura A.24 – Exemplo de alterações propostas durante a Conferência Semiótica (diagrama principal)

Número do Relatório: 13-14	**Data:** 03/11/2002 17/11/2002
Atividades: Starting Conference Instalar máquinas Reunião	**Duração: 4**h (2+2)
Objetivo: Definir com o usuário como colocar o Pokayoke em produção, o treinamento e a atividade de suporte	**Ciclo:** 5°
Participantes: 3 designers 4 trabalhadores	

Como resultados desta visita:

- O Pokayoke foi instalado em 18 máquinas na empresa e no novo servidor
- Foram cadastrados os usuários do sistema, fornecedores e clientes da Delphi.

- Ficou definido que o Márcio dará treinamento para os demais usuários da empresa. Isto segue a idéia dos próprios usuários que participaram diretamente do desenvolvimento e da implantação do sistema, deveriam ser os responsáveis por treinar os demais
- No dia 17 ficou constatado um problema de interface. O sistema não dava mensagem quando o usuário tentava fechar o quarto passo sem ter ações concluídas
- Ficou estabelecido que no próximo treinamento os designers participariam somente como ouvintes e poderiam ser solicitados caso necessário
- Foi destacado que a atividade de suporte e manutenção do sistema foge ao escopo acadêmico do projeto, portanto estas tarefas deveriam ser realizadas através de contrato futuro com a empresa

Número do Relatório: 15	**Data:** 07/01/2003
Atividades: Reunião	**Duração:** 2h
Objetivo: Definir com a organização esquema para manutenção	**Ciclo:** 5°
Participantes: 2 designers 3 trabalhadores	

- Foi discutido como a manutenção do sistema seria realizada

- Foi destacado que não era necessário no momento expandir o sistema ou realizar mudanças no sistema que exigissem um alto esforço
- Ficou definido um esquema de suporte para no qual seriamos chamados para resolver problemas no sistema
- Ainda não foi definido o valor e o contrato

Número do Relatório: Universidade	**Data:** 20/11/2002
Atividades: Uso do sistema na resolução de um problema fictício PHE	**Duração:** 4h
Objetivo: Avaliar o sistema e discutir a técnica de PHE	**Ciclo:** 5°
Participantes: 5 Pesquisadores de Interfaces e SO	

Em um primeiro momento, foi realizada uma atividade para os especialistas conhecerem mais de perto o sistema. Um problema fictício de um domínio conhecido por todos foi incluído no sistema, um dos especialistas fez o papel de "emitente" (responsável pelo primeiro e último passo), outro responsável do 5 passos (responsável pelo segundo, terceiro e quarto passo) e os demais foram membros do time multifuncional. Após isso, eles preencheram o mesmo formulário que foi preenchido pelos funcionários da empresa.

Com o objetivo de complementar a aplicação da técnica de PHE (que necessita também a visão do usuário) aplicamo-la ao grupo de especialistas em IHC. Os resultados obtidos desta atividade diferenciam-se dos obtidos na organização, principalmente, porque neste caso, foram identificados mais problemas orientados à produto. Uma série de bugs e problemas técnicos que até então não estavam visíveis foram identificados pelos especialistas.

Este formulário foi utilizado durante a análise do sistema Pokayoke

As percentagens indicam o resultado da análise com 8 pessoas: 4 usuários e 4 especialistas em IHC (que não participaram do desenvolvimento do sistema):

1) O Sistema correspondeu a suas expectativas:
[0%] **Nenhuma**
[0%] **Muito Pouco**
[12,5%] **Em Partes**
[62,5%] **A Maioria**
[12,5%] **Completamente**
[12,5%] **Não Opinaram**

- Em que o Sistema não correspondeu ?
"Em termos de design diria que correspondeu completamente às minhas espectativas. O que falta é completar as funcionalidades já previstas e corrigir bugs que aparecem no uso em larga escala"
"Minhas expectativas não estão contextualizadas como as dos prováveis usuários. Não creio que seja relevante minha resposta a esta pergunta"

2) Em comparação ao 5 passos em papel, o sistema é:
[0%] **Impossível de entender**
[12,5%] **Mais difícil de entender**
[12,5%] **Equivalente**
[12,5%] **Mais fácil de entender**
[25,0%] **Muito mais fácil de entender**
[37,5%] **Não Opinaram**

- **Porque ?**
"Achei mais fácil de entender os 5 passos no sistema, porque me deu a visão geral deles, como um se relaciona com o outro. No papel estas relações ficam desconexas"
"Não tenho experiência com os 5 passos em papel"
"Não tenho como comparar. Não conheço o procedimento em papel"
"Acho mais difícil por ser algo novo, mas ainda não fiz nenhum preenchimento na prática"
"Ferramenta simples de uso rápido permitindo que você preencha todos os campos obrigatoriamente"
"Equivalente, mas requer uso prático para facilitar o entendimento"

3) Quanto à necessidade de treinamento:
[12,5%] **Estritamente necessária**
[12,5%] **Necessária**
[50,0%] **Pode não ser necessária caso seja adicionada funcionalidades de ajuda/help no próprio software**
[12,5%] **Poderia agregar algum valor**

[0%] **Totalmente Desnecessária**
[12,5%] **Não Opinaram**

- **Porque ?**

"Acredito que o treinamento não seja necessário porque os usuários participaram do design e o produto (o sistema) faz parte do cotidiano de trabalho deles"
"Depende da experiência anterior dos usuários"
"Necessário para esclarecer prováveis dúvidas"
"Necessário para conhecimento pleno da ferramenta"
"Não é necessário se a pessoa conhecer o procedimento de análise de problemas em 5 passos"

4) O software poderia ser utilizado em larga escala na empresa:
[0%] **Nunca**
[0%] **Com muitas alterações/novas funcionalidades**
[25%] **Com algumas alterações/novas funcionalidades**
[75%] **Esta versão já pode ser utilizada, porém, alterações e inclusões de novas funcionalidades devem ser feitas logo que possível**
[0%] **Esta versão já pode ser utilizada, incluir novas funcionalidades que podem agregar valor**
[0%] **Não Opinaram**

- **Que alterações e inclusões de funcionalidades deveriam ser feitas ?**

"Devem ser incluídas as funcionalidades já previstas como os 5 porques, lições aprendidas, etc. Também é fundamental a correção dos bugs que só são visíveis quando o sistema é colocado em uso pra valer"
"Inclusão do sistema de ajuda"
"Arrumar o tamanho dos textos no Brainstorm, informar a data/hora em que uma idéia foi postada, alimentar o help do sistema"
"Gerenciamento de respostas, gerenciamento de 5passos a responder"

5) Na versão atual este software tem:
[0%] **Muitos "bugs"/falhas**
[50%] **Alguns "bugs"/falhas**
[12,5%] **Está na média dos software que normalmente trabalho**
[0%] **Raramente percebi "bugs"/falhas no sistema**
[25%] **Não percebi nenhum "bug"/nenhuma falha no sistema**
[12,5%] **Não Opinaram**

- **Cite os "bugs"/falhas que mais incomodaram ?**

"Não pode ver todo o texto de uma idéia enviada em ação imediata e o botão cancelar não tem funcionalidade (com a semântica de fechar a janela apenas)"
"Problemas com o servidor da aplicação"
"O servidor trava de vez em quando"
"O problema da memória do ambiente de implementação (que joga o sistema para fora do ar)"

6) Que funcionalidades extra você gostaria que o Sistema tivesse?
"Banco de dados com lições aprendidas, demonstrar uso do sistema de análise de problemas em 5 passos"
"Gráficos de 5 passos abertos/fechados por fornecedor"
"Adicionar planilha de recuperação de custo para cada problema"
"Seria interessante que houvesse um encadeamento de idéias: exibir um grafo que indique que uma idéia deu origem a outra"
"Quem fez a ação poderia concluí-la"

7) Entre as funcionalidades abaixo, qual você acha que é mais relevante para seu trabalho no momento, ou seja, o que deve ser adicionado primeiro à ferramenta (enumere)?

[2,2,,2,1] **Mecanismo para gerência de conhecimento (gerenciar os problemas que ocorreram anteriormente na fábrica, sua resolução e o processo que levou a solução)**

[1,1,2,4,,] **Ajuda/Help do sistema e do 5 passos (explicando como utilizar o sistema e o cinco passos)**

[4,4,1,3,,] **Funcionalidades para gerenciar o acesso externo aos problemas**

[3,3,,1,,] **Ferramentas para apoio à decisão (o sistemas estaria "apontando" soluções para problemas e analisando soluções propostas)**

- **Alguma destas funcionalidades você acha desnecessária?**

"Mecanismo de gerenciamento ao vencimento das datas previstas para fechamento do 5passos"

Exemplos do Terceiro Passo da Análise Semântica Agrupamento de Candidatos

Estes fragmentos foram construídos durante o primeiro ciclo de prototipação e foram utilizados para construir os primeiros diagramas de ontologia. Os fragmentos são apresentados, neste anexo, com o objetivo de exemplificar o terceiro passo da análise durante o design do sistema Pokayoke.

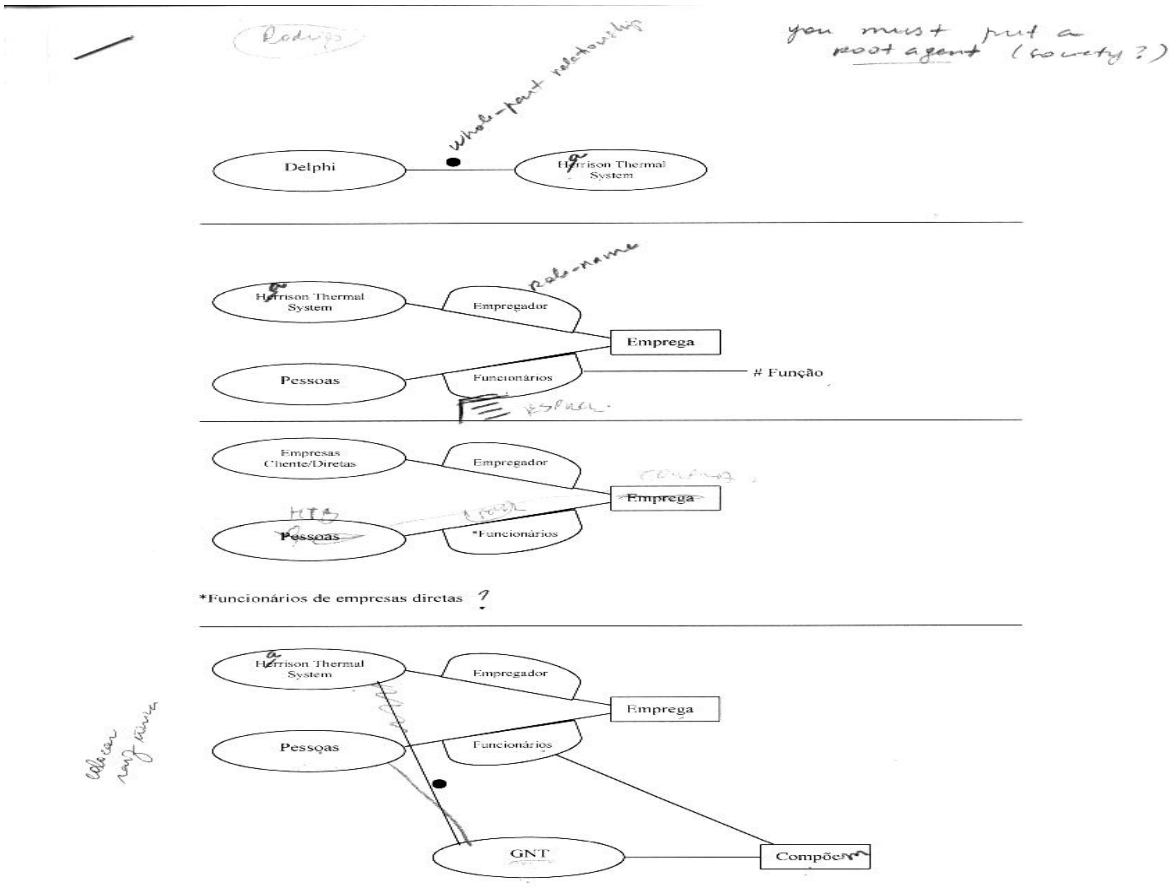


Figura A.25 – Exemplos de fragmentos de digramas de ontologia (parte I)

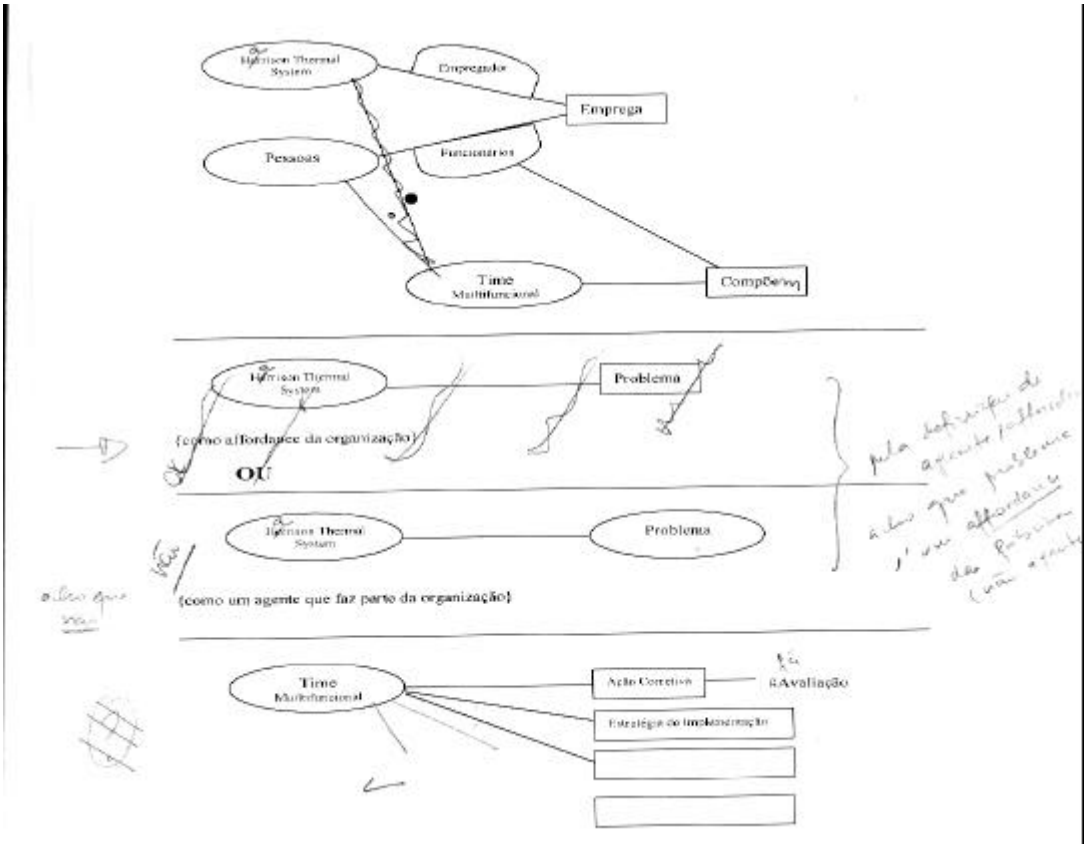


Figura A.26 – Exemplos de fragmentos de digramas de ontologia (parte II)

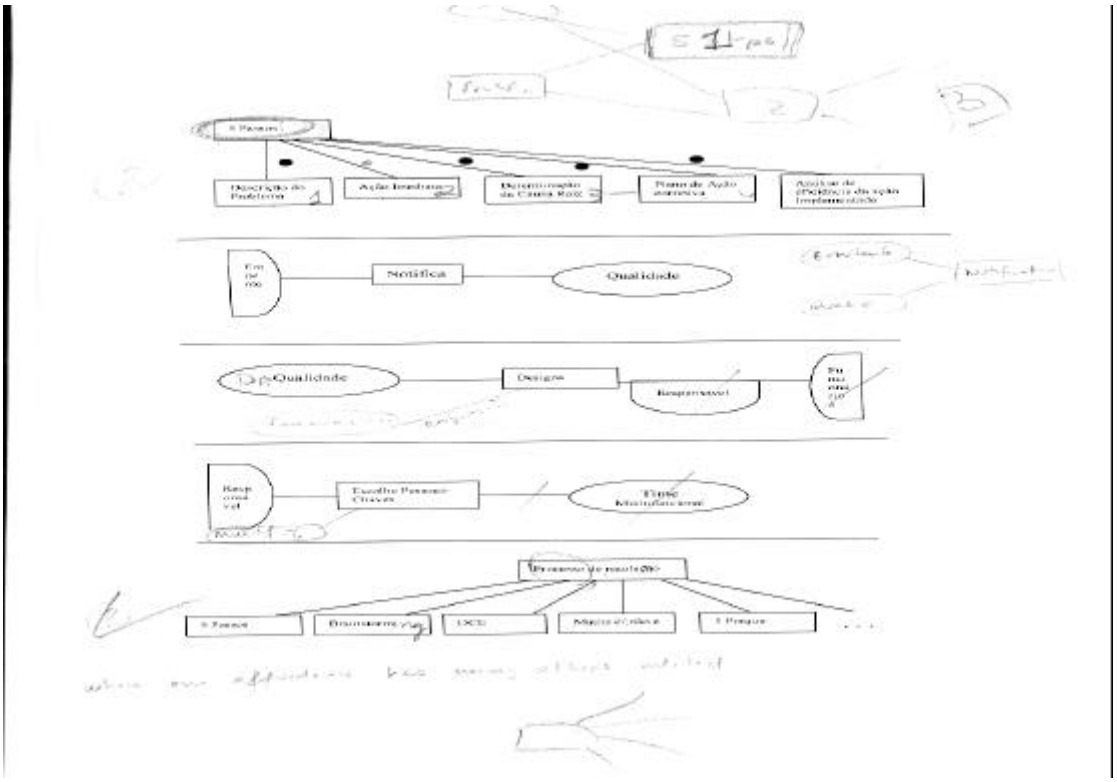


Figura A.27 – Exemplos de fragmentos de digramas de ontologia (parte III)

