UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Um Framework Multi-Agentes para o Projeto de Interfaces Homem-Computador

Autora: **Cecilia Inés Sosa Arias Peixoto**
Orientadora: **Dra. Beatriz Mascia Daltrini**

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como requisito parcial para obtenção do Título de Doutor em Engenharia Elétrica.

26 de junho 1997

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

P359f

Peixoto, Cecilia Inés Sosa Arias
Um framework multi-agentes para o projeto de interfaces homem-computador / Cecilia Inés Sosa Arias Peixoto.--Campinas, SP: [s.n.], 1997.

Orientadora: Beatriz Mascia Daltrini.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Interação homem-máquina. 2. Interfaces de usuário (Sistema de computador). 3. Sistemas operacionais distribuidos. 4. Inteligência artificial. I. Daltrini, Beatriz Mascia. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Este trabalho contou com o apoio financeiro da CAPES

À minha família

AGRADECIMENTOS

Quero agradecer, de maneira especial, à minha orientadora Beatriz M. Daltrini, pela sua dedicação ao trabalho e virtudes humanas com que enriqueceu estes anos de estudo.

A Mauren F. Brenner, grande amiga, pela sua ajuda incondicional, muito obrigada.

Meus sinceros agradecimentos à familia Piñón Pereira Diaz.

Aos meus pais pelo permanente incentivo.

Aos companheiros do grupo de Interfaces, em especial a Elton J. da Silva e Oscar. F. de Carvalho.

A Sílvia R. Vergilio, Ruth Campomanes e Carolina A. Godoy pela valiosa ajuda.

A todos os meus companheiros de curso, por esses agradáveis anos juntos.

Sumário

Obter sucesso no projeto de interface de usuário requer coordenação de diversas disciplinas para criar um produto de software de alta usabilidade. Esta é a motivação deste trabalho: definir a arquitetura de um *framework* que permita integrar o trabalho das diversas disciplinas durante o processo de projeto. O objetivo é obter um projeto de sistema interativo onde diversos usuários possam trabalhar considerando um aspecto específico da interface. Neste trabalho de tese, define-se um *framework* para o projeto conceitual e detalhado de sistemas interativos, o qual segue a abordagem multi-agentes na sua arquitetura. Ele possui as características multi-disciplinares tais como cooperação e flexibilidade, próprias do processo de desenvolvimento de interfaces. Para conseguir que várias pessoas trabalhem com vários agentes, determinamos que o projeto da interface seja tratado também como um conjunto de agentes, chamados de agentes da interface da aplicação (AIA). Os AIAs reúnem, na sua arquitetura, as descrições declarativas de todas as partes do projeto de uma aplicação interativa (apresentação da interface, descrição das tarefas da interface, descrição do diálogo, modelos de funções e dados) naturalmente orientada ao projetista.

Abstract

Attaining success in user interface design requires coordination of several subjects to create a software product with high usability. This is the motivation of this work: to define the architecture of a framework which allows integration of the work of the various subjects during the design process. The goal is to obtain an interactive system design where several users can work considering only one specific aspect of the interface design. In this thesis the framework for conceptual and detailed design of interactive systems is defined. This framework has the typical multidisciplinary characteristics of the interface design process, such as cooperation and flexibility. In order to enable different people to work with various agents in the design definition, we have decided to see the interface design as a set of agents too, called application interface agents (AIA). The AIAs include, in their architecture, the declarative description of all parts of the design of an interactive application (presentation of the interface, dialog description, interface task description, data and functions model) in a natural, designer-oriented way.

Sumário

Lista de Figuras

1

Introdução

1.1 Interfaces de Usuário: Uma Visão Crítica

Interfaces homem-computador tornaram-se mais importantes com o aumento do número de usuários e de aplicações. Os últimos dez anos viram uma mudança nas perspectivas de projeto, avaliação e uso de sistemas interativos. O campo de interação homem-computador (IHC)[1] passou de um componente pequeno do sistema ao foco da atenção de pesquisadores de várias áreas, incluindo psicologia e ciências sociais. Estudos e investigações levaram a uma evolução gradual do conceito da "interface" e do processo de desenvolvimento de software. Como resultado, o desenvolvimento de sistemas interativos é considerado como parte integral do processo de engenharia de software.

Hoje, considera-se que IHC assume um papel cada vez mais importante dentro da engenharia de software. Ela requer o entendimento do usuário, da tarefa a ser realizada, dos fatores que podem afetar como e quando o usuário irá interagir com o sistema, e as soluções de hardware e software que podem ser oferecidas.

A interação homem-computador está muito relacionada com o processo de projeto, provendo soluções que levam em consideração todas as restrições e requisitos. O aspecto de projeto da IHC é importante e não se aplica apenas a hardware ou a software. Em IHC o que é projetado é um sistema que inclui os usuários, juntamente com hardware e software. O propósito do projeto de uma interface é possibilitar que a tarefa que está sendo automatizada seja executada mais eficientemente e com maior satisfação [Joh 92].

Para isto, o desenvolvimento de interfaces de usuário deve considerar o projeto da comunicação entre o usuário e o sistema, o material gráfico e textual, a

[1] *Human-Computer Interaction* (HCI) é definida como o que acontece quando o usuário humano e um sistema de computação executam tarefas de modo cooperativo [Hix 93].

informação e as tarefas, como também o software através do qual os resultados de todas as outras decisões de projeto irão ser refletidos.

A ênfase emergente em usabilidade tenta coordenar estas características. Com elas, temos a possibilidade de investigar as propriedades do processo de projeto de sistemas interativos, o que traz grandes expectativas tanto para aplicações como para pesquisa básica.

1.2 Este Trabalho

Inúmeros artigos na área de interfaces homem-computador mostram a necessidade de se reduzir o *gap* entre os usuários e os projetistas quando estes trabalham juntos no desenvolvimento da interface. Se essa redução for conseguida, problemas relacionados à documentação, ajudas (*helps*) e compreensão do sistema poderão ser resolvidos com maior facilidade.

Identificamos que a redução deste *gap* pode ser conseguida através de um ambiente de desenvolvimento de interfaces (*framework*) que opere de acordo com o modo de pensar das pessoas [Nor 86].

Nosso *framework* é uma base conceitual que integra e põe à disposição do usuário uma série de conhecimentos, de ferramentas especialmente escritas para o ambiente e de ferramentas com as quais o usuário está acostumado. A multi-disciplinaridade das ferramentas permite que atividades profissionais tão distintas como a dos projetistas, psicólogos, *designers*, etc., possam ser reunidas de forma a trabalhar cooperativamente para o desenvolvimento da interface.

Duas linhas de pesquisa foram investigadas para se definir a arquitetura interna do *framework*:

1. **Sistemas Multi-Agentes**
   Os sistemas multi-agentes possuem características próprias tais como paralelismo, inteligência e aprendizagem;

2. **Técnicas de Especificação de Interfaces**
   Metodologias que permitem determinar como a interface do usuário pode ser especificada e desenvolvida dentro do ambiente.

Estas linhas de pesquisa dão o arcabouço conceitual deste trabalho. Como dito anteriormente, pretende-se apresentar à área de interfaces homem-computador um ambiente onde flexibilidade, cooperação e multi-disciplinaridade sejam suas características principais.

Definimos a arquitetura interna dos agentes tendo em consideração aspectos provenientes do funcionamento do sistema nervoso central humano. Este paralelo com o sistema biológico facilita a identificação dos agentes como unidades autônomas dentro do *framework* e permite definir um comportamento próprio para cada agente.

Considerando estas características, e a bibliografia do estudo da interação homem computador, foram levantados os seguintes objetivos específicos para o ambiente. A saber:

1. **Flexibilidade**
   As tecnologias correntes de interface de usuário geralmente impõem um conjunto rígido de restrições sobre como a aplicação deve ser estruturada [Tay 95]. É possível identificar a flexibilidade do *framework* em termos das diferentes ferramentas que possam ser incorporadas, tais como *hypercards* [New 95], etc.;

2. **Multi-disciplinaridade**
   O processo de desenvolvimento de sistemas interativos é uma atividade multi-disciplinar que utiliza conhecimentos derivados de várias áreas da ciência, engenharia e arte [Joh 92]. A abordagem multi-agentes vem de encontro a essa característica através da existência de diferentes agentes com diversas funcionalidades;

3. **Cooperação**
   Esta característica está relacionada principalmente com a definição de sistemas multi-agentes. Um agente que não possa solucionar seu próprio problema local (ou porque não tem habilidade suficiente ou porque não pode atingir a performance requerida) pede a outros que cooperem com ele na busca da solução [Sic 92];

4. **Artefatos de software reutilizáveis**
   A chave para a reusabilidade de software é estruturar os sistemas de modo que eles usem componentes modulares com uma interface bem definida. A arquitetura interna dos agentes adere a este princípio;

5. **Múltiplos look-and-feel**
   A "forma" através da qual os usuários interagem com o computador equivale ao vocábulo *feel*. *Look* refere-se à apresentação [Luc 92]. Ter a possibilidade de especificar uma interface de aplicação e apresentá-la com diferentes estilos, levou à identificação de uma maneira modular de trabalhar. Assim, uma mesma aplicação pode ter um *look-and-feel* diferente simplesmente comunicando-se com um agente do ambiente que usa um estilo diferente na construção dos protótipos da interface da aplicação;

6. **Robustez**
   Provê a separação do assunto em dois níveis, entre a aplicação e o software da interface e entre o software da interface e o sistema de janelas e *toolkits*[2];

[2] *Toolkits* são bibliotecas de rotinas, usadas pelos programadores, para implementar a interface. A utilização de *toolkits* livra o programador de detalhes de mais baixo nível encapsulando o *look and feel* de técnicas de interação utilizadas em aplicações interativas. Os tipos de dados ou objetos básicos utilizados no encapsulamento são chamados de *widgets*.

7. **Especificação do projeto da interface**
   Através de diferentes documentos que são gerados pelos agentes e que incluem textos, gráficos e regras de decisão.

Ressaltamos que o item 7 é a motivação da definição de um ambiente com esse tipo de arquitetura, a qual já propicia de forma natural uma solução adequada aos demais itens. Aprofundando-nos nele, podemos definir certos objetivos, a saber:

1. Explorar como é possível construir sistemas interativos que considerem a multi-disciplinaridade do projeto;
2. Identificar como diversas metodologias podem ser incluídas nas descrições do projeto;
3. Entender o processo de desenvolvimento de sistemas interativos e a conexão entre as diversas fases no ciclo de vida.

E as seguintes características que devem fazer parte deste ambiente. A saber:

1. Aceitar a especificação da análise como entrada do ambiente seguindo a metodologia proposta por Hix & Hartson [Hix 93]. A análise está composta da análise de necessidades, análise do usuário, análise das tarefas, análise funcional e alocação de tarefa-função;
2. Gerar descrições que farão parte da especificação da interface da aplicação, conforme as diversas disciplinas envolvidas durante o desenvolvimento;
3. Validar a especificação conforme os *guidelines* de projeto de sistemas interativos;

Os itens acima justificam e fornecem motivações para a execução deste trabalho e são aprofundados nos capítulos seguintes.

1.3 Organização do Texto

No capítulo 2, **Introdução à Interação Homem-Computador**, é apresentada uma visão geral do desenvolvimento de sistemas interativos e são discutidos os conceitos envolvidos no seu desenvolvimento. Por último, é realizada uma comparação entre o ciclo de vida tradicional de software e o ciclo de vida para sistemas interativos.

No capítulo 3, **Visão Geral sobre os Sistemas Multi-Agentes**, é apresentado um resumo dos estudos gerais sobre a abordagem multi-agentes. Neste capítulo são fornecidos os conceitos fundamentais e as vantagens da aplicação desta abordagem.

No capítulo 4, **Ambiente Multi-Agentes para Projeto de Interfaces**, é apresentado o *framework* proposto. Neste capítulo propõe-se uma arquitetura interna para os agentes constituintes do *framework* e descreve-se seu funcionamento geral.

No capítulo 5, **Arquitetura Interna dos Agentes**, é apresentada uma breve descrição dos componentes principais da arquitetura interna dos agentes.

O capítulo 6, **Funcionamento dos Agentes do Framework**, contém os detalhes do funcionamento dos agentes do *framework* com ênfase naqueles agentes que têm relação direta com a estruturação do projeto da interface.

No capítulo 7, **Exemplo de Aplicação**, é apresentado, através de um exemplo, o funcionamento do ambiente no processo de projeto da interface da aplicação segundo a abordagem do *framework*.

No capítulo 8, **Conclusão**, são apresentadas as contribuições e perspectivas de continuação deste trabalho.

2

Introdução à Interação Homem-Computador

Nos últimos anos, os sistemas de computação têm mudado de enfoque no que diz respeito ao comportamento do usuário. Os primeiros sistemas forçavam o usuário a se comunicar de uma maneira que fosse fácil de implementar em hardware ou em software. Isto não resultava sempre na melhor forma de comunicação no contexto humano [Pre 92].

Hoje, grande parte dos recursos computacionais é dedicada exclusivamente para tornar mais fácil a vida do usuário. A comunicação com o sistema se tornou pelo menos tão importante quanto a computação executada por ele.

Por sua vez, os usuários estão cada vez mais exigentes e descartam qualquer interface que seja difícil e desconfortável de usar pois, para eles, a interface é o sistema [Nor 86]. Assim, para garantir a usabilidade de um sistema, devemos incluir preocupações com a interface e com todo o processo de desenvolvimento de software.

Este capítulo apresenta a importância da interface homem-computador, com ênfase no desenvolvimento de software. O capítulo discorre sobre as componentes da interface, atividades de desenvolvimento e ciclo de vida, assim como as metodologias mais utilizadas e ambientes de desenvolvimento.

2.1 Importância da Interface Homem-Computador

Vários autores concordam que desenvolver interfaces amigáveis é uma tarefa difícil, longa e custosa. Existem razões para esforços na busca de melhorias na interação homem-computador:

- A atitude do usuário está mudada; o usuário, que antes era limitado pela interface, agora, com mais conhecimento, quer ver no computador um companheiro de trabalho produtivo, ou seja, que não limite sua inteligência;
- A interface de usuário é altamente visível para todos os usuários de um sistema e é um dos principais critérios de aceitação de um sistema de software [DeS 89];
- A inovação tecnológica aumenta a capacidade de trabalho; ela se manifesta na melhoria do desempenho do sistema.

O desenvolvimento de interfaces deve responder a estas exigências e estar em sintonia com a evolução tecnológica através de um duplo esforço em direção à adaptação: satisfazendo as necessidades crescentes dos usuários e levando em consideração uma tecnologia altamente evolutiva [Cou 91].

Projetar a interface de um sistema também significa "projetar o usuário", no sentido que as entradas que o sistema reconhece restringe as ações dos usuários, e as saídas definem a informação que o usuário manipula. O projeto da interface de usuário é uma tarefa delicada [Nor 94].

De acordo com Baecker & Buxton [Bae 87] o projeto da interface de usuário requer um projeto gráfico e industrial, entendimento da cognição humana, percepção e habilidades, conhecimento de tecnologia de tela, dispositivos de entrada, etc.

Obter sucesso no projeto de interface de usuário requer coordenação de diversas disciplinas. Assim, com o objetivo de criar produtos de software de alta usabilidade, os engenheiros de fatores humanos podem contribuir para o projeto, que pode ser implementado por engenheiros de software. Documentadores técnicos podem projetar e desenvolver a documentação e ajuda *on-line*. Especialistas em educação podem desenvolver treinamento do usuário, o qual deve ser integrado com o pacote de documentação e ajuda. Telas gráficas de alta resolução estão criando novas oportunidades para projetistas gráficos e industriais. Representantes de *marketing* podem usar sua familiaridade com os clientes e com produtos competidores para determinar especificações mais precisas [Gru 89].

Todos estes aspectos dão uma idéia da característica multi-disciplinar que deve ser considerada por uma equipe integrada de desenvolvimento.

2.2 Desenvolvimento das Interfaces

Na maioria dos sistemas interativos atuais, uma média de 70% do código é referente ao desenvolvimento da interface com o usuário [Pre 92]. Do ponto de vista

dos engenheiros de software, a interface não é, em realidade, parte integrante do sistema interativo como um todo, mas somente uma caixa que pode ser acoplada separadamente ao processo. A ênfase emergente em usabilidade está mudando esta perspectiva.

A interface tem se tornado uma parte crítica do sistema interativo como um todo, e o seu desenvolvimento considerado como parte integrante do processo de engenharia de software [Hix 93].

Este novo enfoque levou à identificação das atividades de desenvolvimento que fazem parte da engenharia de software e aquelas atividades do desenvolvimento relacionadas ao estudo da interação homem-computador.

2.2.1 Atividades do Processo de Desenvolvimento de Sistemas Interativos

Existem diferenças e semelhanças entre o desenvolvimento da interação com o usuário e o desenvolvimento tradicional de software.

Devido às interfaces com o usuário serem implementadas com o uso de software, muitos engenheiros acreditam que as "técnicas bem estabelecidas" para desenvolvimento de software podem ser aplicadas ao desenvolvimento de interfaces.

Certamente, os mesmos tipos de conceitos podem ser aplicados aos dois tipos de desenvolvimento (definição de requisitos, especificação, projeto, métricas, avaliação, manutenção, documentação e ciclo de vida) mas, para a interação com o usuário, cada uma dessas atividades deve ser feita de uma maneira diferente, pois um enfoque maior deve ser dado ao elemento humano.

Tendo em consideração estes aspectos, Hix & Hartson [Hix 93] discutiram a natureza das atividades genéricas que compõem o desenvolvimento de um software. Elas podem ser divididas em dois grandes grupos:

1. **Atividades no desenvolvimento de software**
   Relacionadas ao desenvolvimento do software;

2. **Atividades no desenvolvimento de interfaces com o usuário**
   Relacionadas ao desenvolvimento do projeto de interação.