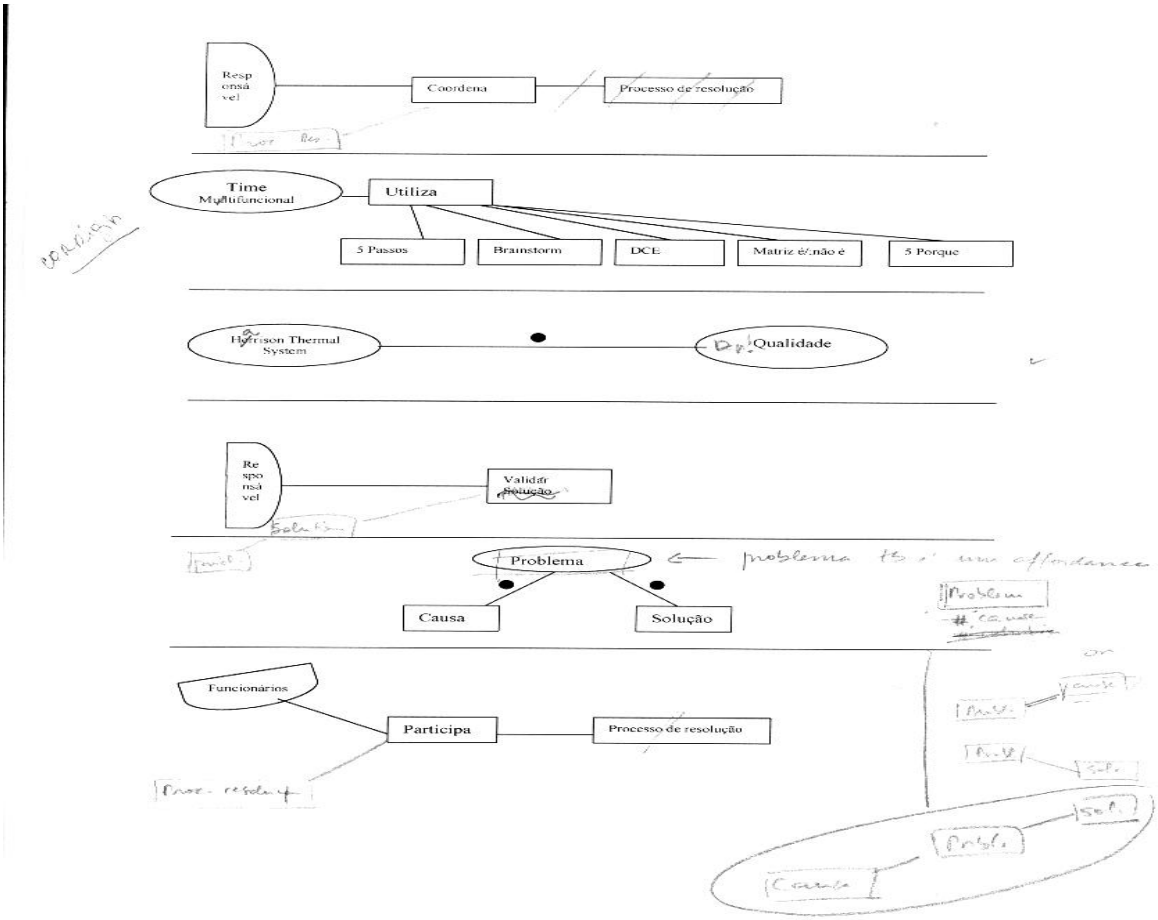


Figura A.28 – Exemplos de fragmentos de digramas de ontologia (parte IV)

Anexo B: Modelos do Pokayoke

B.1 Diagrama de Ontologia

O diagrama de ontologia, que descreve o contexto do sistema Pokayoke foi dividido em seis partes para facilitar a sua impressão e manipulação. A primeira parte, pode ser vista como um diagrama "principal" que mostra a relação do cinco passos com os demais elementos existentes na fábrica. Os outros diagramas descrevem o que existe em cada um dos passos. As partes podem ser agrupadas em um só diagrama ligando os elementos marcados com "*" dos diagramas correspondentes a cada passo com os elementos de mesmo "nome" no diagrama principal.

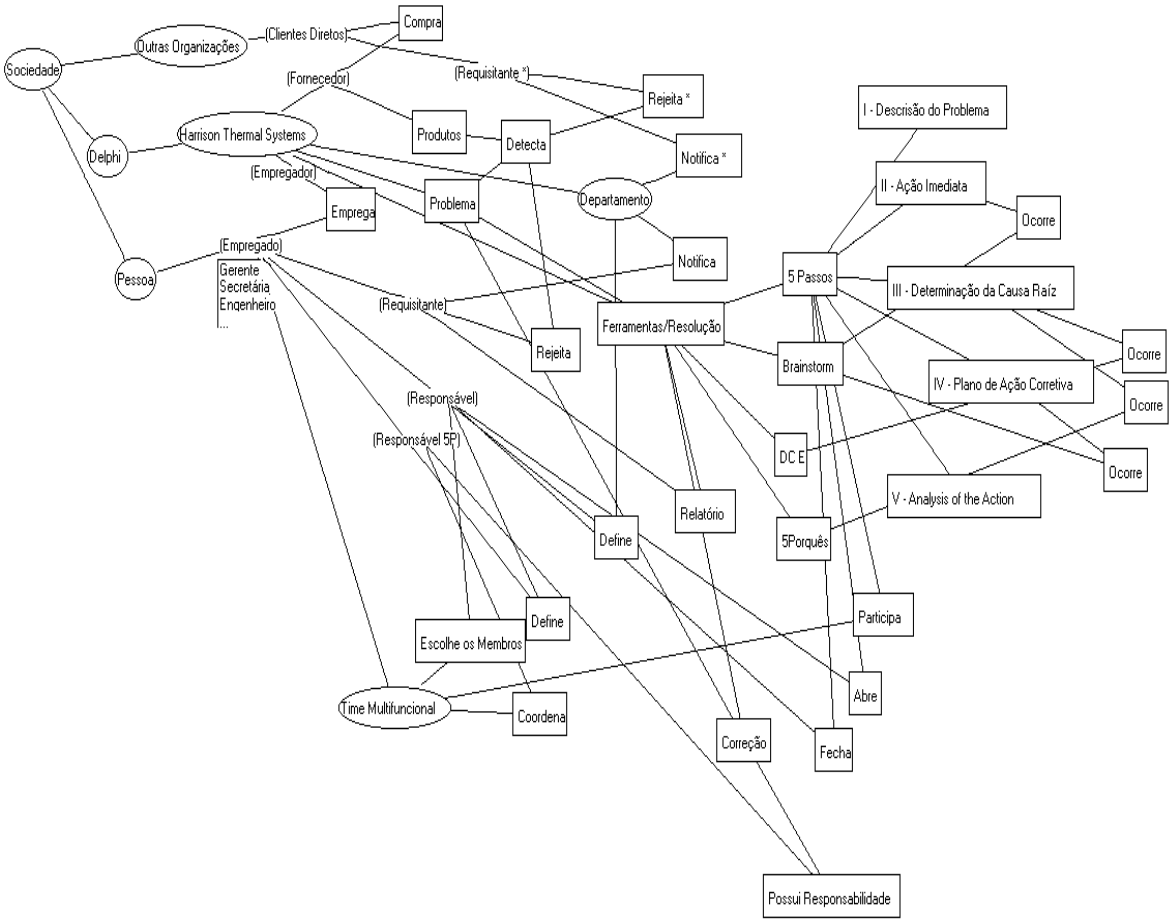


Figura B1 - Diagrama de ontologia "principal"

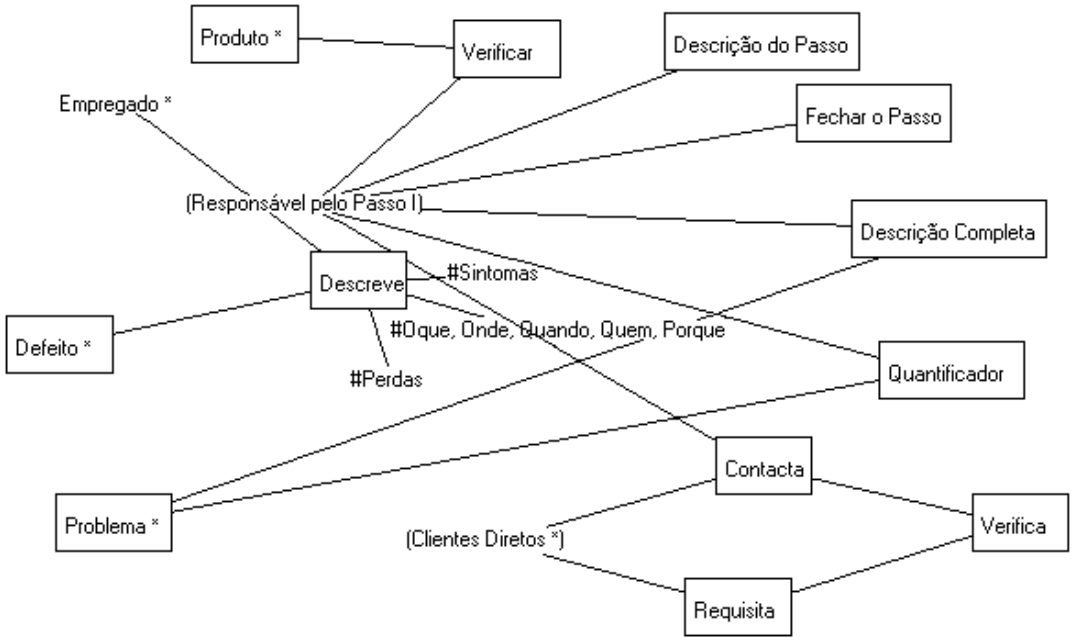


Figura B2 - Parte do diagrama para o primeiro passo

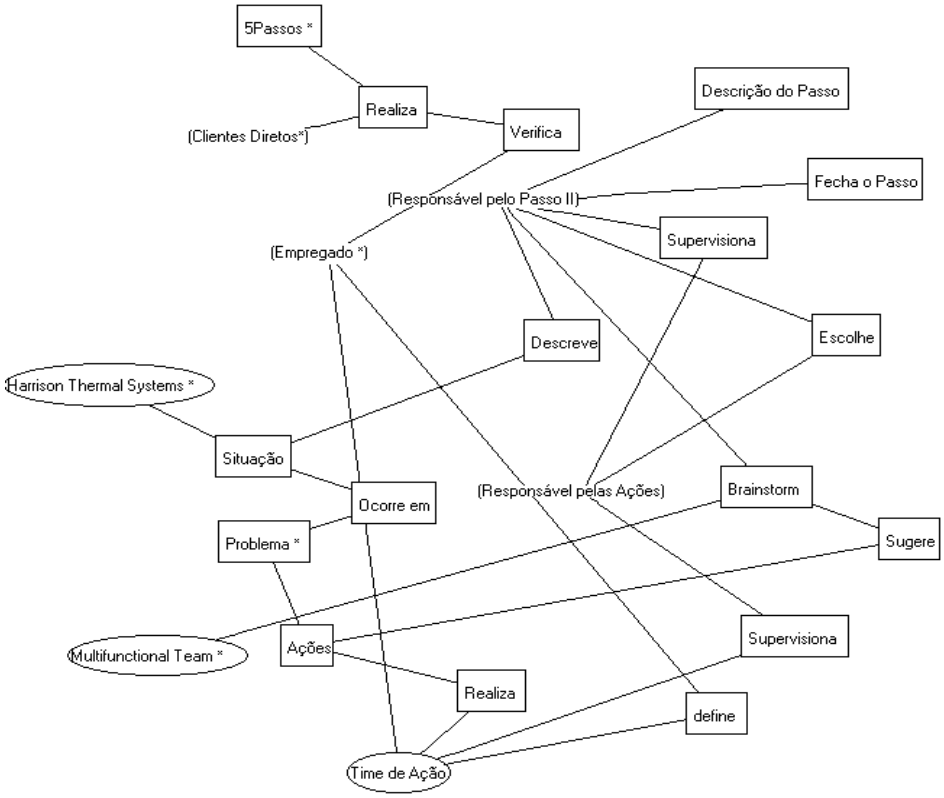


Figura B3 - Parte do diagrama para o segundo passo

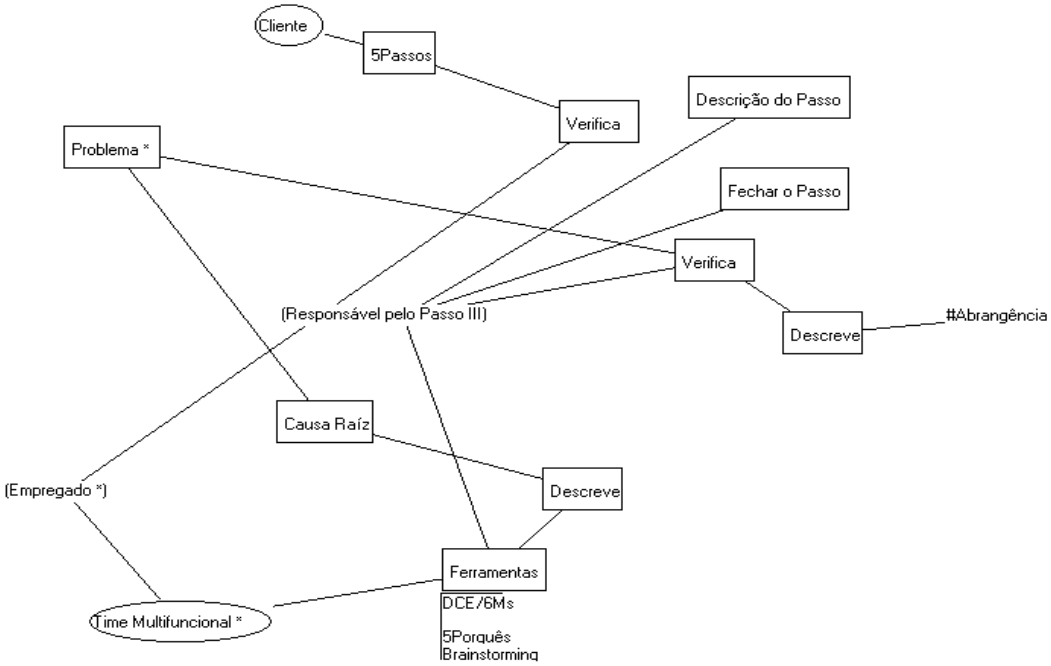


Figura B4 - Parte do diagrama para o terceiro passo

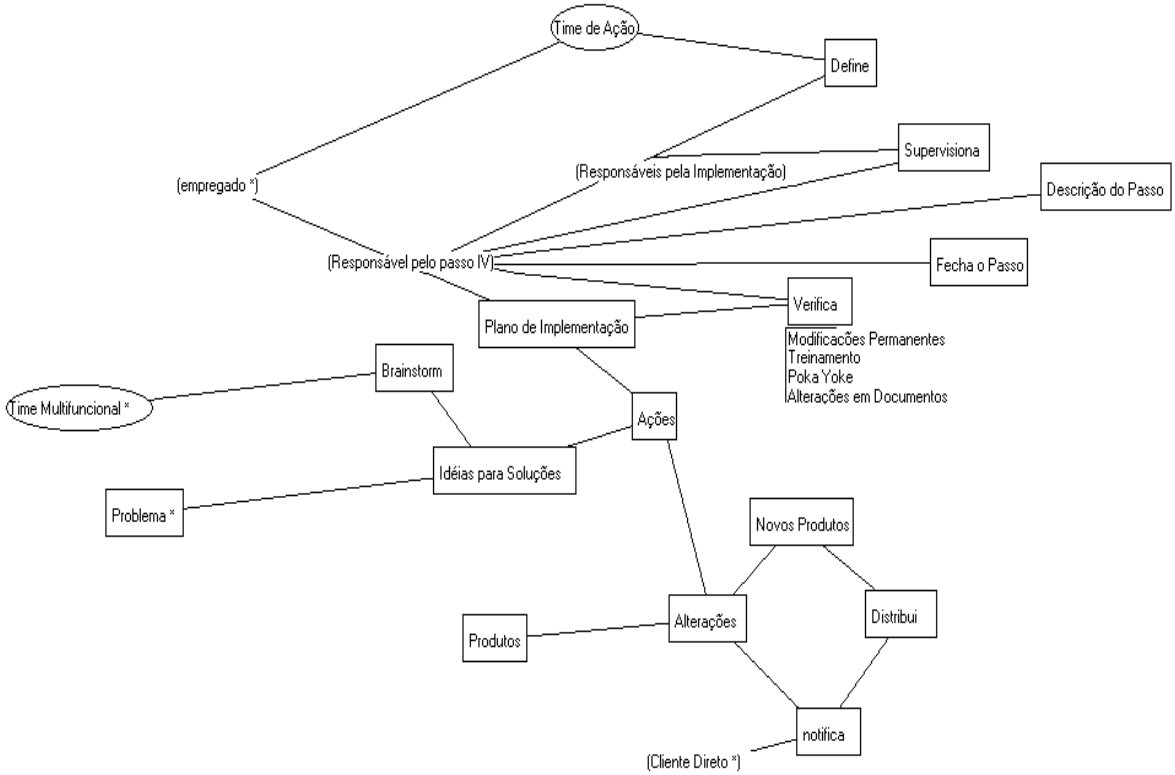


Figura B5 - Parte do diagrama para o quarto passo

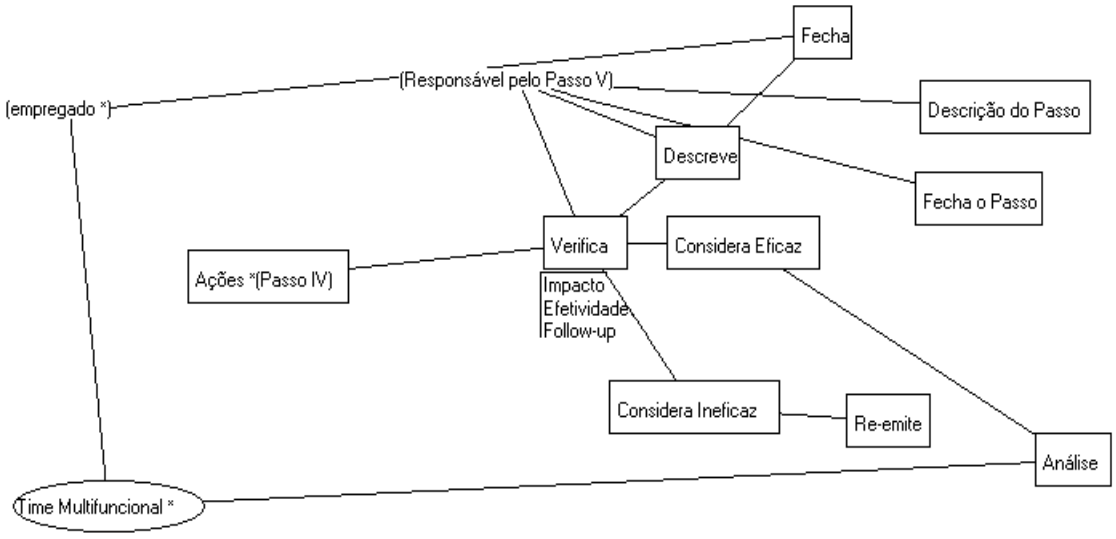


Figura B6 - Parte do diagrama para o quinto passo

Tabela B1 - Principais normas que especificam a dinâmica do procedimento de cinco passos