Ambas as atividades partem de uma base comum, a análise do sistema (Figura 2.1). A partir da mesma, existem dois caminhos a se tomar. Um relacionado às atividades próprias do desenvolvimento do software tradicional e o outro relacionado às atividades do desenvolvimento de interfaces com o usuário. Vemos, na Figura 2.1, que são obtidos produtos de software ao finalizar cada caminho. Ambos são um ponto de partida para o processo de teste/avaliação que é novamente compartilhado entre as atividades de desenvolvimento do software da interface e do não referente à interface.

A Figura 2.2 foi completada com a adição das atividades correspondentes a cada um dos caminhos. Essa Figura mostra o paralelismo entre as atividades de

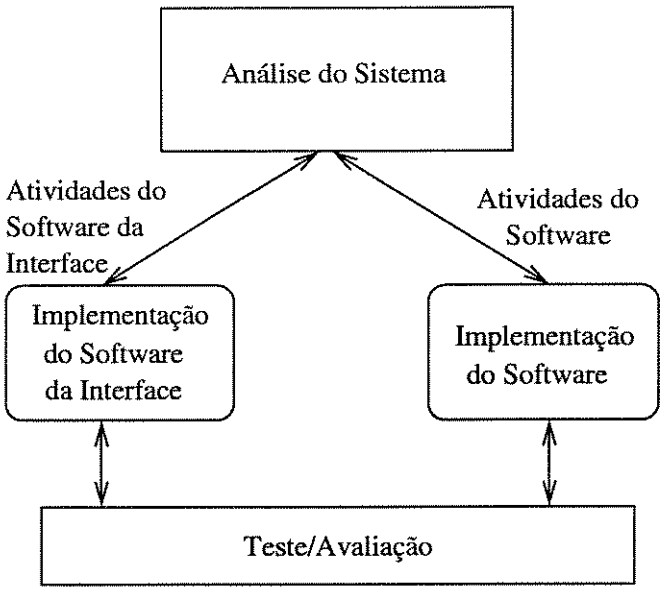

Figura 2.1: Modelo Genérico do Desenvolvimento de Sistemas Interativos

desenvolvimento da interface e do desenvolvimento do software não pertencente à interface. Também mostra a distinção entre esses caminhos paralelos de desenvolvimento. Na realidade, existe uma necessidade de mais canais de comunicação verticais conectando caixas dos dois caminhos, comunicando os desenvolvedores da interface e os desenvolvedores da funcionalidade computacional conectada à interface.

Na Figura 2.2 as caixas de prototipagem rápida e avaliação formativa são de grande importância no desenvolvimento de sistemas interativos. Através da prototipagem rápida, um desenvolvedor de interação tem a oportunidade de avaliar projetos propostos muito cedo no ciclo de vida e de assegurar que a usabilidade esteja presente no projeto em evolução.

Avaliação formativa feita em cada ciclo de iteração produz dados quantitativos através dos quais os desenvolvedores podem comparar as especificações estabelecidas de usabilidade, e também produzir dados quantitativos que sejam usados para ajudar a determinar o que mudar para fazer com que o projeto da interação melhore sua usabilidade.

Atividades no Desenvolvimento de Software

As atividades do desenvolvimento de software podem ser esquematizadas conforme a Figura 2.3.

O resultado da análise do sistema é um conjunto de requisitos para os projetistas de software. Estes requisitos são declarações dos objetivos do sistema, necessidades, funcionalidade desejada, e características nas quais o projeto de software estará baseado.

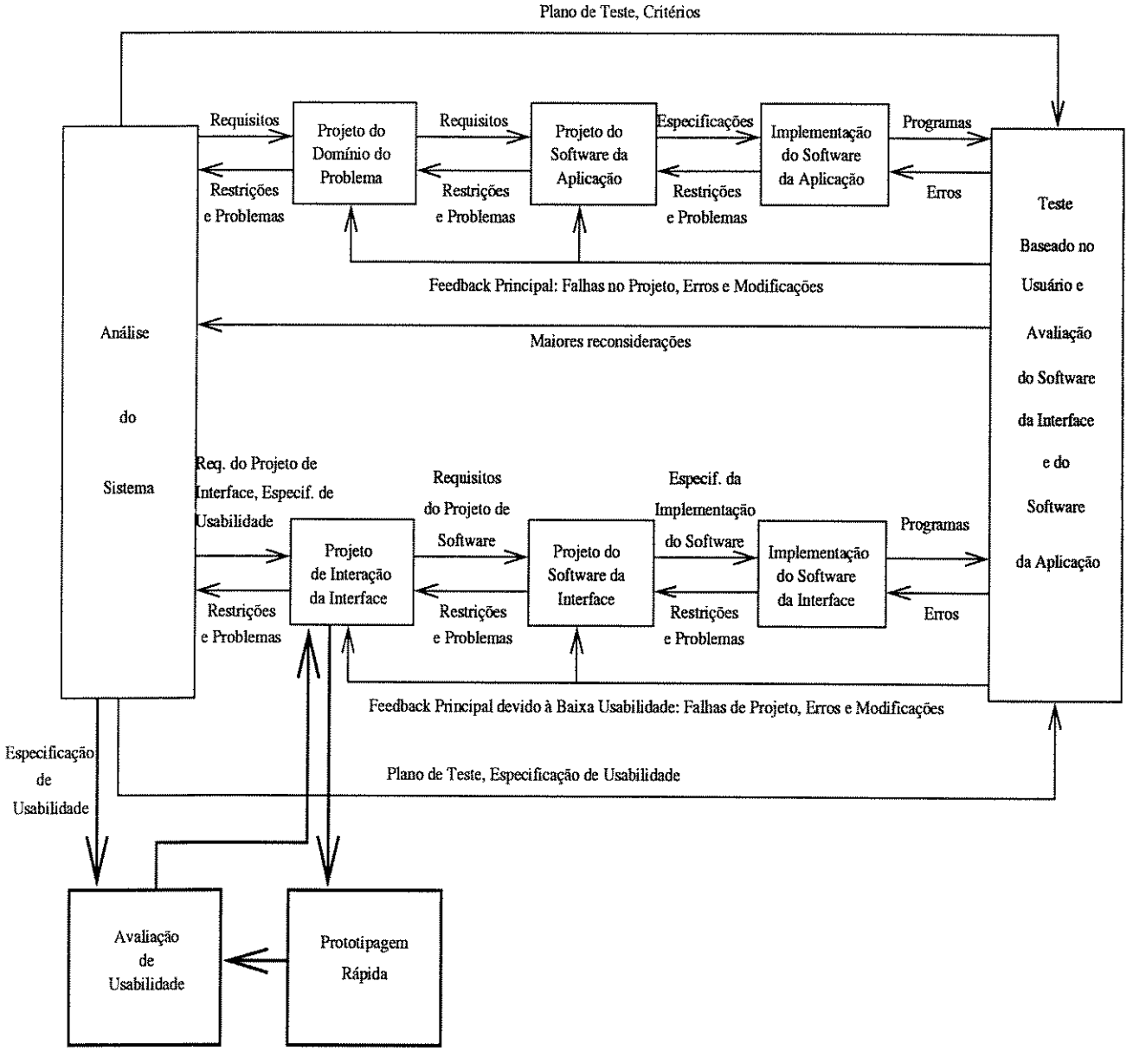

Figura 2.2: Atividades do Processo de Desenvolvimento de Sistemas Interativos (conforme [Hix 93], Pg. 113).

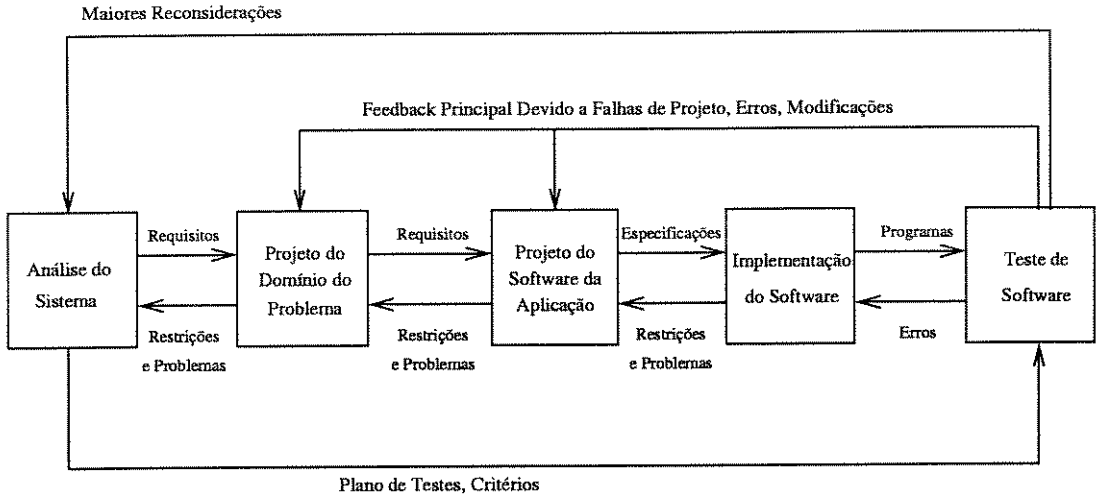

Figura 2.3: Atividades do Desenvolvimento de Software (conforme [Hix 93], Pg. 109).