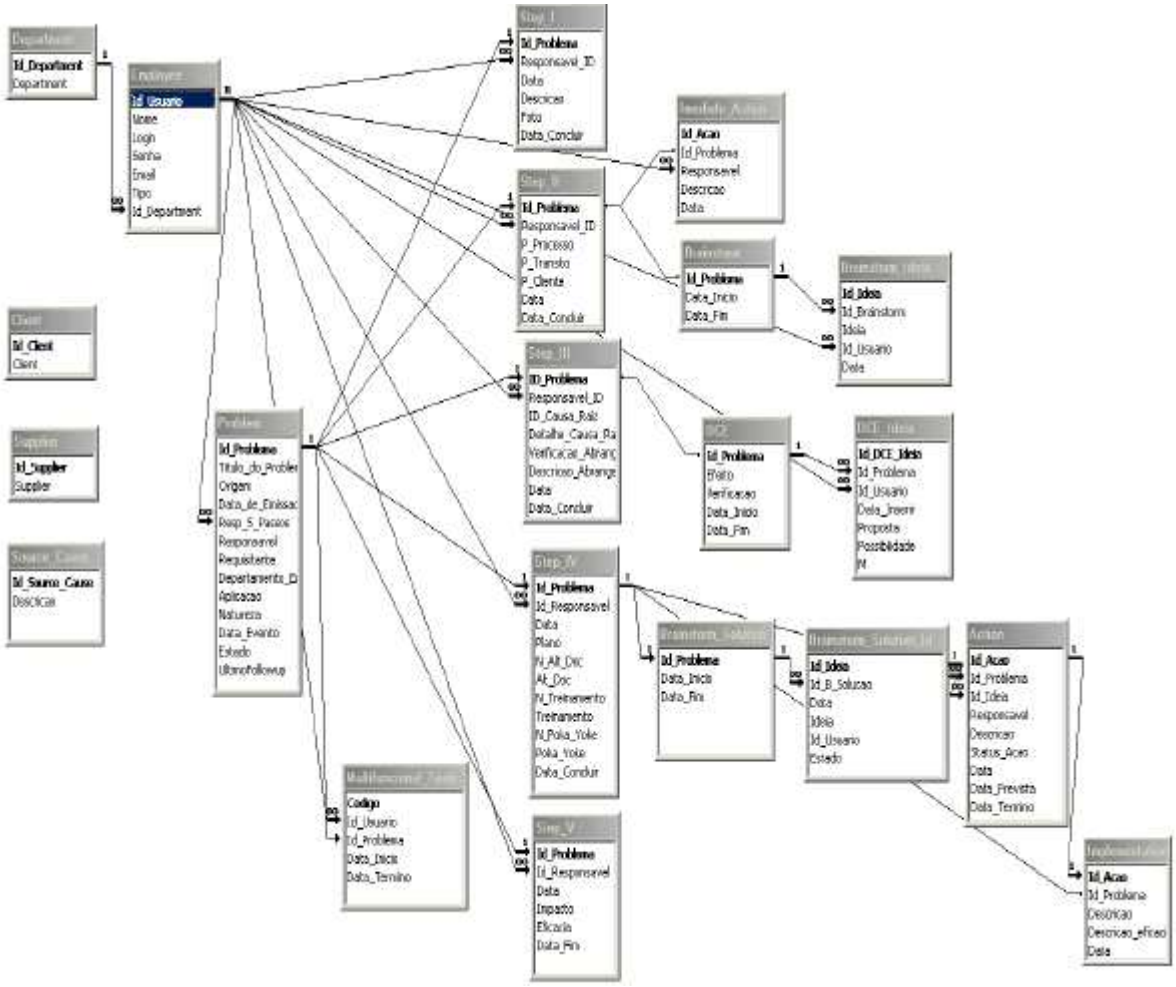
Affordance	Norma
Relatório (figura B1)	É permitido que qualquer funcionário da fábrica emita relatórios de não conformidade
Time multifuncional (figura B1)	Uma vez selecionado, é permitido que qualquer funcionário da fábrica participe de um time multifuncional
Escolhe membros (figura B1)	O responsável deve escolher os membros do time multifuncional
Abre (figura B1)	Funcionários só deve abrir cinco passo referentes ao seu departamento, com exceção da qualidade que poderá abrir cinco passos para todos os departamentos
Define (figura B1)	O "responsável" é obrigado definir o "responsável pelo 5 passos"
Participa (figura B1)	Os membros do time multifuncional podem visualizar e participar do processo de resolução de problemas (como é definido mais adiante)
Participa (figura B1)	O "responsável" e o "responsável pelo 5passos" são obrigados participar da resolução de um problema
Participa (figura B1)	Funcionários que não são membros do 5passos não podem participar do processo de resolução de problemas
Participa (figura B1)	Os funcionários podem visualizar somente os problemas de sua "planta"
Participa (figura B1)	Gerentes e auditores devem ter acesso a todos os problemas
Fecha (figura	Somente o "responsável" pode fechar o 5 passos

B1)	
Notifica (figura B1)	O requisitante é obrigado a notificar seu departamento, se o requisitante for externo este deve notificar o departamento de qualidade
Cliente Direto (figura B1)	Somente pessoas jurídicas são "clientes diretos"
Responsável (figura B1)	Quem "abre" o 5 passos é obrigatoriamente o "responsável"
Coordena (figura B1)	O "responsável pelo 5 passos" tem obrigação de coordenar o time multifuncional
Possui responsabilidade (figura B1)	O "responsável pelo 5 passos" possui responsabilidade sobre a correção do problema
Ação Imediata (figura B1)	O "segundo passo" só poderá ser iniciado se o primeiro estiver completo
Determinação da Causa Raiz (figura B1)	O "terceiro passo" só poderá ser iniciado se o segundo estiver completo
Plano de Ação Corretiva (figura B1)	O "quarto passo" só poderá ser iniciado se o terceiro estiver completo
Análise de Implantação e Eficácia da Ação (figura B1)	O "quinto passo" só poderá ser iniciado se o quarto estiver completo As ações do "quarto passo" devem estar concluídas para serem analisadas no quinto passo
Análise de Implantação e Eficácia da Ação (figura B1)	Uma vez concluído o quinto passo não são permitidas mais alterações na descrição dos passos anteriores
Responsável pelo passo I (figura B2)	O "responsável" (figura B2) obrigatoriamente o "responsável pelo passo I"
Verificar (figura B2)	O "responsável pelo passo I" é obrigado verificar o produto com problema. Ele poderá tirar fotos do produto
Descrição Completa (figura B2)	O "responsável pelo passo I" é obrigado a fazer a descrição completa do problema
Verifica (figura B2)	O "responsável pelo passo I" é obrigado verificar o problema com o cliente
Fechar o Passo (figura B2)	Somente o "responsável pelo passo I" tem permissão de fechar o primeiro passo
Responsável pelo passo II (figura B3)	O "responsável pelo 5passos" (figura B2) é obrigatoriamente o "responsável pelo passo II"
Verifica (figura	O "responsável pelo passo II" tem obrigação de verificar o cinco passos

B3)	com o cliente
Escolhe (figura B3)	O “responsável pelo passo II” tem obrigação de escolher os “responsáveis pelas ações”
Supervisiona (figura B3)	O “responsável pelo passo II” tem obrigação de supervisionar os “responsáveis pelas ações”
Responsável pelas ações (figura B3)	Somente um grupo de funcionários tem permissão para assumir a responsabilidade sobre as ações
Brainstrom (figura B3)	Somente os membros do time multifuncional tem permissão para dar idéias sobre a ação imediata a ser tomada
Define (figura B3)	O “responsável pelas ações” deve definir o “time de ações”
Time de Ações (figura B3)	Qualquer funcionário pode ser chamado para participar do time de ações
Supervisiona (figura B3)	O “responsável pelas ações” tem a obrigação de supervisionar o time de ações
Sugere (figura B3)	O “responsável pelo passo” deve escolher as ações sugeridas pelo brainstorm
Responsável pelo passo III (figura B4)	O “responsável pelo 5passos” (figura B2) é obrigatoriamente o “responsável pelo passo III”
Verifica (figuraB4)	O “responsável pelo 5passos” é obrigado a verificar o cinco passos do cliente
Descreve (figuraB4)	O “responsável pelo 5passos” é obrigado a descrever a abrangência do problema
Ferramentas (figuraB4)	Somente os membros do time multifuncional têm permissão para dar idéias em um DCE
Ferramentas (figuraB4)	Somente “responsável pelo 5passos” pode realizar a verificação das idéias no DCE
Fechar o Passo (figura B4)	Somente o “responsável pelo passo III” tem permissão de fechar o terceiro passo
Responsável pelo passo IV (figura B5)	O “responsável pelo 5passos” (figura B2) é obrigatoriamente o “responsável pelo passo IV”
Supervisiona (figura B5)	O “responsável pelo passo IV” tem obrigação de supervisionar o “responsáveis pelas implementações”
Plano de Implementação (figura B5)	O “responsável pelo passo IV” tem obrigação de descrever o plano de implementação
Verificar (figura B5)	O “responsável pelo passo IV” tem obrigação de verificar a necessidade de “modificações permanentes”, “treinamento”, “Poka yoke” e “alterações em documentos”
Idéias para Solução (figura B5)	Somente os membros do time multifuncional tem permissão para dar idéias sobre a solução do problema na seção de brainstorm
Distribui (figura	Peças poderão ser distribuídas novamente ao cliente somente após todos

B5)	os problemas serem solucionados
Fecha o Passo (figura B5)	Somente o "responsável pelo passo IV" tem permissão de fechar o quarto passo
Responsável pelo passo V (figura B5)	O "responsável" (figura B2) é obrigatoriamente o "responsável pelo passo V"
Verifica (figura B6)	O "responsável pelo passo V" deve obrigatoriamente verificar o "impacto" e "efetividade" da ação implementada, e também verificar os "follow-up"
Considera Eficaz (figura B6)	Uma ação é considerada eficaz se atingir todos os itens destacados durante o terceiro passo
Análise (figura B6)	Uma análise da solução deverá ser conduzida se ela for eficaz
Considera Ineficaz (figura B6)	Uma ação é considerada ineficaz se não atingir todos os itens destacados durante o terceiro passo
Re-emite (figura B6)	O usuário deverá re-emitir os cinco passos se considerar a solução ineficaz

B.2 Modelo do Banco de Dados



B.3 Diagramas de Classes

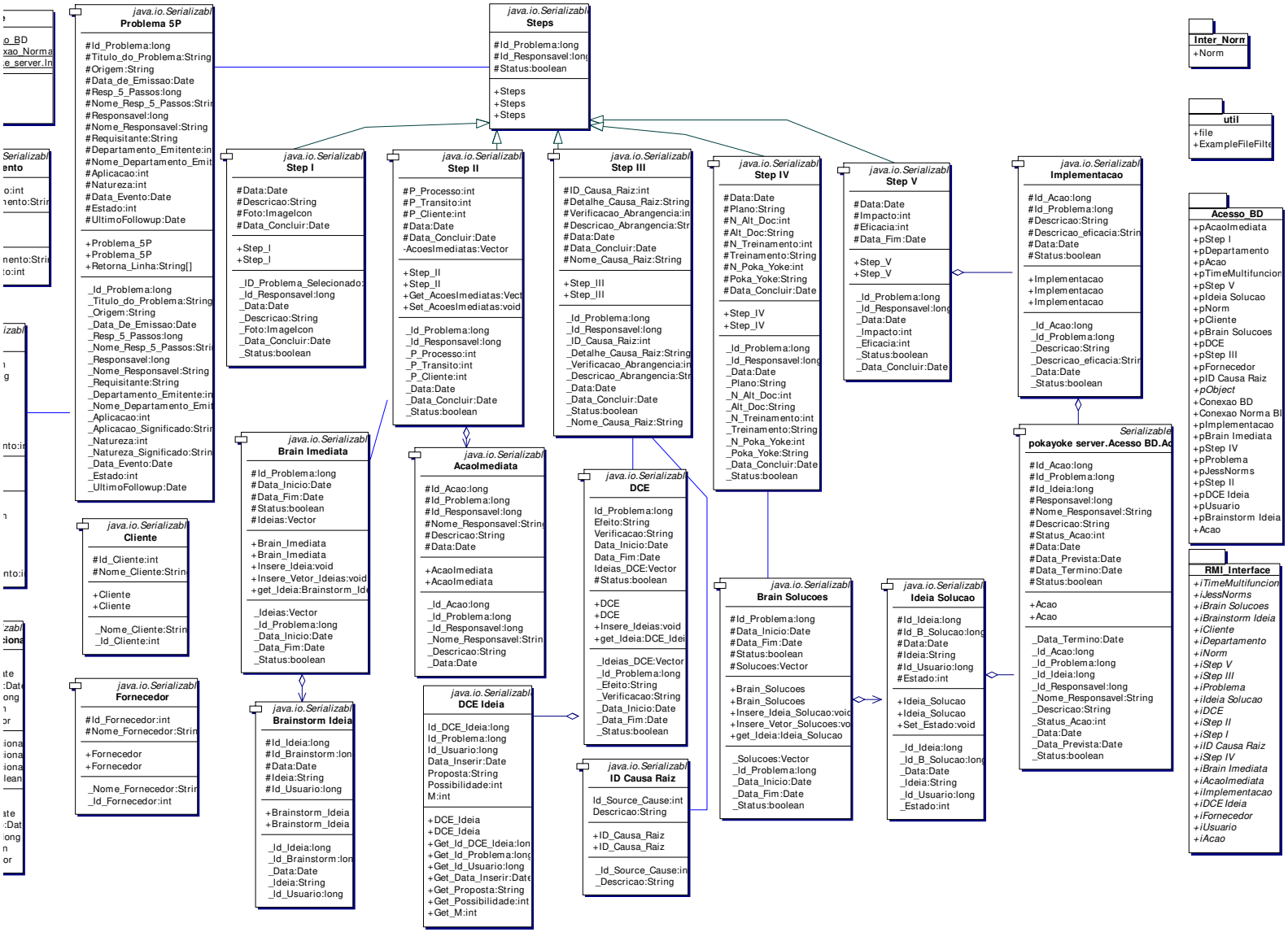
Nesta seção apresentamos os principais diagramas de classes do sistema Pokayoke (nível de implementação). Primeiro são apresentados os diagramas do servidor Pokayoke:

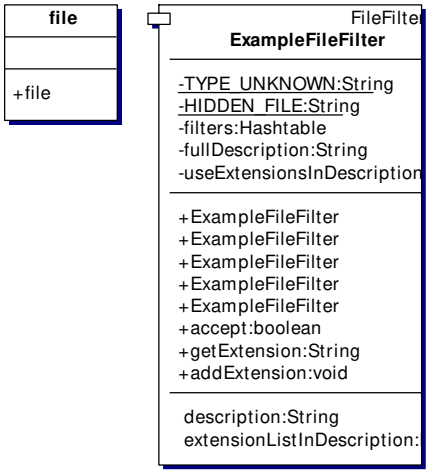
- O primeiro deles é o diagrama que escreve as classes de negócio. Estas classes especificam todos os objetos de negócio que são manipulados pelas classes controladoras e "persistidos" no banco de dados. Com exceção da classe "Pokayoke", a maioria delas são classes são "serialisáveis" pois são parâmetros e retornos das chamadas em RMI;
- O segundo diagrama contém duas classes utilizadas pelo servidor na manipulação de arquivos;
- O terceiro diagrama especifica a classe "norma";
- O quarto diagrama contém as interfaces RMI da camada de persistência utilizada pelo cliente para persistir e recuperar objetos de negócio (não seguem o padrão EJB);

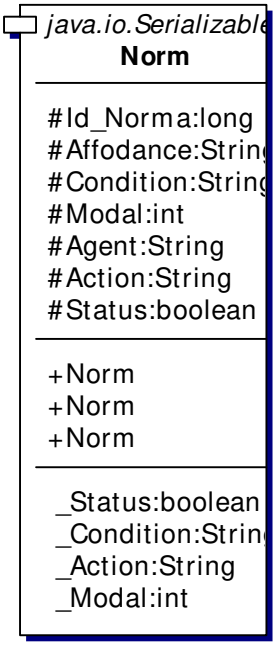
- O quinto diagrama especifica as classes da camada de persistência. Classes que manipulam objetos seguem a seguinte nomenclatura: p*Nomedoobjeto,* onde "*NomedoObjeto*" é o mesmo nome da classe de negócio que especifica o objeto que a classe da camada de persistência manipula.

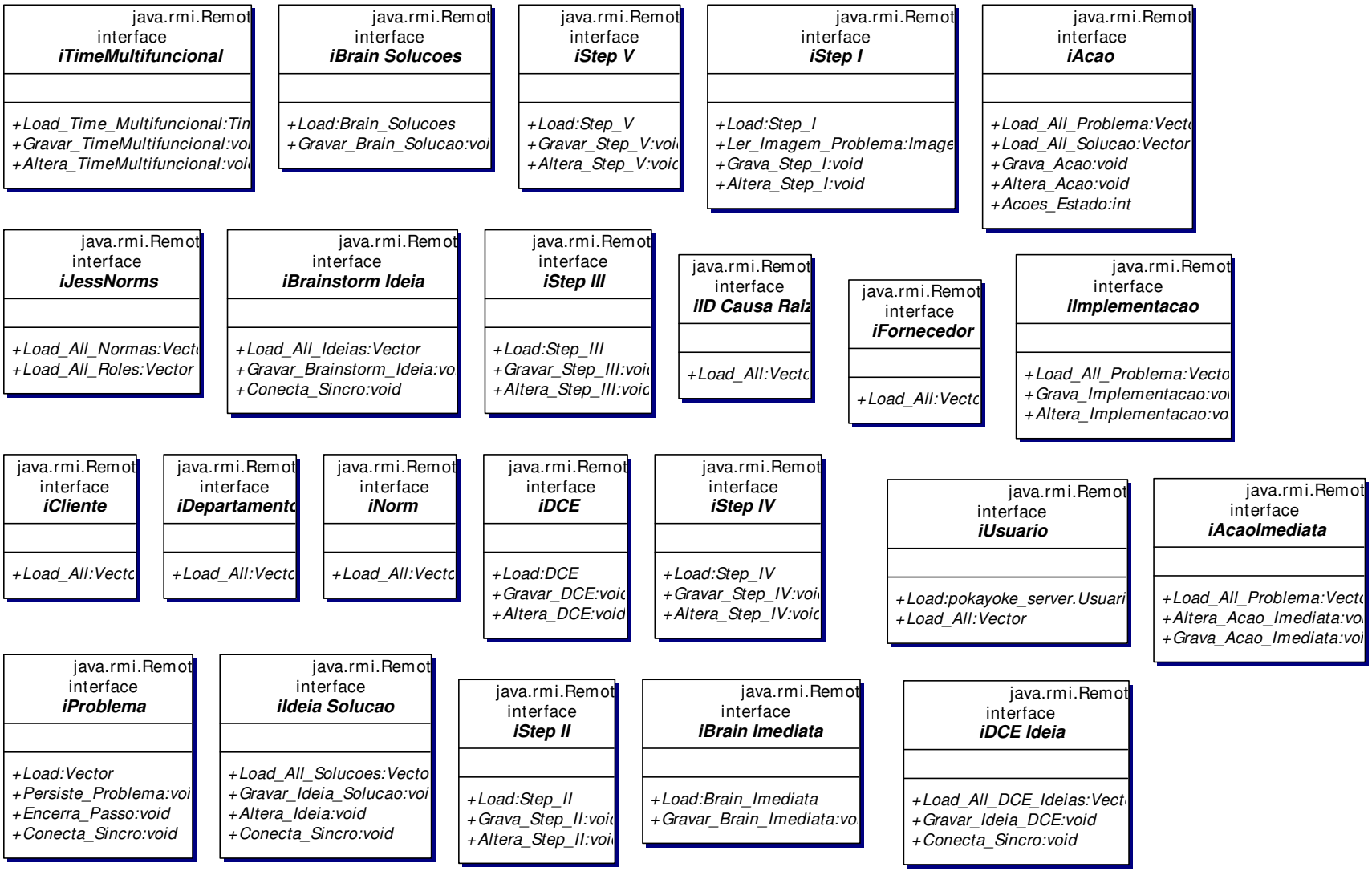
Os próximos diagramas são diagramas do lado do cliente:

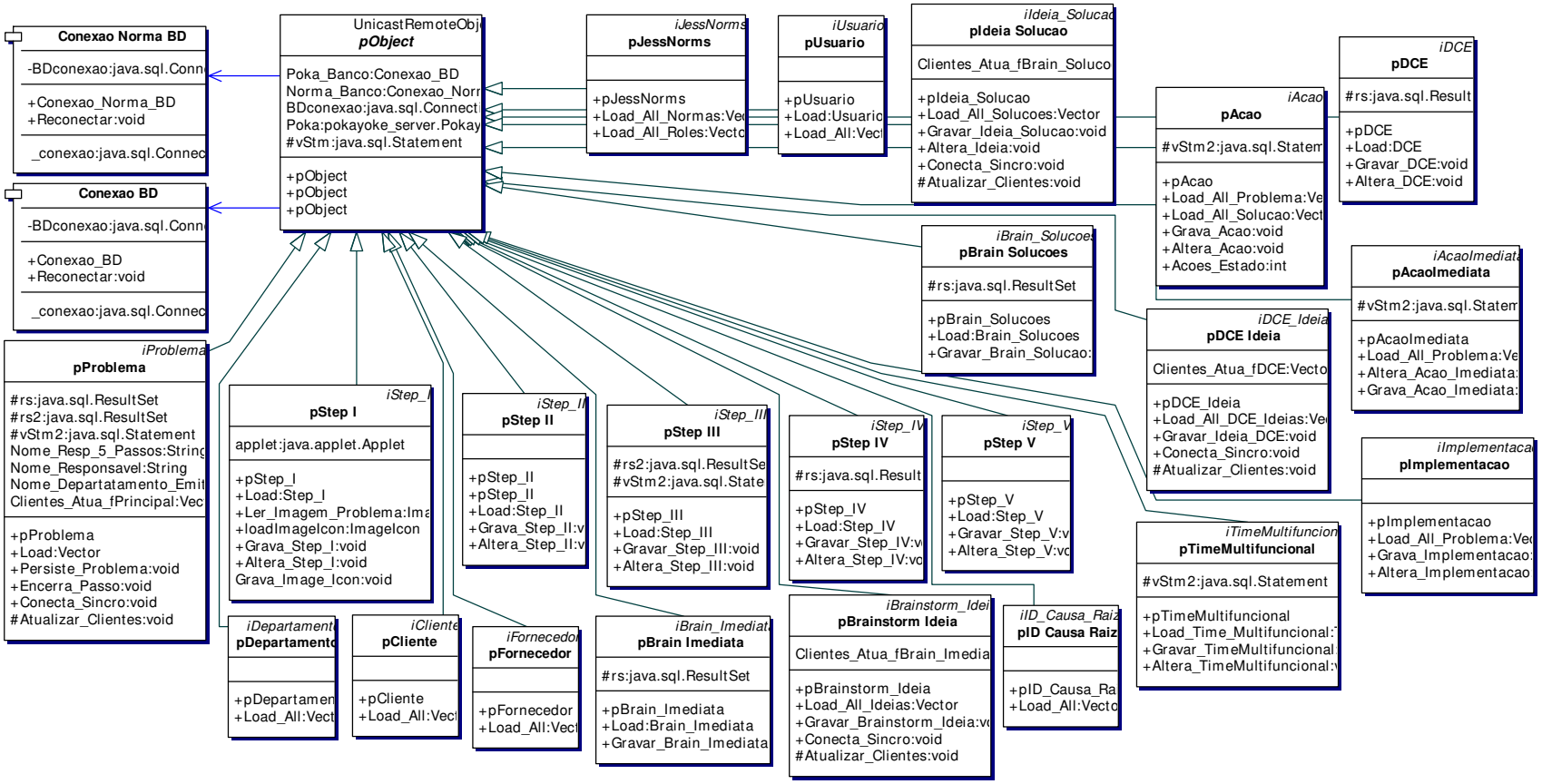
- O sexto diagrama especifica as classes controladoras. As classes seguem a seguinte nomenclatura: m*NomedoObjeto*, onde "*NomedoObjeto*" é o mesmo nome da classe de negócio que especifica o principal objeto que a classe controladora manipula. A classe "pokayoke" inicia o programa e faz a conexão com o servidor;
- O sétimo diagrama apresenta as classes de interface (telas). Para facilitar a leitura e a impressão não são apresentados os atributos, que são muitos. As comunicações entre os objetos de interface ocorrem sempre via as classes controladoras, por isso não existem associações entre as classes deste diagrama. As classes seguem a seguinte nomenclatura: f*NomedoFrame*. Junto com este diagrama, também é apresentada a classe "D_Interface", ela implementa o mecanismo de percepção e ação, ela configura a interface das demais;
- O oitavo diagrama apresenta as interfaces RMI e classes utilizadas para a sincronização entre os clientes Pokayoke;
- O nono diagrama apresenta as classes utilizadas para manipular arquivos e uma fábrica de "estilos" para montar os relatórios do cinco passos.

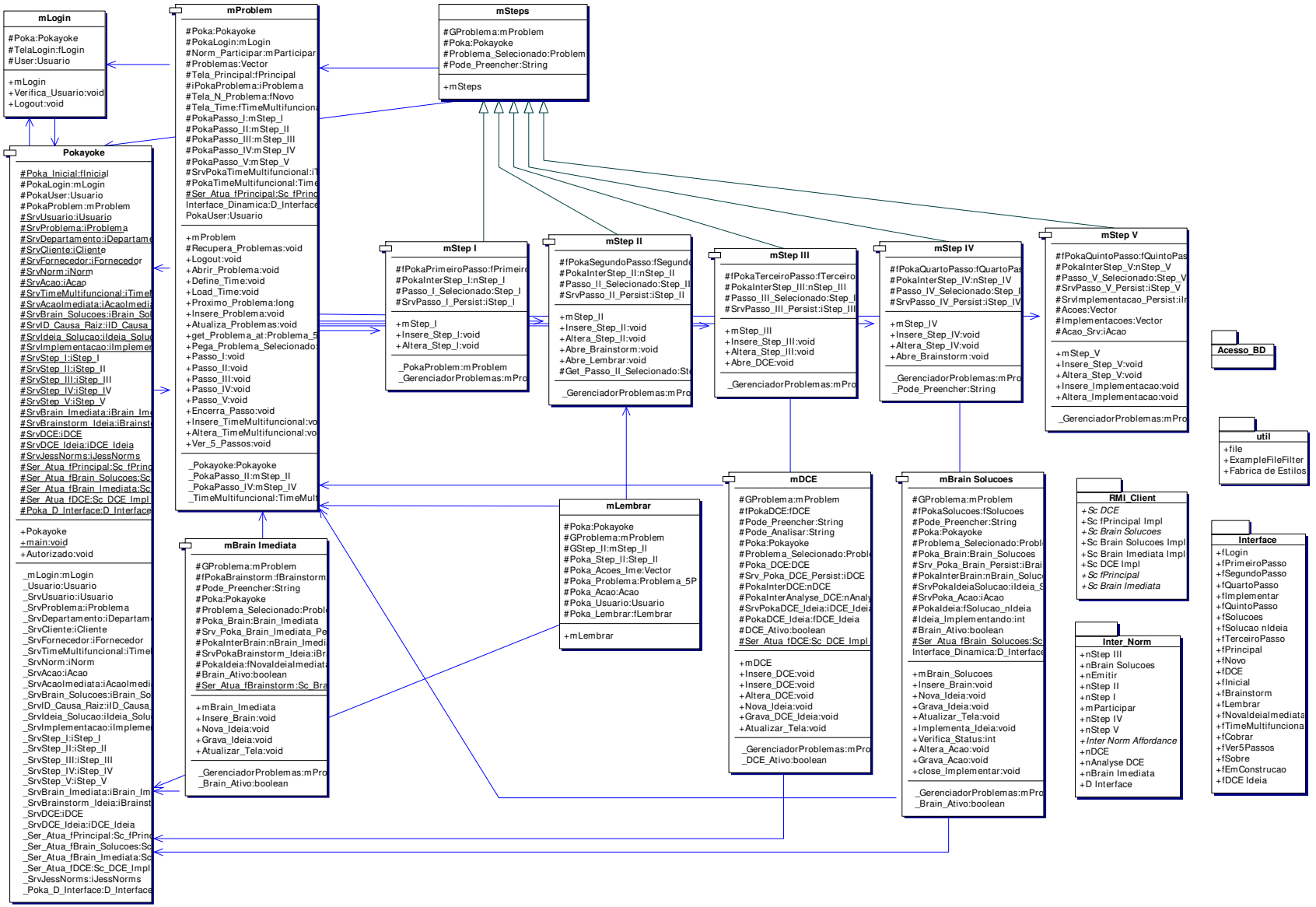

java.io.Serializabl
Problema 5P
#Id_Problema:long
#Titulo_do_Problema:String
#Origem:String
#Data_de_Emissao:Date
#Resp_5_Passos:long
#Nome_Resp_5_Passos:Strin
#Responsavel:long
#Nome_Responsavel:String
#Requisitante:String
#Departamento_Emitente:in
#Nome_Departamento_Emit
#Aplicacao:int
#Natureza:int
#Data_Evento:Date
#Estado:int
#UltimoFollowup:Date
+Problema_5P
+Problema_5P
+Retorna_Linha:String[]
_Id_Problema:long
_Titulo_do_Problema:String
_Origem:String
_Data_De_Emissao:Date
_Resp_5_Passos:long
_Nome_Resp_5_Passos:Stri
_Responsavel:long
_Nome_Responsavel:String
_Requisitante:String
_Departamento_Emitente:in
_Nome_Departamento_Emit
_Aplicacao:int
_Aplicacao_Significado:Strin
_Natureza:int
_Natureza_Significado:Strin
_Data_Evento:Date
_Estado:int
_UltimoFollowup:Date
java.io.Serializabl
Steps
#Id_Problema:long
#Id_Responsavel:long
#Status:boolean
+Steps
+Steps
+Steps
java.io.Serializabl
Step I
#Data:Date
#Descricao:String
#Foto:ImageIcon
#Data_Concluir:Date
+Step_I
+Step_I
_ID_Problema_Selecionado:
_Id_Responsavel:long
_Data:Date
_Descricao:String
_Foto:ImageIcon
_Data_Concluir:Date
_Status:boolean
java.io.Serializabl
Step II
#P_Processo:int
#P_Transito:int
#P_Cliente:int
#Data:Date
#Data_Concluir:Date
-AcoesImediatas:Vector
+Step_II
+Step_II
+Get_AcoesImediatas:Vect
+Set_AcoesImediatas:void
_Id_Problema:long
_Id_Responsavel:long
_P_Processo:int
_P_Transito:int
_P_Cliente:int
_Data:Date
_Data_Concluir:Date
_Status:boolean
java.io.Serializabl
Step III
#ID_Causa_Raiz:int
#Detalhe_Causa_Raiz:String
#Verificacao_Abrangencia:in
#Descricao_Abrangencia:Str
#Data:Date
#Data_Concluir:Date
#Nome_Causa_Raiz:String
+Step_III
+Step_III
_Id_Problema:long
_Id_Responsavel:long
_ID_Causa_Raiz:int
_Detalhe_Causa_Raiz:String
_Verificacao_Abrangencia:in
_Descricao_Abrangencia:Str
_Data:Date
_Data_Concluir:Date
_Status:boolean
_Nome_Causa_Raiz:String
java.io.Serializabl
Step IV
#Data:Date
#Plano:String
#N_Alt_Doc:int
#Alt_Doc:String
#N_Treinamento:int
#Treinamento:String
#N_Poka_Yoke:int
#Poka_Yoke:String
#Data_Concluir:Date
+Step_IV
+Step_IV
_Id_Problema:long
_Id_Responsavel:long
_Data:Date
_Plano:String
_N_Alt_Doc:int
_Alt_Doc:String
_N_Treinamento:int
_Treinamento:String
_N_Poka_Yoke:int
_Poka_Yoke:String
_Data_Concluir:Date
_Status:boolean
java.io.Serializabl
Step V
#Data:Date
#Impacto:int
#Eficacia:int
#Data_Fim:Date
+Step_V
+Step_V
_Id_Problema:long
_Id_Responsavel:long
_Data:Date
_Impacto:int
_Eficacia:int
_Status:boolean
_Data_Concluir:Date
java.io.Serializabl
Implementacao
#Id_Acao:long
#Id_Problema:long
#Descricao:String
#Descricao_eficacia:Strin
#Data:Date
#Status:boolean
+Implementacao
+Implementacao
+Implementacao
_Id_Acao:long
_Id_Problema:long
_Descricao:String
_Descricao_eficacia:Strin
_Data:Date
_Status:boolean
java.io.Serializabl
Brain Imediata
#Id_Problema:long
#Data_Inicio:Date
#Data_Fim:Date
#Status:boolean
#Ideias:Vector
+Brain_Imediata
+Brain_Imediata
+Insere_Ideia:void
+Insere_Vetor_Ideias:void
+get_Ideia:Brainstorm_Id
_Ideias:Vector
_Id_Problema:long
_Data_Inicio:Date
_Data_Fim:Date
_Status:boolean
java.io.Serializabl
AcaoImediata
#Id_Acao:long
#Id_Problema:long
#Id_Responsavel:long
#Nome_Responsavel:Strin
#Descricao:String
#Data:Date
+AcaoImediata
+AcaoImediata
_Id_Acao:long
_Id_Problema:long
_Id_Responsavel:long
_Nome_Responsavel:Strin
_Descricao:String
_Data:Date
java.io.Serializabl
DCE
Id_Problema:long
Efeito:String
Verificacao:String
Data_Inicio:Date
Data_Fim:Date
Ideias_DCE:Vector
#Status:boolean
+DCE
+DCE
+Insere_Ideias:void
+get_Ideia:DCE_Idei
_Ideias_DCE:Vector
_Id_Problema:long
_Efeito:String
_Verificacao:String
_Data_Inicio:Date
_Data_Fim:Date
_Status:boolean
java.io.Serializabl
Brain Solucoes
#Id_Problema:long
#Data_Inicio:Date
#Data_Fim:Date
#Status:boolean
#Solucoes:Vector
+Brain_Solucoes
+Brain_Solucoes
+Insere_Ideia_Solucao:void
+Insere_Vetor_Solucoes:vo
+get_Ideia:Ideia_Solucao
_Solucoes:Vector
_Id_Problema:long
_Data_Inicio:Date
_Data_Fim:Date
_Status:boolean
java.io.Serializabl
Ideia Solucao
#Id_Ideia:long
#Id_B_Solucao:long
#Data:Date
#Ideia:String
#Id_Usuario:long
#Estado:int
+Ideia_Solucao
+Ideia_Solucao
+Set_Estado:void
_Id_Ideia:long
_Id_B_Solucao:long
_Data:Date
_Ideia:String
_Id_Usuario:long
_Estado:int
Serializable
pokayoke server.Acesso BD.Ac
#Id_Acao:long
#Id_Problema:long
#Id_Ideia:long
#Responsavel:long
#Nome_Responsavel:String
#Descricao:String
#Status_Acao:int
#Data:Date
#Data_Prevista:Date
#Data_Termino:Date
#Status:boolean
+Acao
+Acao
_Data_Termino:Date
_Id_Acao:long
_Id_Problema:long
_Id_Ideia:long
_Id_Responsavel:long
_Nome_Responsavel:String
_Descricao:String
_Status_Acao:int
_Data:Date
_Data_Prevista:Date
_Status:boolean
java.io.Serializabl
Cliente
#Id_Cliente:int
#Nome_Cliente:Strin
+Cliente
+Cliente
_Nome_Cliente:Strin
_Id_Cliente:int
java.io.Serializabl
Fornecedor
#Id_Fornecedor:int
#Nome_Fornecedor:Strin
+Fornecedor
+Fornecedor
_Nome_Fornecedor:Strin
_Id_Fornecedor:int
java.io.Serializabl
Brainstorm Ideia
#Id_Ideia:long
#Id_Brainstorm:lon
#Data:Date
#Ideia:String
#Id_Usuario:long
+Brainstorm_Ideia
+Brainstorm_Ideia
_Id_Ideia:long
_Id_Brainstorm:lon
_Data:Date
_Ideia:String
_Id_Usuario:long
java.io.Serializabl
DCE Ideia
Id_DCE_Ideia:long
Id_Problema:long
Id_Usuario:long
Data_Inserir:Date
Proposta:String
Possibilidade:int
M:int
+DCE_Ideia
+DCE_Ideia
+Get_Id_DCE_Ideia:lon
+Get_Id_Problema:long
+Get_Id_Usuario:long
+Get_Data_Inserir:Date
+Get_Proposta:String
+Get_Possibilidade:int
+Get_M:int
java.io.Serializabl
ID Causa Raiz
Id_Source_Cause:int
Descricao:String
+ID_Causa_Raiz
+ID_Causa_Raiz
_Id_Source_Cause:in
_Descricao:String
Inter_Norm
+Norm
util
+file
+ExampleFileFilte
Acesso_BD
+pAcaoImediata
+pStep I
+pDepartamento
+pAcao
+pTimeMultifuncion
+pStep V
+pIdeia Solucao
+pNorm
+pCliente
+pBrain Solucoes
+pDCE
+pStep III
+pFornecedor
+pID Causa Raiz
+pObject
+Conexao BD
+Conexao Norma BI
+pImplementacao
+pBrain Imediata
+pStep IV
+pProblema
+pJessNorms
+pStep II
+pDCE Ideia
+pUsuario
+pBrainstorm Ideia
+Acao
RMI_Interface
+iTimeMultifuncion
+iJessNorms
+iBrain Solucoes
+iBrainstorm Ideia
+iCliente
+iDepartamento
+iNorm
+iStep V
+iStep III
+iProblema
+iIdeia Solucao
+iDCE
+iStep II
+iStep I
+iID Causa Raiz
+iStep IV
+iBrain Imediata
+iAcaoImediata
+iImplementacao
+iDCE Ideia
+iFornecedor
+iUsuario
+iAcao