O projeto do domínio do problema é um modelo formal da aplicação. O projetista do domínio do problema dá *feedback* ao analista de sistemas no que diz respeito aos requisitos faltantes e inconsistentes para o projeto do software.

O projeto do software é então o projeto das estruturas de dados e algoritmos para converter o projeto do domínio do problema em um programa de computador. O projetista do software provê *feedback* ao projetista do domínio do problema reportando qualquer inconsistência, omissão, ou ambigüidade nos requisitos do software.

O teste de software retorna possíveis erros ou problemas encontrados para os projetistas, implicando em modificação nas especificações, implementação, podendo requerer reconsideração na análise do sistema.

Atividades no Desenvolvimento de Interfaces com o Usuário

As atividades envolvidas no desenvolvimento do software da interface são análogas àquelas referentes ao desenvolvimento do software não ligado à interface, como mostra a Figura 2.4 [Hix 93].

Na Figura 2.4 podemos observar que o projeto de interfaces tem uma parte relacionada ao software e uma parte relacionada à interação com o usuário, independente do software.

O projeto de interação envolve ações do usuário, *feedback*, aparência da tela, tarefas do usuário, seqüenciamento e estilos de interação. Já, o projeto do software da interface envolve algoritmos, estruturas de dados, módulos, *toolkits*.

Os dois tipos de atividades contribuem para o desenvolvimento da interface do usuário mas requerem habilidades, atitudes, perspectivas, técnicas e ferramentas diferentes. Projetar o software, mesmo o software da interface, é direcionado ao

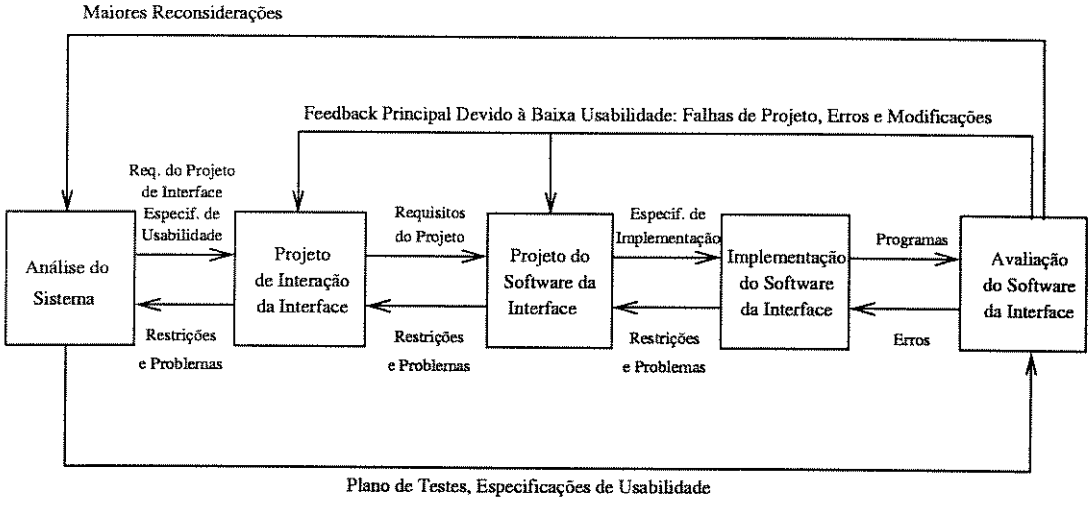

Figura 2.4: Atividades do Desenvolvimento de Interfaces (conforme [Hix 93], Pg. 110).

sistema, enquanto o projeto da interação é direcionada ao usuário [Hix 93].

Considerar ambas as atividades durante o desenvolvimento de sistemas é de fundamental importância para a obtenção de um sistema de alta usabilidade.

2.2.2 Ciclo de Vida Estrela

A idéia de que o ciclo de vida tradicional é inadequado para o desenvolvimento de sistemas interativos origina-se da natureza "tentativa e erro" do desenvolvimento de interfaces, que consiste de um processo cíclico. Projetistas de software ainda não conseguem prever as conseqüências de decisões de projeto no uso de um sistema. Prever o desempenho de uma interface é uma tarefa difícil, pois neste processo existe um "coprocessador de comportamento altamente imprevisível": o usuário. Em conseqüência, são indispensáveis rapidez e facilidade de modificação. Ainda não se conhecem princípios de projeto que, uma vez seguidos, garantam resultados satisfatórios. Desse modo, é freqüentemente mais efetiva uma abordagem de evolucionismo com a construção de protótipos.

Por esta razão, foi proposto o modelo em estrela [Har 89], mostrado na Figura 2.5. Este ciclo de vida tem como objetivo fornecer uma estrutura de desenvolvimento mais adequada para interfaces. Ele é o resultado de observações empíricas. As observações mostram que o ciclo de vida tradicional, *top-down*, isola as atividades de requisitos, projeto, implementação e teste; impor este paradigma conduz a interfaces de qualidade pobre. Uma das bases do modelo estrela está no que foi chamado de *alternating waves*. Refere-se ao fato de que o projeto de interfaces não é uma atividade nem *top-down*, nem *bottom-up*, mas uma mistura de ambas. As primeiras atividades de projeto são *bottom-up*, baseadas em cenários e freqüentemente ampliadas com representações da evolução do diálogo. Estão relacionadas

à visão do usuário. As atividades seguintes seriam *top-down*, relacionadas à visão do sistema. Contudo, o projetista da interface pode alternar entre elas repetidas vezes.

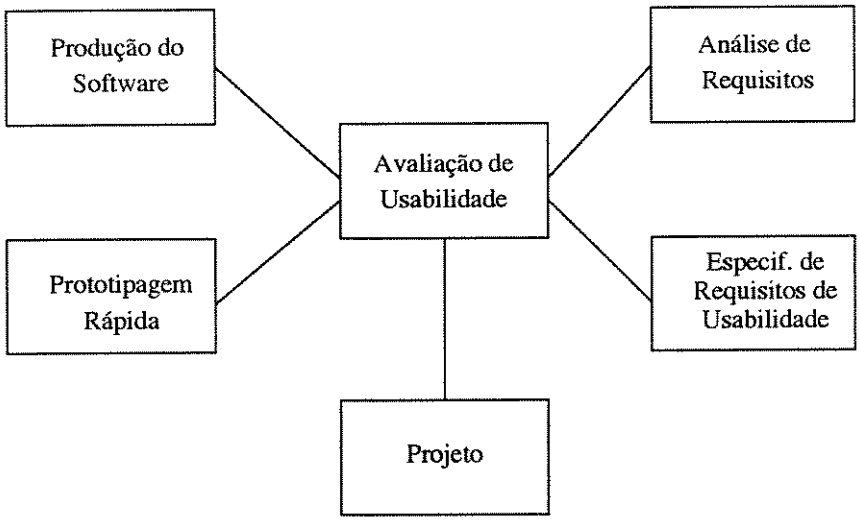

Figura 2.5: O Modelo Estrela [Har 89]

No modelo estrela, qualquer que seja a atividade em desenvolvimento, ela pode ser rapidamente substituída por outra e se reiniciar. Este modelo contempla especificamente o desenvolvimento da interface porque permite "pequenos ciclos de iteração", importantes para os testes com interfaces.

Neste modelo, o projeto e o desenvolvimento da interface com o usuário dura enquanto existirem motivação e orçamento para fazer modificações. Um mecanismo de controle deve ser usado pelo projetista para decidir quando parar no processo de refinamentos sucessivos. Este controle é baseado nas especificações quantitativas de usabilidade.

A avaliação formativa (ou de usabilidade) é usada para comparar a usabilidade atual do projeto de interação com as especificações de usabilidade à medida que o desenvolvimento prossegue. As várias atividades são altamente interconectadas através do processo de avaliação de usabilidade. Cada atividade é avaliada antes de se passar para a próxima atividade.

Diversas metodologias podem ser aplicadas nas diferentes fases de análise, projeto, representação de projeto e produção do software, como veremos na seção 2.3.

A seguir, apresentamos as diversas fases envolvidas no modelo estrela.

A Fase de Análise

Devido à nova ênfase em usabilidade, a fase de análise foi enriquecida com a participação efetiva do usuário. Com este enfoque, Hix & Hartson [Hix 93], definem uma série de atividades que devem fazer parte da tarefa de análise. São elas:

1. **Análise das necessidades**
   Estabelece se um novo sistema é de fato necessário, baseado nos objetivos da organização e na demanda do mercado. Isso determina os objetivos básicos, propósitos e características desejadas para o sistema de aplicação. As características do sistema são as capacidades do sistema que aparecem para os usuários. O resultado é uma visão externa do que o usuário será capaz de fazer com o sistema de aplicação. A informação recolhida durante esta etapa é especificada em um documento de conceitos do sistema [Pre 92];

2. **Análise do usuário**
   Combina a teoria cognitiva dos usuários, informações específicas sobre as funções e tarefas dos usuários potenciais, juntamente com considerações do fluxo de trabalho social e organizacional. Os fatores que predizem as diferenças em habilidades (dos usuários) baseadas em computador incluem experiência do usuário, aptidão particular, idade, domínio da área do problema, habilidades específicas, entre outras;

3. **Análise das tarefas**
   Provê uma descrição completa da tarefa, subtarefa e métodos envolvidos no uso do sistema, identificando os recursos necessários para que os usuários e o sistema executem as tarefas cooperativamente. A análise das tarefas envolve entendimento das seqüências requeridas, porque são requeridas, qual é o fluxo de informação, o que o usuário contribui para o procedimento, e o que pode ser automatizado;

4. **Análise funcional**
   Resulta em uma visão interna das funções a serem projetadas para a componente computacional do sistema (similar à análise de tarefas). Essas funções, quando combinadas com a interface de usuário, vão prover as características prometidas na análise das necessidades;

5. **Alocação de tarefa-função**
   Produz decisões sobre qual parte da tarefa será executada pelo usuário e qual será executada pelo sistema. Algumas tarefas podem ser manuais enquanto outras automáticas. Esta atividade permite determinar aquelas, dentro de um grupo de tarefas, que serão executadas manualmente e aquelas que serão automatizadas;

6. **Análise de requisitos**
   Utiliza-se da análise de necessidades, análise do usuário, análise das tarefas e análise funcional para estabelecer requisitos formais para o projeto.

Uma vez determinadas as necessidades e características que serão providas pelo sistema, inicia-se a fase de projeto. Contudo, a atividade de projeto da interface é muito diferente de projetar o software. Para projetar o software, o analista deve ter a visão do "computador" e, para projetar a interação, o analista deve focalizar o "comportamento do usuário" [Hix 94].

O Projeto da Interface

O objetivo principal da fase de projeto é atingir uma implementação estável. Para que isso aconteça, devemos alcançar dois objetivos: obter uma realização concreta do projeto em um protótipo que siga princípios estabelecidos de usabilidade, e verificar empiricamente o projeto com usuários reais para assegurar-nos de que o projeto vai ao encontro às necessidades dos usuários [Nie 92].

Isto nos leva a determinar alguns aspectos essenciais a serem considerados durante esta fase:

1. **Projeto participativo**
   Os usuários devem se envolver no processo de projeto através de encontros regulares com os projetistas. Os usuários às vezes levantam questões que a equipe de projeto nem sequer havia pensado [Nie 92];

2. **Projeto coordenado**
   Para cada projeto em desenvolvimento, deve haver uma autoridade central para coordenar os vários aspectos da interface a fim de tornar consistente[1] toda a interface [Nie 92];

3. ***Guidelines* e análise heurística**
   Os *guidelines* listam os princípios bem conhecidos do projeto de interfaces de usuário [Joh 95]. É possível executar avaliação heurística em termos de *guidelines*. O projeto da interface é examinado determinando se cada um de seus elementos segue cada um dos *guidelines* estabelecidos para o projeto. Mas, como certos *guidelines* conflitam entre si, um tratamento especial deve ser dado nesses casos;

4. **Conseqüência do projeto**
   As conseqüências das várias decisões de projeto devem ser tornadas explícitas e anotadas. Como freqüentemente se muda a interface, deve-se saber as razões que levaram ao novo projeto;

5. **Multi-disciplinaridade do projeto**
   Um projeto de sucesso requer coordenação entre diversas disciplinas. O aporte de engenheiros de fatores humanos, documentadores técnicos, especialistas em educação, projetistas gráficos e industriais, etc. resulta em um projeto de alta usabilidade;

6. **Projeto focado no usuário**
   A tarefa principal é aprender como o usuário pensa e trabalha, o que ele conhece e o que ele gosta. Antes do início do projeto, deve-se ter uma boa compreensão das perspectivas dos usuários [Pow 90]. Descobrir o que o usuário

[1] A consistência às vezes conflita com outras características desejáveis de usabilidade. Alguma flexibilidade é necessária para evitar impor um projeto mal feito aos usuários em nome da consistência.

quer é uma tarefa muito importante, embora muitas vezes não se cumpra[2]. As diferenças entre usuários e a variabilidade de tarefas são dois fatores que possuem maior impacto na usabilidade. O ambiente de trabalho e o contexto social do usuário são também muito importantes [Nie 92];

7. **Design interativo**
   Shneiderman [Shn 92] observou que são necessários métodos interativos de projeto que permitam testes em estágios iniciais do protótipo, revisões baseadas em *feedback* dos usuários e refinamentos incrementais sugeridos pelos administradores de teste para se alcançar um sistema com sucesso. A forma mais confiável de se produzir interfaces de qualidade é através de um processo que testa protótipos com os usuários e modifica o projeto baseando-se nos comentários recebidos [Mye 89].

Prototipação Rápida

O modelo estrela enfatiza a importância da prototipagem, identificando uma fase separada das demais. Através dos protótipos é possível detectar problemas de incompletude, ambigüidades e inconsistências no projeto, permitindo ao desenvolvedor resolver esses problemas enquanto existe flexibilidade para fazer mudanças. Embora a prototipação seja muitas vezes associada a ferramentas de software automáticas, é importante enfatizar que ela pode ser vista como uma técnica, não somente como uma ferramenta. Mais ainda, o trabalho de Gutierrez [Gut 89] identifica que o protótipo depende das características do problema. Assim, deve-se determinar em que ponto do desenvolvimento o protótipo é mais efetivo e que abordagem é a mais apropriada dependendo do propósito e do nível de conhecimento dos usuários.

Produção do Software

A interface fornece ao usuário uma máquina abstrata com a qual ele pode interagir [Luc 92]. A estruturação desta máquina abstrata pode corresponder a diversas abordagens, como por exemplo, uma hierarquia de máquinas abstratas com implementação de estilo cliente-servidor, ou a utilização do paradigma de orientação a objetos e até o paradigma de programação orientada a agentes [Sho 92].

Especificação de Requisitos de Usabilidade e Avaliação de Usabilidade

O modelo em estrela está centrado na avaliação formativa[3]. A avaliação de usabilidade é usada para comparar a usabilidade atual do projeto com as especificações

[2]Como Powel [Pow 90] afirma: "É difícil descartar a própria experiência e começar de novo, tentando ver o que o usuário vê. Muito mais difícil ainda é colocar de lado o próprio *background* e tentar aprender como o usuário pensa".

[3]Enquanto a interface está sendo construída, se "formando".

a fim de determinar problemas de usabilidade. Contudo, as mudanças para resolver estes problemas podem introduzir novos problemas. Esta é a razão para ter um projeto e avaliação iterativa [Nie 92]. Em [Hix 93] são apresentadas as diversas atividades envolvidas na especificação e avaliação de usabilidade para sistemas interativos. Consideramos que é necessário estabelecer especificações de usabilidade junto da análise das necessidades. Isto favorece ter uma primeira avaliação de usabilidade já na fase de análise e evitar que problemas se propaguem para o protótipo. Igualmente, a utilização de metodologias específicas para entender, avaliar, melhorar e representar o projeto de sistemas interativos permite a descoberta de inconsistências antes que a interface seja definitivamente implementada [Sil 95].

2.3 Metodologias para Desenvolvimento de Sistemas Interativos

Diversas metodologias têm sido propostas na literatura com o objetivo de modelar a interação homem-computador. Uma forma particular de modelar esta interação é conhecida como *Command Language Grammar* (CLG) [Mor 81].

CLG se concentra na geração de idéias de projeto e provê uma abordagem *top-down* para o projeto de interfaces de usuário. A maior vantagem do formalismo CLG é que ele permite que o projetista pense sobre a interface do usuário e os componentes conceituais do sistema. Como desvantagem podemos citar que representa pobremente os aspectos dinâmicos das interfaces e que não é de fácil entendimento por parte do usuário. CLG não tenta predizer o comportamento do usuário ou a quantidade de treinamento/aprendizagem requerida pelo usuário [Joh 92]; isto é uma grande diferença com relação à abordagem de *Task Action Grammar* (TAG) [Pay 86].

TAG descreve detalhadamente a estrutura da linguagem de comando da forma que ela é percebida pelo usuário. A intenção de TAG é prover um formalismo para modelar a representação mental de uma linguagem de interação, e permitir que uma especificação formal desta linguagem seja percebida pelo usuário. A visão de uma tarefa definida por TAG é, relativamente, de baixo nível. As descrições em TAG não enfatizam a representação do fluxo de controle durante a execução e, como em gramáticas no geral, não é facilmente observável.

De modo similar a CLG e TAG, *Task Action Language* (TAL) [Rei 81] utiliza uma gramática para avaliar (empregando métricas de usabilidade) o projeto da interface do usuário. TAL descreve as seqüências de pressionamento dos botões, movimentos do *mouse*, etc. TAL e GOMS[4] [Car 83] são voltados para a avaliação da usabilidade do projeto da interface, predizendo o comportamento do usuário sem ter que testar usuários reais em sistemas em produção [Joh 92].