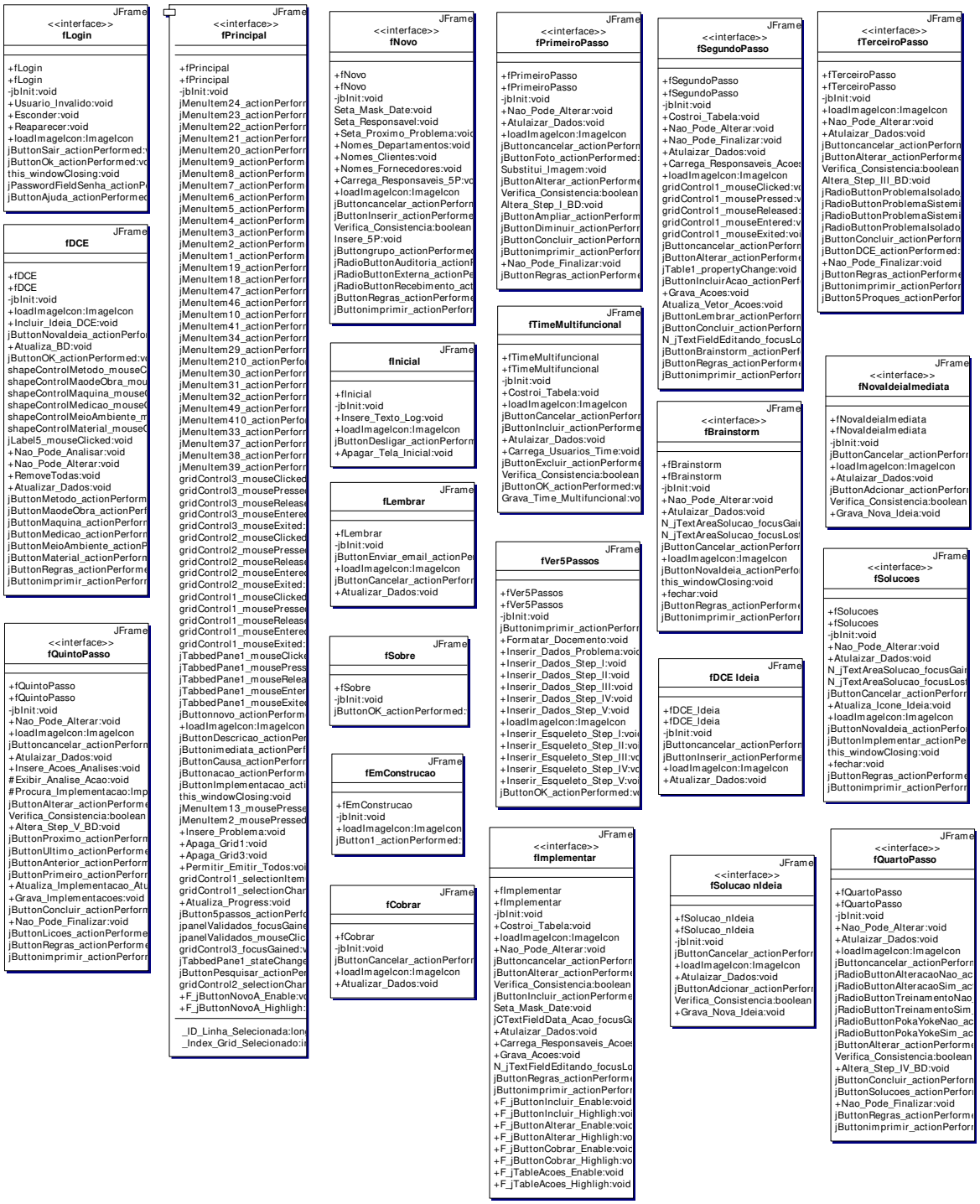

file
+file
FileFilter
ExampleFileFilter
-TYPE_UNKNOWN:String
-HIDDEN_FILE:String
-filters:Hashtable
-fullDescription:String
-useExtensionsInDescription
+ExampleFileFilter
+ExampleFileFilter
+ExampleFileFilter
+ExampleFileFilter
+ExampleFileFilter
+accept:boolean
+getExtension:String
+addExtension:void
description:String
extensionListInDescription:

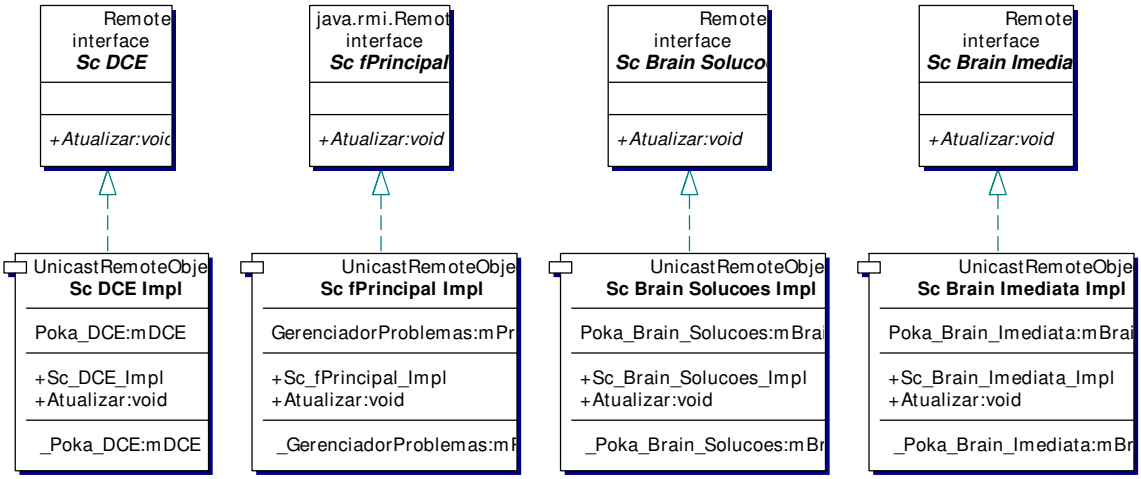

java.io.Serializable
Norm
#Id_Norma:long
#Affodance:String
#Condition:String
#Modal:int
#Agent:String
#Action:String
#Status:boolean
+Norm
+Norm
+Norm
_Status:boolean
_Condition:String
_Action:String
_Modal:int

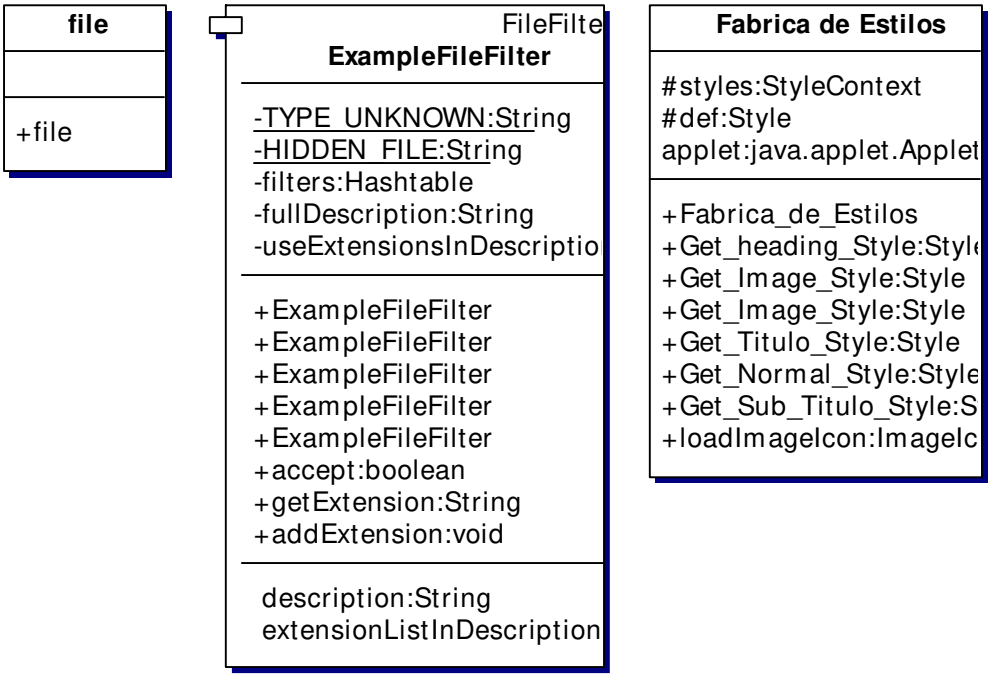

java.rmi.Remot
interface
iTimeMultifuncional
+Load_Time_Multifuncional:Tin
+Gravar_TimeMultifuncional:voi
+Altera_TimeMultifuncional:voi
java.rmi.Remot
interface
iBrain Solucoes
+Load:Brain_Solucoes
+Gravar_Brain_Solucao:voi
java.rmi.Remot
interface
iStep V
+Load:Step_V
+Gravar_Step_V:voi
+Altera_Step_V:voic
java.rmi.Remot
interface
iStep I
+Load:Step_I
+Ler_Imagem_Problema:Image
+Grava_Step_I:void
+Altera_Step_I:void
java.rmi.Remot
interface
iAcao
+Load_All_Problema:Vecto
+Load_All_Solucao:Vector
+Grava_Acao:void
+Altera_Acao:void
+Acoes_Estado:int
java.rmi.Remot
interface
iJessNorms
+Load_All_Normas:Vecto
+Load_All_Roles:Vector
java.rmi.Remot
interface
iBrainstorm Ideia
+Load_All_Ideias:Vector
+Gravar_Brainstorm_Ideia:vo
+Conecta_Sincro:void
java.rmi.Remot
interface
iStep III
+Load:Step_III
+Gravar_Step_III:voi
+Altera_Step_III:voic
java.rmi.Remot
interface
iID Causa Raiz
+Load_All:Vecto
java.rmi.Remot
interface
iFornecedor
+Load_All:Vecto
java.rmi.Remot
interface
iImplementacao
+Load_All_Problema:Vecto
+Grava_Implementacao:voi
+Altera_Implementacao:vo
java.rmi.Remot
interface
iCliente
+Load_All:Vecto
java.rmi.Remot
interface
iDepartamento
+Load_All:Vecto
java.rmi.Remot
interface
iNorm
+Load_All:Vecto
java.rmi.Remot
interface
iDCE
+Load:DCE
+Gravar_DCE:voi
+Altera_DCE:void
java.rmi.Remot
interface
iStep IV
+Load:Step_IV
+Gravar_Step_IV:voi
+Altera_Step_IV:voic
java.rmi.Remot
interface
iUsuario
+Load:pokayoke_server.Usuari
+Load_All:Vector
java.rmi.Remot
interface
iAcaoImediata
+Load_All_Problema:Vecto
+Altera_Acao_Imediata:voi
+Grava_Acao_Imediata:voi
java.rmi.Remot
interface
iProblema
+Load:Vector
+Persiste_Problema:voi
+Encerra_Passo:void
+Conecta_Sincro:void
java.rmi.Remot
interface
iIdeia Solucao
+Load_All_Solucoes:Vecto
+Gravar_Ideia_Solucao:voi
+Altera_Ideia:void
+Conecta_Sincro:void
java.rmi.Remot
interface
iStep II
+Load:Step_II
+Grava_Step_II:voi
+Altera_Step_II:voi
java.rmi.Remot
interface
iBrain Imediata
+Load:Brain_Imediata
+Gravar_Brain_Imediata:vo
java.rmi.Remot
interface
iDCE Ideia
+Load_All_DCE_Ideias:Vect
+Gravar_Ideia_DCE:void
+Conecta_Sincro:void


Conexao Norma BD
-BDconexao:java.sql.Conn
+Conexao_Norma_BD
+Reconectar:void
_conexao:java.sql.Connec
Conexao BD
-BDconexao:java.sql.Conn
+Conexao_BD
+Reconectar:void
_conexao:java.sql.Connec
UnicastRemoteObj
pObject
Poka_Banco:Conexao_BD
Norma_Banco:Conexao_Norr
BDconexao:java.sql.Connect
Poka:pokayoke_server.Pokay
#vStm:java.sql.Statement
+pObject
+pObject
+pObject
iJessNorms
pJessNorms
+pJessNorms
+Load_All_Normas:Ve
+Load_All_Roles:Vecto
iUsuario
pUsuario
+pUsuario
+Load:Usuario
+Load_All:Vect
iIdeia_Solucao
pIdeia Solucao
Clientes_Atua_fBrain_Soluco
+pIdeia_Solucao
+Load_All_Solucoes:Vector
+Gravar_Ideia_Solucao:void
+Altera_Ideia:void
+Conecta_Sincro:void
#Atualizar_Clientes:void
iAcao
pAcao
#vStm2:java.sql.Statem
+pAcao
+Load_All_Problema:Ve
+Load_All_Solucao:Vect
+Grava_Acao:void
+Altera_Acao:void
+Acoes_Estado:int
iDCE
pDCE
#rs:java.sql.Result
+pDCE
+Load:DCE
+Gravar_DCE:void
+Altera_DCE:void
iBrain_Solucoes
pBrain Solucoes
#rs:java.sql.ResultSet
+pBrain_Solucoes
+Load:Brain_Solucoes
+Gravar_Brain_Solucao:
iAcaoImediata
pAcaoImediata
#vStm2:java.sql.Statem
+pAcaoImediata
+Load_All_Problema:Ve
+Altera_Acao_Imediata:
+Grava_Acao_Imediata:
iDCE_Ideia
pDCE Ideia
Clientes_Atua_fDCE:Vecto
+pDCE_Ideia
+Load_All_DCE_Ideias:Ve
+Gravar_Ideia_DCE:void
+Conecta_Sincro:void
#Atualizar_Clientes:void
iProblema
pProblema
#rs:java.sql.ResultSet
#rs2:java.sql.ResultSet
#vStm2:java.sql.Statement
Nome_Resp_5_Passos:String
Nome_Responsavel:String
Nome_Departamento_Emit
Clientes_Atua_fPrincipal:Vec
+pProblema
+Load:Vector
+Persiste_Problema:void
+Encerra_Passo:void
+Conecta_Sincro:void
#Atualizar_Clientes:void
iStep_I
pStep I
applet:java.applet.Applet
+pStep_I
+Load:Step_I
+Ler_Imagem_Problema:Ima
+loadImageIcon:ImageIcon
+Grava_Step_I:void
+Altera_Step_I:void
Grava_Image_Icon:void
iStep_II
pStep II
+pStep_II
+pStep_II
+Load:Step_II
+Grava_Step_II:v
+Altera_Step_II:v
iStep_III
pStep III
#rs2:java.sql.ResultSe
#vStm2:java.sql.State
+pStep_III
+Load:Step_III
+Gravar_Step_III:void
+Altera_Step_III:void
iStep_IV
pStep IV
#rs:java.sql.Result
+pStep_IV
+Load:Step_IV
+Gravar_Step_IV:v
+Altera_Step_IV:vo
iStep_V
pStep V
+pStep_V
+Load:Step_V
+Gravar_Step_V:v
+Altera_Step_V:vo
iImplementacao
pImplementacao
+pImplementacao
+Load_All_Problema:Vec
+Grava_Implementacao:
+Altera_Implementacao
iTimeMultifuncion
pTimeMultifuncional
#vStm2:java.sql.Statement
+pTimeMultifuncional
+Load_Time_Multifuncional:
+Gravar_TimeMultifuncional
+Altera_TimeMultifuncional:
iDepartament
pDepartamento
+pDepartamen
+Load_All:Vect
iCliente
pCliente
+pCliente
+Load_All:Vec
iFornecedor
pFornecedor
+pFornecedor
+Load_All:Vect
iBrain_Imediat
pBrain Imediata
#rs:java.sql.ResultSet
+pBrain_Imediata
+Load:Brain_Imediata
+Gravar_Brain_Imediata
iBrainstorm_Idei
pBrainstorm Ideia
Clientes_Atua_fBrain_Imedia
+pBrainstorm_Ideia
+Load_All_Ideias:Vector
+Gravar_Brainstorm_Ideia:v
+Conecta_Sincro:void
#Atualizar_Clientes:void
iID_Causa_Raiz
pID Causa Raiz
+pID_Causa_Ra
+Load_All:Vecto


mLogin
#Poka:Pokayoke
#TelaLogin:fLogin
#User:Usuario
+mLogin
+Verifica_Usuario:void
+Logout:void
mProblem
#Poka:Pokayoke
#PokaLogin:mLogin
#Norm_Participar:mParticipar
#Problemas:Vector
#Tela_Principal:fPrincipal
#iPokaProblema:iProblema
#Tela_N_Problema:fNovo
#PokaPasso_I:mStep_I
#PokaPasso_II:mStep_II
#PokaPasso_III:mStep_III
#PokaPasso_IV:mStep_IV
#PokaPasso_V:mStep_V
Interface_Dinamica:D_Interface
PokaUser:Usuario
+mProblem
#Recupera_Problemas:void
+Logout:void
+Abrir_Problema:void
+Define_Time:void
+Load_Time:void
+Proximo_Problema:long
+Insere_Problema:void
+Atualiza_Problemas:void
+Pega_Problema_Selecionado:
+Passo_I:void
+Passo_II:void
+Passo_III:void
+Passo_IV:void
+Passo_V:void
+Encerra_Passo:void
+Ver_5_Passos:void
_Pokayoke:Pokayoke
_PokaPasso_II:mStep_II
_PokaPasso_IV:mStep_IV
mSteps
#GProblema:mProblem
#Poka:Pokayoke
#Problema_Selecionado:Problem
#Pode_Preencher:String
+mSteps
Pokayoke
#Poka_Inicial:fInicial
#PokaLogin:mLogin
#PokaUser:Usuario
#PokaProblem:mProblem
#SrvUsuario:iUsuario
#SrvProblema:iProblema
#SrvCliente:iCliente
#SrvFornecedor:iFornecedor
#SrvNorm:iNorm
#SrvAcao:iAcao
#SrvStep_I:iStep_I
#SrvStep_II:iStep_II
#SrvStep_III:iStep_III
#SrvStep_IV:iStep_IV
#SrvStep_V:iStep_V
#SrvDCE:iDCE
#SrvDCE_Ideia:iDCE_Ideia
#SrvJessNorms:iJessNorms
#Poka_D_Interface:D_Interface
+Pokayoke
+main:void
+Autorizado:void
_mLogin:mLogin
_Usuario:Usuario
_SrvUsuario:iUsuario
_SrvProblema:iProblema
_SrvCliente:iCliente
_SrvFornecedor:iFornecedor
_SrvStep_I:iStep_I
_SrvStep_II:iStep_II
_SrvStep_III:iStep_III
_SrvStep_IV:iStep_IV
_SrvStep_V:iStep_V
_SrvDCE:iDCE
_SrvDCE_Ideia:iDCE_Ideia
_SrvJessNorms:iJessNorms
_Poka_D_Interface:D_Interface
mStep I
#fPokaPrimeiroPasso:fPrimeiro
#PokaInterStep_I:nStep_I
#Passo_I_Selecionado:Step_I
#SrvPasso_I_Persist:iStep_I
+mStep_I
+Insere_Step_I:void
+Altera_Step_I:void
_PokaProblem:mProblem
mStep II
#fPokaSegundoPasso:fSegundo
#PokaInterStep_II:nStep_II
#Passo_II_Selecionado:Step_II
#SrvPasso_II_Persist:iStep_II
+mStep_II
+Insere_Step_II:void
+Altera_Step_II:void
+Abre_Brainstorm:void
+Abre_Lembrar:void
#Get_Passo_II_Selecionado:St
_GerenciadorProblemas:mPro
mStep III
#fPokaTerceiroPasso:fTerceiro
#PokaInterStep_III:nStep_III
#Passo_III_Selecionado:Step_I
#SrvPasso_III_Persist:iStep_III
+mStep_III
+Insere_Step_III:void
+Altera_Step_III:void
+Abre_DCE:void
_GerenciadorProblemas:mPro
mStep IV
#fPokaQuartoPasso:fQuartoPas
#PokaInterStep_IV:nStep_IV
#Passo_IV_Selecionado:Step_I
#SrvPasso_IV_Persist:iStep_IV
+mStep_IV
+Insere_Step_IV:void
+Altera_Step_IV:void
+Abre_Brainstorm:void
_GerenciadorProblemas:mPro
_Pode_Preencher:String
mStep V
#fPokaQuintoPasso:fQuintoPas
#PokaInterStep_V:nStep_V
#Passo_V_Selecionado:Step_V
#SrvPasso_V_Persist:iStep_V
#SrvImplementacao_Persist:iIr
#Acoes:Vector
#Implementacoes:Vector
#Acao_Srv:iAcao
+mStep_V
+Insere_Step_V:void
+Altera_Step_V:void
+Insere_Implementacao:void
+Altera_Implementacao:void
_GerenciadorProblemas:mPro
Acesso_BD
util
+file
+ExampleFileFilter
+Fabrica de Estilos
mLembrar
#Poka:Pokayoke
#GProblema:mProblem
#GStep_II:mStep_II
#Poka_Step_II:Step_II
#Poka_Acoes_Ime:Vector
#Poka_Problema:Problema_5P
#Poka_Acao:Acao
#Poka_Usuario:Usuario
#Poka_Lembrar:fLembrar
+mLembrar
mBrain Imediata
#GProblema:mProblem
#fPokaBrainstorm:fBrainstorm
#Pode_Preencher:String
#Poka:Pokayoke
#Poka_Brain:Brain_Imediata
#Srv_Poka_Brain_Imediata_Pe
#PokaInterBrain:nBrain_Imedi
#SrvPokaBrainstorm_Ideia:iBr
#Brain_Ativo:boolean
+mBrain_Imediata
+Insere_Brain:void
+Nova_Ideia:void
+Grava_Ideia:void
+Atualizar_Tela:void
_GerenciadorProblemas:mPro
_Brain_Ativo:boolean
mDCE
#GProblema:mProblem
#fPokaDCE:fDCE
#Pode_Preencher:String
#Pode_Analisar:String
#Poka:Pokayoke
#Srv_Poka_DCE_Persist:iDCE
#PokaInterDCE:nDCE
#DCE_Ativo:boolean
+mDCE
+Insere_DCE:void
+Insere_DCE:void
+Altera_DCE:void
+Nova_Ideia:void
+Grava_DCE_Ideia:void
+Atualizar_Tela:void
_GerenciadorProblemas:mPro
_DCE_Ativo:boolean
mBrain Solucoes
#GProblema:mProblem
#fPokaSolucoes:fSolucoes
#Pode_Preencher:String
#Poka:Pokayoke
#Poka_Brain:Brain_Solucoes
#Srv_Poka_Brain_Persist:iBrai
#PokaInterBrain:nBrain_Soluc
#SrvPokaIdeiaSolucao:iIdeia_S
#SrvPoka_Acao:iAcao
#Pokaldeia:fSolucao_nIdeia
#Ideia_Implementando:int
#Brain_Ativo:boolean
#Ser_Atua_fBrain_Solucoes:Sc
Interface_Dinamica:D_Interface
+mBrain_Solucoes
+Insere_Brain:void
+Nova_Ideia:void
+Grava_Ideia:void
+Atualizar_Tela:void
+Implementa_Ideia:void
+Verifica_Status:int
+Altera_Acao:void
+Grava_Acao:void
+close_Implementar:void
_GerenciadorProblemas:mPro
_Brain_Ativo:boolean
RMI_Client
+Sc DCE
+Sc fPrincipal Impl
+Sc Brain Solucoes
+Sc Brain Solucoes Impl
+Sc Brain Imediata Impl
+Sc DCE Impl
+Sc fPrincipal
+Sc Brain Imediata
Inter_Norm
+nStep III
+nBrain Solucoes
+nEmitir
+nStep II
+nStep I
+mParticipar
+nStep IV
+nStep V
+Inter Norm Affordance
+nDCE
+nAnalyse DCE
+nBrain Imediata
+D Interface
Interface
+fLogin
+fPrimeiroPasso
+fSegundoPasso
+fQuartoPasso
+fImplementar
+fQuintoPasso
+fSolucoes
+fSolucao nIdeia
+fTerceiroPasso
+fPrincipal
+fNovo
+fDCE
+fInicial
+fBrainstorm
+fLembrar
+fNovaldeialmediata
+fTimeMultifunciona
+fCobrar
+fVer5Passos
+fSobre
+fEmConstrucao
+fDCE Ideia