GOMS faz uso do modelo do comportamento humano na forma de um modelo de processador humano (MPH). A intenção é que uma tarefa de usuário seja ana-

[4] *Goals, Operators, Methods and Selection rules.*

lisada e que o MPH seja usado para prover as predições de desempenho para o comportamento associado com as tarefas.

Uma extensão de GOMS é *Cognitive Complexity Theory* (CCT) [Kie 85]. Ela tem duas descrições paralelas, uma tipo GOMS que utiliza regras de produção para os objetivos dos usuários, e uma do tipo GTN (*Generalised Transition Networks*) para descrever a gramática do sistema. CCT prediz as dificuldades que uma pessoa vai encontrar no aprendizado e uso de um sistema interativo, modela o conhecimento que se presume que um usuário tenha para usar um sistema interativo particular, e tenta representar o modelo mental do usuário sobre o sistema. Contudo, não pode ser usado nem para ajudar o projetista a tomar decisões sobre a tela, os ícones, os botões, etc., nem pode ser usado para avaliar comportamento interativo muito detalhado.

Ao contrário de GOMS, *Interacting Cognitive Subsystems* (ICS) [Bar 87] modela o comportamento humano com um alto grau de sensibilidade. A análise começa fazendo suposições sobre quais processadores e subsistemas cognitivos do usuário seriam envolvidos na execução de qualquer parte da tarefa. ICS provê uma psicologia explícita das tarefas em termos dos processos psicológicos envolvidos. ICS subdivide o sistema humano de processamento de informação em um conjunto de sistemas especializados. ICS é mais difícil de ser aplicado do que GOMS ou CCT [Joh 92]. É de difícil aplicação para engenheiros/analistas, e mais apropriado para psicólogos.

O oposto da descrição estática de tarefas de GOMS e CCT é o *Programmable User Models* (PUM) [You 89], o modelo dinâmico dos planos dos usuários. Young tem considerado técnicas para analisar semi-automaticamente o comportamento do usuário usando técnicas de inteligência artificial (IA) baseadas em SOAR[5]. PUM prediz erros e permite uma avaliação da usabilidade de um projeto antes que ele seja efetivamente iniciado. O projetista especifica o conhecimento que o usuário necessita para executar uma dada tarefa. Baseado neste conhecimento, a arquitetura cognitiva de PUM tenta elaborar um plano. Se nenhum plano puder ser gerado, o projetista é notificado de um problema potencial de usabilidade e aprendizado [Cou 95]. Uma desvantagem de PUM é que não considera o modelamento e junção de tarefas dos usuários e do software [Joh 92] e é ainda um projeto de pesquisa.

User Action Notation (UAN) [Hix 93] permite a avaliação de usabilidade analisando a estrutura das tarefas e detectando possíveis ambigüidades e inconsistências no projeto. A abstração primária de UAN é a tarefa de usuário; contudo, a escolha da tarefa diz respeito ao sistema e não ao usuário. Uma interface é representada como uma estrutura quase hierárquica de tarefas assíncronas. Uma característica na especificação em UAN é que *feedback* na interface e informação de mudança de

[5]SOAR é uma arquitetura de solução de problemas realizada como uma série aninhada de espaços-problema. SOAR vem com um mecanismo embutido de aprendizado, chamado aprendizado por etapas, o qual é fortemente entrelaçado com o mecanismo de solução de problemas. SOAR pode ser considerada como uma máquina para aplicar conhecimento a situações a fim de gerar comportamento [You 89].

estado é provida no nível mais baixo. Em todos os níveis, as ações do usuário e as tarefas são combinadas com relações temporais tais como seqüenciamento, alternância e concorrência para descrever o comportamento do usuário, relacionado ao tempo. O problema principal é a falta de uma semântica clara e, em comum com outras técnicas analíticas, o fato de que a descrição de problemas grandes ser muito trabalhosa [Har 95].

2.3.1 Resumo das Metodologias

As diferentes metodologias proporcionam modelos que permitem que fatores que afetam o usuário e a tarefa sejam levados em consideração de modo mais explícito no ciclo de projeto.

GOMS, CLG e TAG focalizam, principalmente, a especificação de objetivos do usuário, e a tradução desses objetivos em ações, conforme os estágios de atividades do usuário envolvidos na execução de uma tarefa [Lau 86]. CCT, ICS e PUM suportam a análise detalhada da interação em termos de cenários e assume uma "psicologia de tarefas". Esses modelos não representam associação direta dos objetivos e ações com o *feedback* e os estados da interface como UAN. UAN provê um contraste interessante com essas notações. Ela não é uma especificação do conhecimento do usuário ou recurso. Ao invés disso, ela foi desenvolvida por projetistas de sistemas que se preocupavam com a complexidade da interação e menos com a complexidade do usuário [Har 95].

Contudo, estas abordagens possuem suas deficiências como visto anteriormente. Muitas delas não permitem documentar tarefas, funções, diálogo, classificação psicológica dos usuários e protótipos da interface numa mesma descrição. Muitas vezes umas metodologias completam a informação não existente em outras.

O ambiente proposto tem por objetivo dar apoio às atividades relacionadas à análise, projeto e prototipagem de sistemas interativos. Assim, pretende-se que os requisitos da aplicação interativa a ser desenvolvida sejam identificados e organizados em uma representação que incorpore e dê suporte a várias metodologias.

2.4 Ambientes de Desenvolvimento de Interfaces

Um ambiente de desenvolvimento de interfaces compreende um conjunto de ferramentas para o auxílio no projeto, construção e avaliação dos sistemas interativos. A construção de interfaces tem recebido uma atenção particular com o estudo da geração automática de interfaces.

Muitas pesquisas têm abordado a construção de interfaces com o intuito de reduzir os custos de desenvolvimento e melhorar a qualidade, fornecendo suporte ao ciclo de vida das interfaces. A próxima seção apresenta uma recapitulação das diferentes ferramentas que existem para o desenvolvimento de interfaces homem-computador.

2.4.1 Ferramentas para o Desenvolvimento da Interface

As ferramentas para projeto de interfaces têm recebido vários nomes ao longo dos anos, sendo UIMS (*User Interface Management Systems*) o mais utilizado.

Contudo, muitas pessoas acham que o termo UIMS deve ser usado apenas para ferramentas que manipulem o seqüenciamento das operações. Assim, usam-se outros termos tais como *toolkits*, *User Interface Development Environments*, *Interface Builders*, *Interface Development Tools*, e *Application Framework*.

Alguns autores diferem na definição de UIMS ([Hix 93];[Mye 95]). Utilizamos a definição de Myers que afirma que os UIMSs estão associados apenas à parte de execução da interface (ao invés da parte de projeto) ou a sistemas que apresentam um componente de controle de diálogo explícito.

Por outro lado, ferramentas denominadas de UIDE (*User Interface Design Environment*), são conjuntos integrados de ferramentas para a criação e gerenciamento das interfaces de usuário. Os UIDEs ajudam o projetista da interface a combinar as técnicas interativas na ordem certa. Myers usa o termo UIDE ao invés de UIMS porque ele vê o UIDE como provendo não só suporte de projeto como também as funções de gerenciamento em tempo de execução dos UIMSs mais antigos.

Na literatura existem diversos sistemas especialistas para geração automática de interfaces de usuário. Como por exemplo, GENIUS [Jan 93] que a partir de modelos entidade-relacionamento e redes de diálogo gera uma interface de usuário. Assim, GENIUS, através de diversos passos, produz uma descrição da interface de usuário a qual é transformada em especificação e interpretada pelo gerenciador de interface de usuário. No entanto, o tipo de abordagem de GENIUS é seqüencial e difere notavelmente do ciclo de vida em estrela, utilizada neste trabalho.

TRIDENT [Van 93] é um conjunto de ferramentas interativas que gera automaticamente uma interface de usuário para aplicações comerciais. Usando regras, um seletor determina automaticamente objetos abstratos de interação a partir das especificações, criando novas especificações chamadas de AIOs (*Abstract Interaction Object*). A interação concreta dos objetos produz uma interface observável. Como essas especificações são independentes do ambiente alvo, um mapeador transforma AIOs em objetos concretos de interação CIO (*Concret Interaction Object*) dependentes do ambiente. Esta arquitetura possui vários pontos interessantes e comuns aos outros trabalhos ([Jan 93];[Chu 93]). Sua arquitetura se apresenta em forma mais modular; algo mais próximo da abordagem utilizada neste trabalho. A desvantagem de utilizar uma ferramenta como esta é que ela só pode ser utilizada por especialistas em computação. Desenhistas gráficos, psicólogos e até usuários podem encontrar grande dificuldade em utilizá-la.

DON [Chu 93] é uma ferramenta que provê uma assistência especializada no projeto da apresentação de interfaces de usuário. Um modelo de bases de dados foi estendido para encapsular elementos, relações e princípios de conhecimentos de projeto relacionados com a organização e a apresentação dos menus e *dialog boxes*. É uma ferramenta adequada na organização da informação, na seleção apropriada dos objetos da interface e de seus atributos associados e no posicionamento dos

objetos selecionados em *dialog boxes* ou em um menu, de forma lógica, consistente e significativa. A separação entre a organização e a apresentação é similar à separação definida neste trabalho[6].

Myers [Mye 90] desenvolveu um UIDE muito sofisticado, chamado de GARNET. A arquitetura de GARNET inclui duas classes principais de ferramentas; ferramentas de alto-nível para construção rápida e fácil de interfaces gráficas e diálogos, e ferramentas de baixo nível que traduzem os projetos de alto nível em algo que possa ser executado em algum sistema operacional. No entanto, GARNET não possui esse caráter multi-disciplinar, ele é apresentado como um ambiente no qual deve-se utilizar as próprias ferramentas de GARNET (tais como *Jade dialogue box creation tool* ou *Lapidary-graphical tool*) e tem que ser utilizado por especialistas em computação. Isto difere notavelmente da abordagem que é tratada nesta proposta. Contudo, é possível estabelecer como GARNET poderia ser incorporada e ela própria representar um agente dentro do sistema multi-agentes.

O modelo PAC (*Presentation Abstraction Control*) [Cou 91] foi uma base para o projeto de outros UIDEs [Joh 92]. Neste modelo, os agentes se organizam em triplas de apresentação, abstração e controle. Assim, o todo de um sistema interativo pode ser construído recursivamente como um conjunto de agentes PAC. PAC provê a habilidade de alternar entre as interações através da parte de controle de cada agente, retendo o estado da interação a um nível local. Isto dá algum grau de entrelaçamento das interações, embora sem concorrência.

Um problema de PAC é que este sugere uma organização hierárquica mas não especifica o papel dos agentes intermediários na hierarquia. Contudo, o modelo PAC é um bom exemplo da aplicação da abordagem multi-agentes.

2.4.2 Resumo dos UIDEs Selecionados

Existe uma variedade muito grande de UIDEs correspondentes a diversas filosofias. Contudo, as ferramentas ainda precisam ser melhoradas. Elas não são fáceis de usar por não-programadores e as especificações geradas apresentam restrições de diversos tipos. A dificuldade principal que encontramos é que não suportam aspectos do processo de projeto tal como análise dos usuários e modelamento de cenários. Estes ambientes não consideram o processo de desenvolvimento como uma atividade multi-disciplinar. Conseqüentemente, eles estão longe de serem ambientes completos de projeto.

2.5 Conclusão

Este capítulo introduziu a noção de que o desenvolvimento de sistemas interativos está formado de duas atividades principais: uma pertencente ao software e outra pertencente à interface com o usuário. A comparação destas atividades reforça

[6]Para maiores informações ver agente de interface e agente de prototipagem no capítulo 4, nas seções 4.5 e 4.7.

a necessidade de uma abordagem para o desenvolvimento da interação que seja diferente das abordagens tradicionais da engenharia de software.

Diversas metodologias têm sido propostas para modelar os diferentes aspectos da interface homem-computador. Estes modelos diferem em seus propósitos ou usos no processo de projeto da interface. Cada abordagem tem considerado o usuário, as tarefas e a interface. Estes três elementos são fundamentais no desenvolvimento de uma boa interface. Atualmente têm surgido diversas ferramentas que tentam garantir a atenção a estes três elementos. Isto nos leva a destacar a necessidade de contar com ferramentas multi-disciplinares, facilitando assim que, durante o desenvolvimento, sejam coordenadas todas aquelas atividades que permitem definir a visão do sistema e aquelas que permitem definir a visão do usuário.

Este trabalho estabelece uma base conceitual onde diversos especialistas trabalham de maneira coordenada no desenvolvimento do projeto da interface. Esta base conceitual é definida segundo a abordagem multi-agentes, que será apresentada no próximo capítulo.

3

Visão Geral sobre os Sistemas Multi-Agentes

A abordagem para resolução de problemas envolvendo inteligência artificial é baseada no modelamento da inteligência humana (metáfora da inteligência). Neste contexto, muito se tem produzido sobre representações de conhecimento e regras de inferência, componentes essenciais da metáfora da inteligência.

Contudo, a complexidade dos problemas que se têm tentado resolver com tal abordagem aumentou tanto, que esta se mostrou inadequada na maioria dos casos. Para solucionar esses problemas, pensou-se em utilizar os esforços de entidades que cooperam entre si. Daí, esta abordagem ser conhecida genericamente por inteligência artificial distribuída (IAD). Em IAD a metáfora utilizada é a de comportamento social, com ênfase em ações e interações.

Neste escopo, uma das grandes linhas de pesquisa é a dos Sistemas Multi-Agentes (*Multi-Agent System*). Estes sistemas possuem características próprias como paralelismo, inteligência, e aprendizagem. Além disso, permitem a solução distribuída de problemas. Este capítulo apresenta os sistemas multi-agentes, suas características e vantagens de implementação.

3.1 Agentes e Sociedade de Agentes

Atualmente, tem havido um grande interesse em agentes surgindo, assim, diferentes conceitos. Entre eles, podemos citar os agentes convincentes (*Believable agents*) [Bat 94] que mostram o convencimento, a persistência e as ilusões da vida

em animações, ambientes educacionais, etc. Ou, por exemplo, os agentes *softbots* (*software robot*) [Etz 94] que provêem uma interface integrada e expressiva para a Internet e auxiliam as pessoas a usar recursos computacionais. Mas, de forma geral, os agentes são por definição entidades que atuam em um ambiente, de acordo com seus próprios objetivos e de acordo com seu estado corrente de conhecimento [Boi 92].

Os agentes correspondem à metáfora do organismo vivo quanto à sua acessibilidade cognitiva e ao seu estilo de comunicação [Lau 90].

Para descrever cada agente utilizam-se descrições chamadas, respectivamente, de descrições internas e externas [Ber 92][1]. A descrição externa é a representação compacta das capacidades de cada agente. Este tipo de descrição refere-se ao comportamento cooperativo e distribuído dos agentes. A descrição interna mostra como o agente se comporta internamente. Este tipo de descrição pode ser explicitamente representada ou não. No caso em que seja explicitamente representada, permite ao agente raciocinar sobre seu próprio comportamento interno. Dois agentes podem ter a mesma descrição externa, mesmo que internamente sejam organizados de forma diferente.

Por sua vez, um agente pode ser dividido em duas partes ([Ber 92];[Boi 92]):

1. **A parte estática**
   Conhecida por representação do conhecimento, é a parte do agente que manipula as definições dos tipos de conhecimento disponíveis ao agente na busca da solução e determina qual conhecimento é relevante para a existência do agente na sociedade (planos a serem executados, escolhas disponíveis, representações explícitas e abstratas dos problemas que o agente tem que resolver, etc.);

2. **A parte de processamento dinâmico**
   Conhecida por métodos, é a parte do processamento que ocorre dentro do agente. Os métodos produzem as atividades de raciocínio do agente, e podem ser divididos em: capacidades de decisão, incluindo mecanismos de escolhas e tomada de decisões e capacidades de raciocínio, tais como planejamento de comunicação, raciocínio sobre o conhecimento e o comportamento dos outros agentes, etc.

O efeito da ação de um agente é percebido pela produção de eventos que modificam o ambiente e isto depende do tipo do agente. Existem dois tipos de agentes: agentes reativos e agentes cognitivos.

3.1.1 Agentes Reativos

Os agentes reativos são baseados em modelos de organização biológica. A idéia fundamental é que um comportamento inteligente e complexo pode emergir de uma

[1] Nossa arquitetura de agentes possui também descrições deste tipo. Para maiores informações ver seção 4.11.

coleção de entidades muito simples.

Os agentes reativos possuem um comportamento estímulo-resposta [Sic 92]. Um sistema estímulo-resposta é um conjunto organizado de agentes capazes de reagir a fenômenos externos determinados (estímulos) e de produzir por sua vez outros estímulos [Cou 91]. Desta forma, agentes reativos não mantêm informações à respeito das atitudes tomadas no passado, e não fazem nenhum tipo de previsão sobre as ações a serem tomadas no futuro. Tudo o que eles sabem a respeito das ações e do comportamento dos outros membros da sociedade é percebido por mudanças no ambiente. Assim, não existe modelo de alto nível para comunicação, nem uma representação explícita do ambiente ou dos membros da sociedade. Uma sociedade de agentes reativos tem usualmente um grande número de membros.

3.1.2 Agentes Cognitivos

Os agentes cognitivos são baseados em modelos de organização social humana (grupos, hierarquias, etc) [Sic 92]. Em cada agente, existe uma representação explícita do ambiente e dos membros da sociedade. Eles podem raciocinar a respeito das ações tomadas no passado e planejar as que serão tomadas no futuro.