JFrame
<<interface>>
fLogin
+fLogin
+fLogin
-jbInit:void
+Usuario_Invalido:void
+Esconder:void
+Reaparecer:void
+loadImageIcon:ImageIcon
jButtonSair_actionPerformed:
jButtonOk_actionPerformed:vo
this_windowClosing:void
jPasswordFieldSenha_actionP
jButtonAjuda_actionPerformed
JFrame
fDCE
+fDCE
+fDCE
-jbInit:void
+loadImageIcon:ImageIcon
+Incluir_Ideia_DCE:void
jButtonNovaIdeia_actionPerfo
+Atualiza_BD:void
jButtonOK_actionPerformed:v
shapeControlMetodo_mouseC
shapeControlMaodeObra_mou
shapeControlMaquina_mouse
shapeControlMedicao_mouse
shapeControlMeioAmbiente_m
shapeControlMaterial_mouse
jLabel5_mouseClicked:void
+Nao_Pode_Analisar:void
+Nao_Pode_Alterar:void
+RemoveTodas:void
+Atualizar_Dados:void
jButtonMetodo_actionPerform
jButtonMaodeObra_actionPerf
jButtonMaquina_actionPerforn
jButtonMedicao_actionPerforn
jButtonMeioAmbiente_actionP
jButtonMaterial_actionPerforn
jButtonRegras_actionPerforme
jButtonimprimir_actionPerforr
JFrame
<<interface>>
fQuintoPasso
+fQuintoPasso
+fQuintoPasso
-jbInit:void
+Nao_Pode_Alterar:void
+loadImageIcon:ImageIcon
jButtoncancelar_actionPerforn
+Atulaizar_Dados:void
+Insere_Acoes_Analises:void
#Exibir_Analise_Acao:void
#Procura_Implementacao:Imp
jButtonAlterar_actionPerforme
Verifica_Consistencia:boolean
+Altera_Step_V_BD:void
jButtonProximo_actionPerforn
jButtonUltimo_actionPerforme
jButtonAnterior_actionPerforn
jButtonPrimeiro_actionPerforr
+Atualiza_Implementacao_Atu
+Grava_Implementacoes:void
jButtonConcluir_actionPerforn
+Nao_Pode_Finalizar:void
jButtonLicoes_actionPerforme
jButtonRegras_actionPerforme
jButtonimprimir_actionPerforr
JFrame
<<interface>>
fPrincipal
+fPrincipal
+fPrincipal
-jbInit:void
jMenuItem24_actionPerforr
jMenuItem23_actionPerforr
jMenuItem22_actionPerforr
jMenuItem21_actionPerforr
jMenuItem20_actionPerforr
jMenuItem9_actionPerform
jMenuItem8_actionPerform
jMenuItem7_actionPerform
jMenuItem6_actionPerform
jMenuItem5_actionPerform
jMenuItem4_actionPerform
jMenuItem3_actionPerform
jMenuItem2_actionPerform
jMenuItem1_actionPerform
jMenuItem19_actionPerforr
jMenuItem18_actionPerforr
jMenuItem47_actionPerforr
jMenuItem46_actionPerforr
jMenuItem10_actionPerforr
jMenuItem41_actionPerforr
jMenuItem34_actionPerforr
jMenuItem29_actionPerforr
jMenuItem210_actionPerfo
jMenuItem30_actionPerforr
jMenuItem31_actionPerforr
jMenuItem32_actionPerforr
jMenuItem49_actionPerforr
jMenuItem410_actionPerfo
jMenuItem33_actionPerforr
jMenuItem37_actionPerforr
jMenuItem38_actionPerforr
jMenuItem39_actionPerforr
gridControl3_mouseClicked
gridControl3_mousePresse
gridControl3_mouseReleas
gridControl3_mouseEntere
gridControl3_mouseExited:
gridControl2_mouseClicked
gridControl2_mousePresse
gridControl2_mouseReleas
gridControl2_mouseEntere
gridControl2_mouseExited:
gridControl1_mouseClicked
gridControl1_mousePresse
gridControl1_mouseReleas
gridControl1_mouseEntere
gridControl1_mouseExited:
jTabbedPane1_mouseClicke
jTabbedPane1_mousePress
jTabbedPane1_mouseRelea
jTabbedPane1_mouseEnter
jTabbedPane1_mouseExite
jButtonnovo_actionPerform
+loadImageIcon:ImageIcon
jButtonDescricao_actionPer
jButtonimediata_actionPerf
jButtonCausa_actionPerforr
jButtonacao_actionPerform
jButtonImplementacao_acti
this_windowClosing:void
jMenuItem13_mousePresse
jMenuItem2_mousePressed
+Insere_Problema:void
+Apaga_Grid1:void
+Apaga_Grid3:void
+Permitir_Emitir_Todos:voi
gridControl1_selectionItem
gridControl1_selectionChar
+Atualiza_Progress:void
jButton5passos_actionPerfo
jpanelValidados_focusGaine
jpanelValidados_mouseClic
gridControl3_focusGained:v
jTabbedPane1_stateChange
jButtonPesquisar_actionPer
gridControl2_selectionChar
+F_jButtonNovoA_Enable:vo
+F_jButtonNovoA_Highligh:
_ID_Linha_Selecionada:lon
_Index_Grid_Selecionado:i
JFrame
<<interface>>
fNovo
+fNovo
+fNovo
-jbInit:void
Seta_Mask_Date:void
Seta_Responsavel:void
+Seta_Proximo_Problema:voi
+Nomes_Departamentos:void
+Nomes_Clientes:void
+Nomes_Fornecedores:void
+Carrega_Responsaveis_5P:vo
+loadImageIcon:ImageIcon
jButtoncancelar_actionPerforn
jButtonInserir_actionPerforme
Verifica_Consistencia:boolean
Insere_5P:void
jButtongrupo_actionPerformed
jRadioButtonAuditoria_actionP
jRadioButtonExterna_actionPe
jRadioButtonRecebimento_act
jButtonRegras_actionPerforme
jButtonimprimir_actionPerforr
JFrame
fInicial
+fInicial
-jbInit:void
+Insere_Texto_Log:void
+loadImageIcon:ImageIcon
jButtonDesligar_actionPerforn
+Apagar_Tela_Inicial:void
JFrame
fLembrar
+fLembrar
-jbInit:void
jButtonEnviar_email_actionPe
+loadImageIcon:ImageIcon
jButtonCancelar_actionPerforn
+Atualizar_Dados:void
JFrame
fSobre
+fSobre
-jbInit:void
jButtonOK_actionPerformed:
JFrame
fEmConstrucao
+fEmConstrucao
-jbInit:void
+loadImageIcon:ImageIcon
jButton1_actionPerformed:
JFrame
fCobrar
+fCobrar
-jbInit:void
jButtonCancelar_actionPerforn
+loadImageIcon:ImageIcon
+Atualizar_Dados:void
JFrame
<<interface>>
fPrimeiroPasso
+fPrimeiroPasso
+fPrimeiroPasso
-jbInit:void
+Nao_Pode_Alterar:void
+Atulaizar_Dados:void
+loadImageIcon:ImageIcon
jButtoncancelar_actionPerforn
jButtonFoto_actionPerformed:
Substitui_Imagem:void
jButtonAlterar_actionPerforme
Verifica_Consistencia:boolean
Altera_Step_I_BD:void
jButtonAmpliar_actionPerform
jButtonDiminuir_actionPerforr
jButtonConcluir_actionPerforn
jButtonimprimir_actionPerforr
+Nao_Pode_Finalizar:void
jButtonRegras_actionPerforme
JFrame
fTimeMultifuncional
+fTimeMultifuncional
+fTimeMultifuncional
-jbInit:void
+Costroi_Tabela:void
+loadImageIcon:ImageIcon
jButtonCancelar_actionPerforr
jButtonIncluir_actionPerforme
+Atulaizar_Dados:void
+Carrega_Usuarios_Time:void
jButtonExcluir_actionPerforme
Verifica_Consistencia:boolean
jButtonOK_actionPerformed:vo
Grava_Time_Multifuncional:vo
JFrame
fVer5Passos
+fVer5Passos
+fVer5Passos
-jbInit:void
jButtonimprimir_actionPerforr
+Formatar_Docemento:void
+Inserir_Dados_Problema:voic
+Inserir_Dados_Step_I:void
+Inserir_Dados_Step_II:void
+Inserir_Dados_Step_III:void
+Inserir_Dados_Step_IV:void
+Inserir_Dados_Step_V:void
+loadImageIcon:ImageIcon
+Inserir_Esqueleto_Step_I:voi
+Inserir_Esqueleto_Step_II:vo
+Inserir_Esqueleto_Step_III:vo
+Inserir_Esqueleto_Step_IV:vo
+Inserir_Esqueleto_Step_V:voi
jButtonOK_actionPerformed:vo
JFrame
<<interface>>
fImplementar
+fImplementar
+fImplementar
-jbInit:void
+Costroi_Tabela:void
+loadImageIcon:ImageIcon
+Nao_Pode_Alterar:void
jButtoncancelar_actionPerforn
jButtonAlterar_actionPerforme
Verifica_Consistencia:boolean
jButtonIncluir_actionPerforme
Seta_Mask_Date:void
jCTextFieldData_Acao_focusGa
+Atulaizar_Dados:void
+Carrega_Responsaveis_Acoes
+Grava_Acoes:void
N_jTextFieldEditando_focusLo
jButtonRegras_actionPerforme
jButtonimprimir_actionPerforr
+F_jButtonIncluir_Enable:void
+F_jButtonIncluir_Highligh:voi
+F_jButtonAlterar_Enable:voic
+F_jButtonAlterar_Highligh:vo
+F_jButtonCobrar_Enable:voic
+F_jButtonCobrar_Highligh:vo
+F_jTableAcoes_Enable:void
+F_jTableAcoes_Highligh:void
JFrame
<<interface>>
fSegundoPasso
+fSegundoPasso
+fSegundoPasso
-jbInit:void
+Costroi_Tabela:void
+Nao_Pode_Alterar:void
+Nao_Pode_Finalizar:void
+Atulaizar_Dados:void
+Carrega_Responsaveis_Acoes
+loadImageIcon:ImageIcon
gridControl1_mouseClicked:vo
gridControl1_mousePressed:v
gridControl1_mouseReleased:
gridControl1_mouseEntered:v
gridControl1_mouseExited:voi
jButtoncancelar_actionPerforn
jButtonAlterar_actionPerforme
jTable1_propertyChange:void
jButtonIncluirAcao_actionPerf
+Grava_Acoes:void
Atualiza_Vetor_Acoes:void
jButtonLembrar_actionPerforn
jButtonConcluir_actionPerforn
N_jTextFieldEditando_focusLo
jButtonBrainstorm_actionPerf
jButtonRegras_actionPerforme
jButtonimprimir_actionPerforr
JFrame
<<interface>>
fBrainstorm
+fBrainstorm
+fBrainstorm
-jbInit:void
+Nao_Pode_Alterar:void
+Atulaizar_Dados:void
N_jTextAreaSolucao_focusGai
N_jTextAreaSolucao_focusLos
jButtonCancelar_actionPerforr
+loadImageIcon:ImageIcon
jButtonNovaIdeia_actionPerfo
this_windowClosing:void
+fechar:void
jButtonRegras_actionPerforme
jButtonimprimir_actionPerforr
JFrame
fDCE Ideia
+fDCE_Ideia
+fDCE_Ideia
-jbInit:void
jButtoncancelar_actionPerforn
jButtonInserir_actionPerforme
+loadImageIcon:ImageIcon
+Atualizar_Dados:void
JFrame
<<interface>>
fSolucao nIdeia
+fSolucao_nIdeia
+fSolucao_nIdeia
-jbInit:void
jButtonCancelar_actionPerforr
+loadImageIcon:ImageIcon
+Atulaizar_Dados:void
jButtonAdcionar_actionPerforr
Verifica_Consistencia:boolean
+Grava_Nova_Ideia:void
JFrame
<<interface>>
fTerceiroPasso
+fTerceiroPasso
+fTerceiroPasso
-jbInit:void
+loadImageIcon:ImageIcon
+Nao_Pode_Alterar:void
+Atulaizar_Dados:void
jButtoncancelar_actionPerforn
jButtonAlterar_actionPerforme
Verifica_Consistencia:boolean
Altera_Step_III_BD:void
jRadioButtonProblemaIsolado
jRadioButtonProblemaSistemi
jRadioButtonProblemaSistemi
jRadioButtonProblemaIsolado
jButtonConcluir_actionPerforn
jButtonDCE_actionPerformed:
+Nao_Pode_Finalizar:void
jButtonRegras_actionPerforme
jButtonimprimir_actionPerforr
jButton5Proques_actionPerfor
JFrame
<<interface>>
fNovaldeiaImediata
+fNovaldeiaImediata
+fNovaldeiaImediata
-jbInit:void
jButtonCancelar_actionPerforr
+loadImageIcon:ImageIcon
+Atulaizar_Dados:void
jButtonAdcionar_actionPerfor
Verifica_Consistencia:boolean
+Grava_Nova_Ideia:void
JFrame
<<interface>>
fSolucoes
+fSolucoes
+fSolucoes
-jbInit:void
+Nao_Pode_Alterar:void
+Atulaizar_Dados:void
N_jTextAreaSolucao_focusGai
N_jTextAreaSolucao_focusLos
jButtonCancelar_actionPerforr
+Atualiza_Icone_Ideia:void
+loadImageIcon:ImageIcon
jButtonNovaIdeia_actionPerfo
jButtonImplementar_actionPe
this_windowClosing:void
+fechar:void
jButtonRegras_actionPerforme
jButtonimprimir_actionPerforr
JFrame
<<interface>>
fQuartoPasso
+fQuartoPasso
+fQuartoPasso
-jbInit:void
+Nao_Pode_Alterar:void
+Atulaizar_Dados:void
+loadImageIcon:ImageIcon
jButtoncancelar_actionPerforn
jRadioButtonAlteracaoNao_ac
jRadioButtonAlteracaoSim_ac
jRadioButtonTreinamentoNao
jRadioButtonTreinamentoSim
jRadioButtonPokaYokeNao_ac
jRadioButtonPokaYokeSim_ac
jButtonAlterar_actionPerforme
Verifica_Consistencia:boolean
+Altera_Step_IV_BD:void
jButtonConcluir_actionPerforn
jButtonSolucoes_actionPerfor
+Nao_Pode_Finalizar:void
jButtonRegras_actionPerforme
jButtonimprimir_actionPerforr
D Interface
KB_Normas:Rete
GPokayoke:Pokayoke
SrvJessNorms:pokayoke_server.RMI_Interface.iJessNorms
Poka_fPrincipal:pokayoke.Interface.fPrincipal
Poka_fImplementar:pokayoke.Interface.fImplementar
+D_Interface
+Carrega_Normas:void
+Percepcao:void
+Action:void
+Recarrega_Normas:void
+Executa_Acao:void
_Poka_fPrincipal:pokayoke.Interface.fPrincipal
_Poka_fImplementar:pokayoke.Interface.fImplementar


Remote
interface
Sc DCE
+Atualizar:voic
java.rmi.Remot
interface
Sc fPrincipal
+Atualizar:void
Remote
interface
Sc Brain Soluco
+Atualizar:void
Remote
interface
Sc Brain Imedia
+Atualizar:void
UnicastRemoteObje
Sc DCE Impl
Poka_DCE:mDCE
+Sc_DCE_Impl
+Atualizar:void
_Poka_DCE:mDCE
UnicastRemoteObje
Sc fPrincipal Impl
GerenciadorProblemas:mPr
+Sc_fPrincipal_Impl
+Atualizar:void
_GerenciadorProblemas:mP
UnicastRemoteObje
Sc Brain Solucoes Impl
Poka_Brain_Solucoes:mBra
+Sc_Brain_Solucoes_Impl
+Atualizar:void
_Poka_Brain_Solucoes:mBr
UnicastRemoteObje
Sc Brain Imediata Impl
Poka_Brain_Imediata:mBrai
+Sc_Brain_Imediata_Impl
+Atualizar:void
_Poka_Brain_Imediata:mBr


file
+file
FileFilte
ExampleFileFilter
-TYPE_UNKNOWN:String
-HIDDEN_FILE:String
-filters:Hashtable
-fullDescription:String
-useExtensionsInDescriptio
+ExampleFileFilter
+ExampleFileFilter
+ExampleFileFilter
+ExampleFileFilter
+ExampleFileFilter
+accept:boolean
+getExtension:String
+addExtension:void
description:String
extensionListInDescription
Fabrica de Estilos
#styles:StyleContext
#def:Style
applet:java.applet.Applet
+Fabrica_de_Estilos
+Get_heading_Style:Style
+Get_Image_Style:Style
+Get_Image_Style:Style
+Get_Titulo_Style:Style
+Get_Normal_Style:Style
+Get_Sub_Titulo_Style:S
+loadImageIcon:ImageIc

Anexo C: Documentação do NBIC

Norm-Based Interface Configurator Documentation

Rodrigo Bonacin
Institute of Computing, State University of Campinas
Caixa Postal 617613083 970 Campinas, SP - Brazil
+55 19 32958178
Email: ra000470@ic.unicamp.br

M Cecilia C. Baranauskas
Institute of Computing, State University of Campinas
Caixa Postal 617613083 970 Campinas, SP - Brazil
+55 19 37885870
Email: cecilia@ic.unicamp.br

Kecheng Liu
Department of Computer Science
University of Reading, Reading, RG6 6AY, UK
Email: k.liu@reading.ac.uk

1. Introduction

The NBIC (Norm-Based Interface Configurator) is a software system constructed with the aim of making the interface maintenance easier and cheaper. Using the NBIC an Interface Engineer can do most of the changes in interface by changing the norms descriptions (Liu, 2000); no code modification is necessary.

The NBIC is independent of the end-user application; any Java application constructed according to a basic structure (see system requirements 3.1) can use it as a tool to help the software maintenance. We give the name "Norm Driven Environment for Interface Configuration" to refer to the architecture that specify this basic structure and the internal parts of the NBIC.

The definitions of the main terms used in this document are:

- *End-User Interface:* it is part of the final application, it is not part of the NBIC. It is the interface that is maintained by the NBIC.
- *End-User:* people that interact with the final application, they do not have direct access to the NBIC.
- *Interface Engineer:* the person responsible for the interface maintenance. The interface engineer interacts with the NBIC interface;
- *Developer:* people in charge of constructing a new End-User application;

This document has been constructed in order to give to the developers and maintainers (Interface Engineers) the essential information of how to model, construct and maintain interfaces using the NBIC. This document has two parts: in the first part we present a description of the architecture developed to deal with the complexity of allowing changes in the computational systems as the organisational norms change; in the second part we present how to use the NBIC (Norm-Base Interface Configurator) to construct and maintain systems based on the architecture described in the first part of the document.

We recommend the reading of whole document to people interested in constructing new software applications using the NBIC (Developers). If you are the person responsible for the interface maintenance (Interface Engineer) we recommend at least the reading of the sections 2, 3.1.3, 3.1.4 and 3.2.

2. PART-ONE: Norm Driven Environment for Interface Configuration

(IDEM A SEÇÃO 6.3)

3. PART-TWO: NBIC

3.1 Constructing a System

To construct a system using the NBIC it is necessary to follow the steps:

1. Semantic and Norm Analysis (It is not part of this document, see Liu (2000))
2. The identification of the dynamic and static interface elements of the end-user interface in accordance with the architecture. Identification of the norms that will be interpreted in the dynamic part (first part of the document)
3. Include in the ICE the norms by using the Norm Manager
4. Link the norms to the interface objects by using the Action Manager
5. Link the Inference Machine (JESS) and the ICE database to the System
6. Inform the Inference Machine and Receive the Inference Values
7. Link the Inferences with Interface Changes

3.1.1 NBIC System Requirements

Software Requirements:

- Windows 9x/Me/NT/2000/XP
- Microsoft Access 97 or later (+ODBC)
- Sun Java Virtual Machine 1.2 or later
- Jess version 6.0 or later

Recommended Hardware Requirements:

- Intel/AMD 1Ghz+
- 256 Mbytes+
- 10 Mbytes of free hard disk space
- 1024x768 resolution display

3.1.2 System Installation

The last version of NBIC do not include an install wizard. One have to follow a few steps to install the NBIC:

1. Install all the programs listed in the software requirements;

2. Copy the files “nbic.jar”, and NormInter.mdb to a specific directory
3. Configure the ODBC. To configure the ODBC it is necessary, Click in Start -> Settings -> Control Panel, after click in “administrative tools” and “Data Sources (ODBC)” icons. Click in the “User DSN” folder and after in the “add” button, choose “Microsoft access driver”. Include the “NormInter” in the “Data Source Name”, click in the “select” button and choose the “NormInter.mdb” database, and at lest click in the button OK (see figure 9).
4. Create a file named “Nbic.bat” in the same directory that you have copied the “nbic.jar”, with the following data:
   ...\javaw -XX:NewSize=64m -Xms128m -Xmx128m -classpath "nbic.jar"
5. Create a “shortcut” in the desktop, to execute, just click on the “shortcut”;

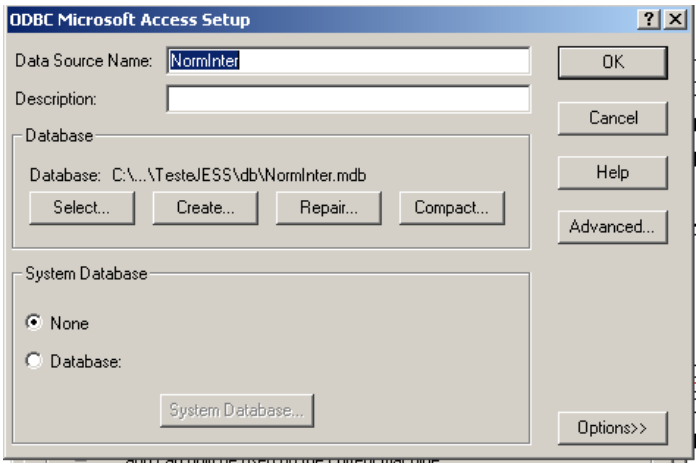


Figure 9: Configuring the ODBC for the ICE

3.1.3 Including Norms

The first step to specify norms using the Norm Manager is to include the affordances associated to norms (after the Semantic and Norm analysis). To include an affordance it is necessary to click in the “new affordance” button, and as figure 10 shows a new line will appear to include the affordances names.

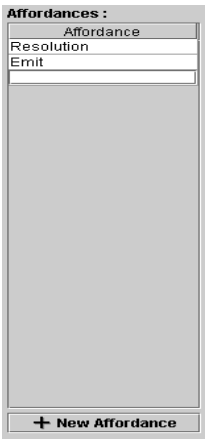


Figure 10: Affordance Description in the Norm Manager

After including the affordances we have to double click the affordance name (figure 10) to which the norms will be associated, afterwards a list with all the norms associated with the affordance will appear at the right side of the screen, as figure 11 shows. The next step is to click the button labelled "New Norm" (a new line will appear in the norm list).

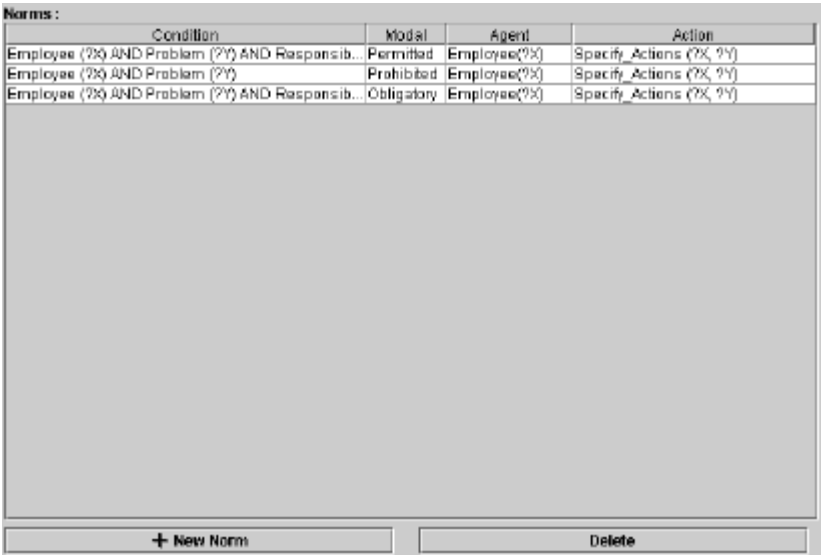


Figure 11: Norm Description at Norm Manager

Four fields must be filled out to specify a norm (see figure 11):

1. *Condition.* It is the condition for the norm to be applied. The syntax specification in Backus-Naur Form (BNF) is:
   <Condition> ::= <Predicate> {AND <Predicate>} {AND <Expression>}
   <Predicate> ::=<Affordance> (<Variable_Name>|<Value> {,< Variable _Name>|<Value>})
   <Expression> ::= < Variable_Name> <Operator> <Value>
   <Affordance> ::= <Alphabet_1>{<Alphabet_1>}
   < Variable_Name> ::= ?<Alphabet_1>{<Alphabet_1>}
   <Value> ::= <Alphabet_1>{<Alphabet_1>}
   <Operator> ::= =|>|<|<=|>=|<>
   <Alphabet> ::= a|...|z|A|...|Z|0|...|9|

   In the condition field a predicate contains affordances and variables or values that specify the context in which a norm is applied; for example,

"*Employee (?X) AND Problem (?Y) AND Responsible(?X,?Y)*" means that the norm will be applied if "*X*" is a *Employee*, "*Y*" is a Problem and "*X*" is *Responsible* for "*Y*". The condition field can also include a conditional expression to evaluate the state of the context. A more complex example of condition is: "*Card_Holder (?C) AND Outstanding_Credit(?C,?X) AND Past_Posting_Days(?X,?D) AND (?D>25) AND (?X>0)*" meaning that the norm will be applied if "*?C*" is a "*Card_Holder*", and after 25 days "*?D>25*" of posting the invoice "*Past_Posting_Days*", there is an amount "X>0" of "*Outstanding_Credit"*.
(Example of norms from Liu2000, p.105)

2. *Modal* or *Deontic*. It specifies the *Permittion*, *Obligation* or *Prohibition* for an agent (third field) to do some action (fourth field). The BNF of this field is:
<Deontic> ::= Permitted|Obligated|Prohibited

   There is a "*combo box*" to include one of these options.

3. *Agent.* It specifies who has to do the action (fourth field). The BNF of this field is:
<Agent> ::= <Affordance> (<Variable_Name>|<Value>)
<Affordance> ::= <Alphabet_l>{<Alphabet_l>}
< Variable_Name> ::= ?<Alphabet_l>{<Alphabet_l>}
<Value> ::= <Alphabet_l>{<Alphabet_l>}
<Alphabet> ::= a|...|z|A|...|Z||0|...|9|

   The affordance in this field can be any one identified as "agent" in the ontology chart. To link the "agent" field with any "agent" specified in the "condition field" you have to use the same variable. For example: if you include at the "*Employee (?X) AND Problem (?Y) AND Responsible(?X,?Y)*" in the first field and "*Employee (?X)*" in the third, it means that the agent that is responsible for the problem is permitted/obligated/prohibited do some action (fourth field). To refer to "any agent", for example any "*Employee"*, "*Manager", "Client", etc* it is necessary to use a variable which was not used in the first field. To refer to "a specific agent". It is necessary to include a value instead of a variable, for example: "*Client(Mary)*".

4. *Action*. It specifies the action that the agent (third field) is permitted/obligated/prohibited (second field) to do. The BNF of this field is:
<Action> ::= <Predicate> {AND <Predicate>}
<Predicate> ::=<Affordance> (<Variable_Name>|<Value> {,< Variable _Name>|<Value>})
<Affordance> ::= <Alphabet_l>{<Alphabet_l>}
< Variable_Name> ::= ?<Alphabet_l>{<Alphabet_l>}
<Value> ::= <Alphabet_l>{<Alphabet_l>}
<Alphabet> ::= a|...|z|A|...|Z||0|...|9|

   For example: if you include "*Employee (?X)*" at the third field, you can include "*Specify_Actions (?X, ?Y)*" that means that the *"Employee*" *?X* is *Permitted*, *Obligated* or *Prohibited* to "*Specify_Actions*" to the *Problem*" ?Y. This action has to be linked with the action roles (section 2.1.2)

Note also that all the quantificators are universal (*for all*). Figure 12 shows the BNF of a complete norm in the NBIC. These two examples illustrate complete norms in the NBIC:

1. Condition*: Employee (?X) AND Problem (?Y) AND Responsible(?X,?Y)*
   Modal: *Permitted*
   Agent: *Employee (?X)*
   Action: *Specify_Actions (?X,?Y)*

   This norm means that whenever an employee is the responsible for a problem (s)he has the permission to specify actions to correct the problem

2. Condition: *Card_Holder (?C) AND Outstanding_Credit(?C,?X) AND Past_Posting_Days(?X,?D) AND (?D>25) AND (?X>0)*
   Modal: *Obligated*
   Agent: *Card_Holder (?C)*
   Action: *Pay_Interest(?C,?X)*

   This norm means that after 25 days of posting the invoice, if there is still an amount of outstanding credit, the card holder will have to pay the interest.

```
Norm ::= <Affordance><Condition><Deontic><Agent><Action>
<Condition> ::= <Predicate> {AND <Predicate>} {AND <Expression>}
<Deontic> ::= Permitted|Obligated|Prohibited
<Agent> ::= <Affordance> (<Variable_Name>|<Value>)
<Action> ::= <Predicate> {AND <Predicate>}
<Predicate> ::= <Affordance> (<Variable_Name>|<Value> {,<Variable_Name>|<Value>})
<Expression> ::= <Variable_Name> <Operator> <Value>
<Affordance> ::= <alphabet_l>{<alphabet_l>}
<Variable_Name> ::= ?<alphabet_l>{<alphabet_l>}
<Value> ::= <alphabet_l>{<alphabet_l>}
<Operator> ::= =|>|<|<=|>=|<>
<alphabet> ::= a|...|z|A|...|Z||0|...|9|
```

Figure 12: The BNF of the norms

To remove a norm it is necessary to select the norm that we want to remove and click in the button with the label "delete" (right button part of figure 11). Be aware that the last version of the ICE has not the "undo" functionality.

After defining the norms, they must be translated to the Jess language in order to have some effect in the end-user interface. To translate the norms just click on the "Translate" button (right side of figure 13). To leave the action manager click on the "OK" button (centre of figure 13). To specify the action roles (next step) click on the button with the label "Action KB" (left side of figure 13) and the Action Manager will appear. The Action specification will be presented during the next section.

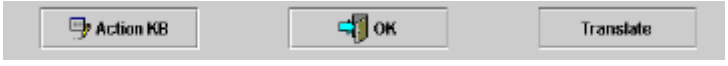


Figure 13: Actions, exit and translation

During the definition of the norms it is necessary to be aware of the limitations of the possible norms that can be written in the NBIC. The reason of the first limitation is that some statements of the norm description cannot be represented in first order logic, they require higher order logics (Filipe, 2000). However, there is no practical construction of higher logic automatic theorem provers. Among the challengers in the construction of theses systems are aspects such as the logical omniscience and undecidablility. The construction of modal logic provers (and deontic logic provers) is currently object of research in a very active area of research; alternatives such as: the use of extensions of the tableaux and resolution, and translations to decidable logics are explored in orther to produce more improved prove methods (Areces et al., 2001; Goré, 1992).

Although there is some improvement in the modal logic provers we have opted to implement the NBIC using a rule based system because of the large background in the development of efficient provers. The use of high performance algorithms as the "Rete algorithm" (used in the inference machine of the JESS) is essential factor to the successful use of the NBIC, since this algorithm is used during the construction of every dynamic element of the interface.

This choice does not mean that the architecture (ICE) is not suitable to the modal/deontic logic provers (it is possible to maintain the same architecture using another logical prover), but the NBIC implementation limits the inference to the rule based system scope. Although rules no longer seem to produce a complete representation of the world, the NBIC is able to deal with a very relevant group of the norm specification.

In this document we do not want discuss the representational and inference limitation of the rule based systems. There is an extensive literature about rule systems in books as Russel's one (Russell, 1995, 2003).

Two practical examples of limitations of the ICE are presented below:

1. *Logical Conflicts:* For example the: "It is obligatory to pay a tax to continue using the bank system. It is prohibited to overdraft." Suppose a situation that if the end-user pays the tax, (s)he will overdraft and (s)he needs the system to make a deposit or transfer money into the account to avoid the overdraft. What the system should do? Allow the end-user to access the system to do the deposit or transference before obligate him/her to pay the tax. This is could be a good alternative, but the ICE do not have any type of autonomy to deal with deontic conflicts. People who specify the norms must avoid this kind of conflict changing the norms. It is not a simple task, and the last version of the NBIC is not able to detect the deontic conflicts and help us to do this task;

2. *Inference limitation.* Consider the example: "The responsible for a problem have to include actions, you can be responsible or collaborator, if you define actions you have to respect the deadlines and if you are collaborator you have to respect the deadlines." If you translate these sentences straightforward to the logic we will have something (eliminating the deontic operations) similar to:
   $\forall x\ P(x) \Rightarrow Q(x)$
   $\forall x\ P(x)\ or\ R(x)$ ***equals*** $\forall x\ \neg P(x) \Rightarrow R(x)$

$\forall x\ Q(x) \Rightarrow S(x)$
$\forall x\ R(x) \Rightarrow S(x)$

Because of the "modus ponens" forward chaining inference limitation, if we inform the NBIC that "John have to respect the deadlines and that he is not a collaborator" the system will not infer that *John* is the responsible. A more detailed explanation about inference "completeness" can be found in Russell (1995, 2003);

3.1.4 Including the Action Rules

The action rules link the norms with the interface changes. The first step is to include the name of the interface objects that have a dynamic behaviour in the end-user interface. You can implement different mechanisms to allow the dynamic behaviour, for example a button could change the colour, size, font, etc or a form could include a new field. At this step it is necessary to include the interface objects that can be changed through the norm interpretation.

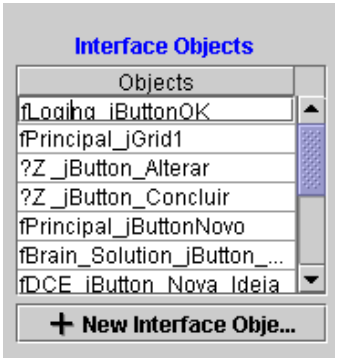


Figure 14: Interface Objects List

Figure 14 shows part of the Norm Manager interface used to include new objects. To do this it is necessary to click in the "New Interface Objects" button; a new line will appear in which we can include the object name. We are free to define the names of the objects, but is recommended to use a nomenclature related to the interface variables names used during the end-user interface construction. An example of the use of the nomenclature is: "fLogin_jButtonOK", that means the object "jButtonOK" of the form "fLogin", where "jButtonOK" is the name of the variable that implements the button in the end-user interface code and fLogin the variable that implements the form in the end-user interface code.

It is also possible to use the universal operator to refer to more than one object, for example: "?Z_jButton_OK", meaning the buttons "jButton_OK" of all forms, or "fLogin_?Z" that means all the interface objects of the form "fLogin". Usually one can configure many interface objects at the same time.

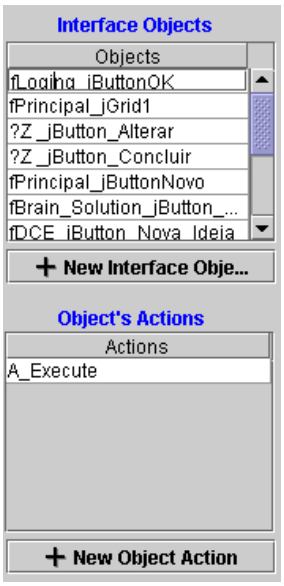


Figure 15: Interface Objects and Object's Actions List

The next step is to include the actions associated to each object, for example the object "fLogin_jButtonOK" could have the following actions "Enable", "Disable", "Highlight", etc. The actions reflect changes in the interface objects, these changes can be complex or very simple, according to what was implemented in the end-user interface code.

To include a new action it is necessary to select (double click) the interface object at the interface object list (top of figure 15). After clicking in the button with the label "new object action" (button of figure 15) include the action name. We are free to define actions names, but they must be recognised by the end-user interface, therefore a common protocol is recommended.

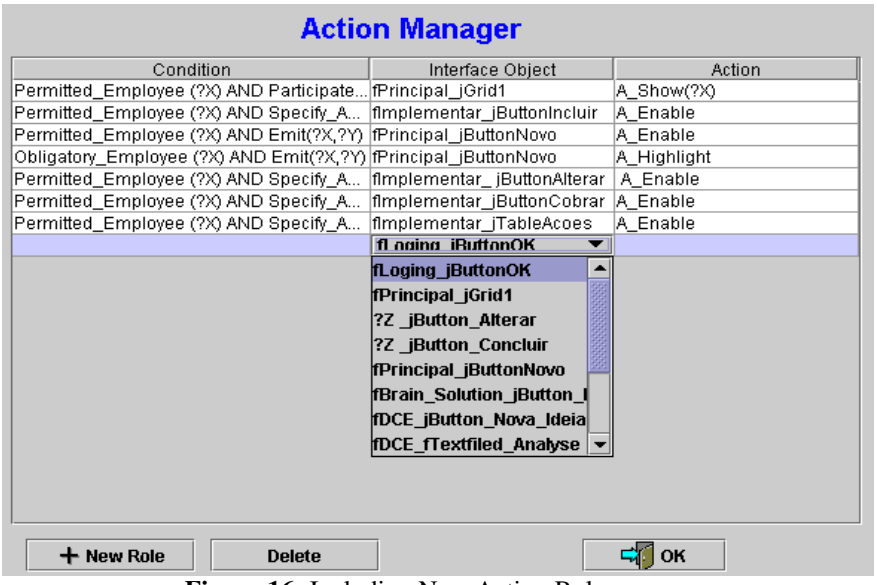


Figure 16: Including New Action Roles

To include a new action it is necessary to fill out three fields:

1. *Condition*: The condition must be related to the norm actions to have some effect in the interface; to do this we have to use a pattern: first it is necessary to include the deontic operator “Permitted_”, “Obligated_” or “Prohibited_” followed by the agent (e.g. “Employee (?X)”); then the action specified in the norm manager, such as: “Specify_Action(?X,?Y)”. The condition “Permitted_Employee (?X) AND Specify_Action(?X,?Y)” means that the rule will fire when a employee has the permission to specify actions;
2. *Interface Object*: An interface object can be selected through a combo box;
3. *Action:* An interface object action can be selected through a combo box; it is the action to be executed whenever the condition is true.

Examples of action rules that can be written through the Action Manager are:

- *Condition:* “Permitted_Employee (?X) AND Specify_Action(?X,?Y)”, *Interface Object:* “?Z _jInclude_Action” and *Action:* “Enable”, means that if a employee has the permission to specify actions than all the buttons with the identification “jInclude_Action” will be enabled.
- *Condition:*“Obligated_Employee (?X) AND Specify_Action(?X,?Y)”, *Interface Object:* “?Z _jInclude_Action” and *Action:* “Highlighted”, means that if a employee has the obligation to specify actions than all the buttons with the identification “jInclude_Action” will be highlighted.
- *Condition:* “Obliged_Card_Holder (?C) AND Pay_Interest(?C,?X)”, *Interface Object:* “jfPayment” and *Action:* “Show”, means that if a card holder is obliged to a pay the interest than the form with the identification “jfPayment” will appear.

Figure 17: Exit and Translate Buttons

To remove an action rule we have to select the rule that we want to remove and click in the button with the label “delete” (left part of figure 16). Be aware that the last version of the ICE does not have the “undo” functionality.

It is necessary to translate the rules to have some effect in the End_User interface; to do this task it is necessary click the button translate (right side of figure 17). To exit it is necessary to click in the button Ok. In next sections we assume that the reader has some knowledge about the java language.

3.1.5 Linking the Inference Machine (JESS) and the ICE database with the System

The first step is to include the archive “jess.jar” in the java “classpath", and import the jess classes “import jess.*;”. The two classes necessary to use ICE are:

1. The “Rete” class that is an inference machine, each object instantiated of this class is a different inference machine. To execute Jess commands to be

executed by the inference machine is necessary to use the "executeCommand" method.

2. The "Value" class , which is used in the ICE to transfer the results of the inference to the java variables values through the "stringValue" method.

The full JESS documentation, licence information (free for academic use) and download version can be found at: http://herzberg.ca.sandia.gov/jess/

The ICE stores the compiled rules in two tables the "Compil_Norms" and "Compil_Roles" of a Microsoft Access database. The first has two fields the "Norm_Number" (long integer type) which is a unique identification of the norm and the "Norm" (memo type) that is the norm Jess code. The other table also has two fields "Role_Number" (long integer type) which is a unique identification of the rule and the "Role" (memo type) that is the rule Jess code.

It is necessary make the ODBC-Java Bridge; Figure 18 shows a code implements it. If you use this code to do the connection it is necessary to instantiate the object and to use the "get_connection" method. If you are not familiar with the use of database in java, you can check the SUN Java documentation at site: http://developer.java.sun.com/developer/infodocs/.

```
import java.sql.*;
import com.borland.dx.sql.dataset.*;

public class Conect_Norma_DB {
  private  java.sql.Connection BDconnection;
  public Conect_Norma_DB() {
    try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      String url = "jdbc:odbc:NormInter";
      BDconnection = java.sql.DriverManager.getConnection(url, "", "");
    }
    catch (java.lang.Exception eBd) {
      System.out.println("Conection Error NormInter: " + eBd);
      eBd.printStackTrace();
    }
  }

  public void Reconect () {
   try {
         BDconnection.close();
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         String url = "jdbc:odbc:NormInter";
         BDconnection = java.sql.DriverManager.getConnection(url,"","");
    }
    catch (java.lang.Exception eBd) {
         System.out.println("Conection Error NormInter (Reconect): " + eBd);
         eBd.printStackTrace();
    }
  }

  public java.sql.Connection get_connection() {
    return BDconnection;
  }
}
```

Figure 18: Java code to connect with the IC data base

To load the Norms and Rules to the JESS inference machine it is required to read all the norms and rules from the data base and use the "executeCommand" method of the Rete

class to each line recovered from the database. Figure 19 have fragments of Java code that can be used to load the Norms from the database (a similar code can be used to load the rules by changing the table and field names). After the execution of each rule at the inference machine (lower part of figure 19) is also necessary to execute the command "(run)".

```
public Vector Load_All_Norms_From_DB() {
    Vector Norms = new Vector();
    String Norm_Jess = new String();
     Conect_Norma_DB Norma_Bank =  new Conect_Norma_DB ();
     try {
        BDconnection = Norma_Bank.get_conexao();
        vStm = BDconnection.createStatement(); // Cria um Statement
        String Query = new String("SELECT * FROM Compil_Norms");
        java.sql.ResultSet rs = vStm.executeQuery(Query);
        while (rs.next()==true) {
          Norm_Jess = rs.getString("Norm");
          Norms.addElement(Norm_Jess);
        }
        return Norms;
    }
    catch (java.lang.Exception eBd) {
          System.out.println("Load_All_Norms Error: " + eBd);
          eBd.printStackTrace();
     }
}
```

```
…
Rete KB_Norms = new Rete(); //(part of class attributes)
…
  public void Load_Norms_Rete () {
    Vector Norms = new Vector();
    Norms = Load_All_Norms_From_DB ();
    try {
       for (int i = 0; i < Norms.size(); i++) {
       KB_Norms.executeCommand( (String) Norms.elementAt(i));
       KB_Norms.executeCommand("(run)");
      }
    }
    catch(Exception e) {
      System.out.println("Exception occured Load_Norms_Rete  : " + e);
    }
}
```

Figure 19: Loading norms from the Database

3.1.6 Informing the Inference Machine and Receiving the Inference Values

We proposed the construction of a "perception method" and "action method". The "perception" method receives the data from the end-user interface context, transmits to the inference machine, receives the inference results and executes the "action method" that interprets it and executes the changes at the interface. The following steps describe the "perception method" (see code at figure 20):

1. Bind the "action_plan" with a "default" action plan (e.g. NOTHINGTODO). The NBIC includes all the actions to be done by the user in the "action_plan" variable. The code implements this step is: *KB_Norms.executeCommand("(bind ?*action_plan* NOTHINGTODO)");*

2. Execute the inference procedure. *KB_Norms.executeCommand("(run)");*
3. Include the new data in the knowledge base: KB_Norms.executeCommand(New_Data);
4. Execute the inference procedure. *KB_Norms.executeCommand("(run)");*
5. Transfer the value of the action_plan variable from the inference machine to a "Value" object. *v = KB_Normas.executeCommand("?*action_plan*");*
6. Execute the Action method passing a string with the action plan. *Action (v.stringValue(KB_Normas.getGlobalContext()));*

```
…
Rete KB_Norms = new Rete(); //(part of class attributes)
…

public void Perception(String New_Data) {
   Value v;
   try {
     KB_Norms.executeCommand("(bind ?*action_plan* NOTHINGTODO)");
     KB_Norms.executeCommand("(run)");
     KB_Norms.executeCommand(New_Data);
     KB_Norms.executeCommand("(run)");
     v = KB_Normas.executeCommand("?*action_plan*");
     Action (v.stringValue(KB_Normas.getGlobalContext()));
   }
   catch(Exception e) {
     System.out.println("Exception occured Perception  : " + e);
   }
  }
```

Figure 20: Perception Method Example

To inform the context to the inference machine it is necessary to use the "assert" command of the Jess language, for example if we want to inform the system that John is an employee, we can use the perception method in the following way: *Perception("(assert (Employee John))").*

To retract facts from the knowledge base we use the "retract" command of the Jess language, for example if we want to inform the system that John is not an employee anymore, we can use the perception method in the following way: *Perception("(retract-string \"(Employee John)\")")*. This command is important by the fact that the Rete algorithm maintains all the inferred facts in the knowledge because of the time optimisation.

The nomenclature to describe the percept elements must be the same of the norms, for example if we use the "Employee (_X)" logical atom during the norm description we can not use "*Worker (John)*". The following examples illustrate the perception mechanism:

1. If we have the Norm:
   Condition: Employee (?X) AND Problem (?Y) AND Responsible(?X,?Y)
   Modal: Permitted
   Agent: Employee (?X)
   Action: Specify_Actions (?X,?Y)
   And IF we inform:
   Perception("(assert (Employee John))")
   Perception("(assert (Problem X11))")
   Perception("(assert (Responsible John X11))")
   The inference machine will infer that the employee has the permission to specify actions.

If we have the Role:
Condition: "Permitted_Employee (?X) AND Specify_Action(?X,?Y)"
Interface Object: "?Z _jInclude_Action"
Action: "A_Enable"
The inference machine will infer that all "include action button" must be enabled to the user named John. The inference machine includes the following actions in the Jess "?*action_plan*" variable:
Asser-?Z_ jInclude_Action-A_Enable //NOTHINGTODO
(where NOTHINGTODO is the default action)

2. If we have the Norm:
Condition: Card_Holder (?C) AND Outstanding_Credit(?C,?X) AND Past_Posting_Days(?X,?D) AND (?D>25) AND (?X>0)
Modal: Obligated
Agent: Card_Holder (?C)
Action: Pay_Interest(?C,?X)
And if we inform:
Perception("(assert (Card_Holder John))")
Perception("(assert (Outstanding_Credit John 100))")
Perception("(assert (Past_Posting_Days 100,30))")
The inference machine will infer that the Card Holder have the obligation of pay the interest of £100
If we have the Role:
Condition: "Obligated_Card_Holder (?C) AND Pay_Interest(?C,?X)"
Interface Object: "jfPayment"
Action: "A_Show"
The inference machine will infer that the payment screen must be shown to the user. The inference machine includes the following actions in the Jess "?*action_plan*" variable:
Asser-?Z_jInclude_Action-A_Show//NOTHINGTODO
(where NOTHINGTODO is the default action)

3.1.7 Linking the Inferences with Interface Changes

The last step is the construction of the *action* part of the end-user interface. We propose the implementation of an *action* method that receives the actions from the perception method (Figure 20), interprets them and loads other methods that make the interface changes. For example if the action method receive "*Asser-?Z_ jInclude_Action-A_Enable //NOTHINGTODO"*, it has to interpret the message and load the method that enable all the buttons.

The BNF of the "action plan" is:
Action_Plan ::= {<Action>//}<Default_Action>
<Action> ::= Asser-<Interface_Object>-<Object_Action>
<Interface_Object> ::= {<Variable_Name>_}<Object_Name>
<Object_Action> ::= <alphabet_l>{<alphabet_l>}
<Object_Name> ::= <alphabet_l>{<alphabet_l>}
<Default_Action> ::= <alphabet_l>{<alphabet_l>}
<Variable_Name> ::= ?<alphabet_l>{<alphabet_l>}
<alphabet> ::= a|...|z|A|...|Z||0|...|9|

Each action of the action plan is separated by "//" and the last action is the default action. All the actions included by the ICE have the "Asser-" identification, the interface object and their actions separated by a "-". Therefore to facilitate the action interpretation we recommend to not use the character "-" in the names of objects and actions.

What is necessary is a parser that read this BNF description and chooses the methods that make changes at the end-user interface. The implementation of these methods (or classes if necessary) is dependent of the end-user application architecture and the technology used to implement the changes. Regarding that architecture described in this document; it does not define the technology used to implement the action. Example of how to implement the changes are from a simple "case" and object setting to "java reflection", component binding or dynamic code generation to implement more complex interface changes.

3.2 Maintenance of the Norms

The maintenance of the end-user interface is facilitated by the fact that it is not necessary to code changes every time that a norm changes. To change a norm it is necessary open the Norm Manager, to click in the affordance that the norm is associated, select a norm and make the changes following the guidelines to describe norms through the Norm Manager, as explained before. Inclusion and exclusion can be done through the "new norm" and "delete" button respectively (see figure 11). After changing the norms, translate them using the "translate" button (see figure 13). The modifications will have immediate effects on the inference machine and consequently on the end-user interface.

Although end-user interface changes by changing norms are relatively simpler than code changes, we have to be aware of the limitation of the possible changes. The set of possible actions associated to the norms is limited by the actions rules, because if there is not any rule that translate the action to the interface objects, no changes at the interface will occur.

If a new action is necessary, it is possible to write new roles to link them with the interface changes. The constructions of these new roles are limited by the action mechanism functionalities; for example: it is not possible to add a rule that includes a new button in a form if there is not an action method that implements it. After that you can include new code to deal with the new action, but is important to notice that more powerful action methods implementations lead to more flexible action rule changes and expansions.

Therefore the maintenance of the norms is divided into three layers, the first one is necessary when only norm changes; the second is also necessary on rule changes and in the third code changes must be done. Empirical data suggests that norms and rule changes are more frequent that code changes, even for a limited implementation of the action mechanism (Bonacin et. al ., 2003).

Acknowledgments

This work was partially supported by grants from Brazilian Research Council (CAPES BEX2214/02-4, CNPq 301656/84-3 and FAPESP 2000/05460-0). The authors also thank Delphi Automotive Systems in Jaguariúna, Brazil, and the University of Reading for collaboration and partnership.

References

Areces C., de Rijke, M. & de Nivelle, H. Resolution in Modal, Description and Hybrid Logic. *Journal of Logic and Computation*, Vol 11, Issue 5: October 2001.

Bonacin, R. and Baranauskas, M. C. C., Semiotic Conference: Work Signs and Participatory Design, Proceedings of 10th International Conference on Human - Computer Interaction, 2003, to appear.

Bonacin, R., Baranauskas, M. C. C. and Cecilia, R. M. Designing and Learning: joining the concepts in work practices. In Paola Forcheri and Alfonso Quarati (eds.), Journal of International Forum of Educational Technology & Society and IEEE Learning Technology Task Force, 2003, v. 6, n. 1, 2003, 3-8. (Online Version: http://ifets.ieee.org/periodical/)

Filipe, J. B. L. *Normative Organisational Modelling using Intelligent Multi-Agent System.* Phd thesis Staffordshire University, 2000.

Goré, R. P. *Cut-free Sequent and Tableau Systems for Propositional Normal Modal Logics*, Phd. thesis University of Cambridge, 1992.

Liu, Kecheng. *Semiotics in information systems engineering*, Cambridge University Press, 2000.

Muller, M. J., Haslwanter, J. H., & Dayton, T. "Participatory Pratices in the Software Lifecycle". In *Handbook of Human-Computer Interaction*, M. Helander, T. K. Landauer and P. Prabhu, eds., Elsevier Science, 2 ed,1997, pp. 255-297.

Russell, S. J. Artificial intelligence: a modern approach. Prentice Hall, (1995).

Russell, S. J. Artificial intelligence: a modern approach. Second Edition, Prentice Hall, (2003).

Stamper, R. K., Althaus, K., Backhouse, J. "MEASUR: Method for Eliciting, Analyzing and Specifying User Requirements". In *Computerized Assistance During the Information Systems Life Cycle.* Olle, T. W., Verrijn-Stuart, A. A. and Bhabuts, L., eds., Elsevier Science, Amsterdam, 1988, p.67-116.

Stamper, R. K. "Information Systems as a Social Science: An Alernative to the FRISCO Formalism". In *Information System Concepts: an Integrated Discipline Emerging,* E. D. Falkenberg, K. Lyytien and A. A. Verrijn-Stuart, eds, Kluwer Academic Publishers, USA, 2000, p.1-51.