As seguintes características devem ser adicionadas a um sistema baseado em conhecimentos, a fim de que se possa chamá-lo um agente cognitivo [Sic 92]:

- **Capacidade de percepção**
  Um agente deve perceber seu ambiente, detectando mudanças nele;
- **Capacidade de comunicação**
  Os agentes devem ser capazes de se comunicar uns com os outros; o termo comunicação significa que os agentes podem trocar não apenas dados, como também partes de conhecimentos e planos parciais;
- **Capacidade de ação**
  Um agente deve ser capaz de atuar em seu ambiente como resultado de suas atividades, visando a resolução do problema;
- **Raciocínio social**
  Um agente deve ser capaz de raciocinar sobre as atividades dos outros agentes via representação interna dos outros membros da sociedade e de seus respectivos mecanismos de raciocínio;
- **Estrutura de controle multi-camada**
  Um agente deve decidir quando deve usar suas habilidades de percepção, se comunicar, planejar e agir.

Os agentes cognitivos por sua própria definição possuem uma arquitetura mais interessante para desenvolver sistemas multi-agentes[2]. Este tipo de agentes é adotado pela maioria dos ambientes e é o utilizado neste trabalho.

[2]Cada referência ao termo agente no texto, implica no fato de ser um agente cognitivo.

Bind citado por Edmonds at. al. [Edm 94] notou que os agentes cognitivos podem ser tanto homogêneos como heterogêneos. Bind considera heterogêneos aqueles agentes que possuam diferenças sintáticas, de controle e semânticas. Caso contrário, ele os considera homogêneos. As diferenças sintáticas ocorrem devido ao uso de diferentes formalismos de representação do conhecimento. Já, as diferenças de controle ocorrem quando diferentes estratégias de inferência são usadas para processar o mesmo conhecimento. Finalmente, as diferenças semânticas ocorrem quando o conhecimento puder ter significados distintos para diferentes agentes.

3.1.3 Sociedade de Agentes

Uma sociedade de agentes, também conhecida por sistema multi-agentes, é uma coleção de agentes, normalmente organizada hierarquicamente, que tem seus objetivos específicos bem definidos, e que trabalha em prol de um objetivo global. Isto determina que existam elos de comunicação entre os agentes. Interações básicas tratam do que é trocado usando esses elos: conhecimento, objetivos, planos ou escolhas [Boi 92].

A sociedade de agentes pode ser definida como tendo dois aspectos: um estático e outro dinâmico ([Ber 92];[Boi 92]). O aspecto estático define o que pode ser feito em sociedade (regras do jogo). O aspecto dinâmico define a forma e o momento da comunicação entre os agentes, levando em consideração as regras predefinidas.

A comunicação entre os agentes pode ser direta ou indireta. Inicialmente, os agentes devem se conhecer e estabelecer comunicação direta. Em seguida, cada agente envia dados ao ambiente e recebe dados do ambiente, de acordo com mecanismos predefinidos estabelecendo uma comunicação indireta.

A comunicação é o principal instrumento dos agentes para conseguir coordenar suas ações [Hüb 95]. A abordagem multi-agentes considera que uma vez que o problema seja apresentado, a sociedade de agentes interage a fim de decompor o problema em subproblemas mais fáceis de se resolver. Assim, é necessário habilitar os agentes para comunicação e interação através de uma linguagem de comunicação e protocolo.

Comunicação, comportamento e controle dos agentes são características fundamentais no trabalho da sociedade e serão analisados na próxima seção.

3.2 Características da Sociedade de Agentes

A engenharia de software baseada em agentes tem sido comparada com o enfoque orientado a objetos. A principal diferença entre as duas abordagens está na linguagem de comunicação entre os agentes. Em geral, no enfoque orientado a objetos, o significado da mensagem pode variar entre objetos. Em engenharia de software baseada em agentes, usa-se uma linguagem comum com uma semântica independente do agente [Gen 94].

Os agentes podem adquirir conhecimento sobre o ambiente e sobre os outros através de sua própria percepção e ações de comunicação [Ber 92]. Percepção significa uma capacidade intrínseca de um agente em relação ao seu ambiente. Comunicação significa troca de dados e de conhecimento entre os agentes.

3.2.1 Comunicação entre Agentes

Como essa comunicação é muito complexa, protocolos de comunicação precisam ser definidos a fim de expressar os conceitos semânticos envolvidos numa solução cooperativa de problemas. Chang e Woo [Ber 92], no seu protocolo SANP[3], consideraram todas as situações de conflito que poderiam ocorrer em um sistema onde o controle é distribuído e respostas globalmente exatas não existem. Seis estágios complexos foram identificados a fim de se construir o protocolo:

1. A situação inicial onde os agentes verificam se possuem um entendimento comum sobre o tópico e se possuem conhecimento suficiente para que possam conversar;
2. O agente ***A*** faz uma proposição e recebe uma concordância ou uma discordância do agente ***B***;
3. Se o agente ***B*** discordou, o agente ***A*** fornece mais argumentos ao agente ***B***;
4. Um estágio tático é iniciado se o agente ***B*** continuar discordando; esse estágio consiste em várias trocas de mensagem onde cada parte tenta defender sua posição, usando táticas diferentes;
5. As duas partes começam a criar um compromisso;
6. Uma proposição final é decidida e a negociação termina.

As trocas podem ser muito complexas no sentido em que um terceiro agente pode intervir a fim de arbitrar a discussão. Intencionalmente, a comunicação é quase totalmente codificada ao nível do protocolo de comunicação. De fato, a linguagem comum que é usada entre os agentes é construída a partir do conteúdo das trocas, ao qual é adicionado o diálogo.

3.2.2 Comportamento Social entre Agentes

Podemos classificar o comportamento dos agentes cognitivos segundo dois critérios [Sic 92]:

1. **Localidade da tarefa**
   Local (interessa unicamente a apenas um agente) ou global;

[3]O protocolo SANP é fundamentado por modelos obtidos do estudo lingüístico das situações comuns nas sociedades humanas.

2. **Capacidade de executar a tarefa**
   Capaz ou incapaz.

De acordo com esses dois critérios, os seguintes comportamentos podem ocorrer:

- **Cooperação**
  Um agente não pode solucionar seu problema e pede a outros que cooperem com ele;

- **Cohabitação**
  Um agente pode solucionar seu próprio problema e o soluciona; um agente não é obrigado a cooperar com os outros simplesmente por estar no mesmo ambiente;

- **Colaboração**
  Um agente pode solucionar uma tarefa global sozinho; se existem muitos agentes que podem solucionar sozinhos uma tarefa global, usa-se um mecanismo de eleição para se determinar quem vai fazê-lo;

- **Distribuição**
  Alguns objetivos globais devem ser executados via uma ação coletiva; nenhum dos agentes pode executar a tarefa global sozinho, assim a tarefa tem que ser dividida e alocada de acordo com algum critério.

Cabe destacar que existem trabalhos onde o comportamento dos agentes pode mudar segundo seu *personality traits* [Mae 94]. Esta característica facilita ter sistemas inteligentes capazes de alterar seu comportamento segundo o estado do ambiente. Os *personality traits* se aplicam ao processo de definir quando atuar em uma solução, conhecer se a ação do agente é apropriada para certa situação e que ação é recomendada para uma situação. Carbonell [Car 80] analisa *personality traits* em termos das árvores de objetivos e predisposições em aplicar certas classes de estratégias de planejamento.

3.2.3 O Controle

Existem dois níveis diferentes de controle: o controle do agente e o controle da sociedade. O primeiro se ocupa de como os agentes devem organizar suas atividades internamente. Por outro lado, o controle da sociedade se ocupa de como organizar o conjunto de agentes e suas interações [Sic 92].

O controle da sociedade pode ser centralizado ou descentralizado. O controle centralizado significa que alguém da sociedade estabelece os objetivos que os agentes devem executar. O controle descentralizado significa que cada membro da sociedade pode participar no estabelecimento do objetivo que os agentes devem executar.

O controle de um agente pode também ser centralizado ou descentralizado. No controle descentralizado, a organização das atividades de cada agente tem como objetivo principal a busca da solução de um problema. Para isto, possuem características especiais na sua arquitetura, tais como ([Sic 92];[Woo 95]):

1. A decomposição da tarefa é feita pelos agentes e não pelo projetista ou pelo usuário; pode acontecer que os agentes decidam mudar seu comportamento a fim de melhor executar seu trabalho;

2. Os agentes são autônomos[4], possuem seus próprios objetivos; neste caso, conflitos freqüentemente aparecem devido aos objetivos locais e globais;

3. Os agentes têm capacidade de interagir com outros agentes dentro da sociedade;

4. Sistemas multi-agentes são sistemas abertos, qualquer agente pode entrar ou sair da sociedade quando desejar; se um novo agente entrar na sociedade, os demais agentes devem incorporar informações sobre suas habilidades e métodos, mantendo-se em cada agente uma representação explícita dos métodos e dos objetivos dos outros agentes;

5. Os agentes têm capacidade de perceber e reagir ao ambiente;

6. O ambiente pode mudar. Os agentes devem incorporar essas alterações em seu modelo do ambiente.

Estas características (comunicação, comportamento e controle) dos agentes e sociedade permitem visualizar claramente as diferenças entre esta abordagem, a tradicional de solução de problemas e a abordagem de sistemas especialistas.

3.3 Visões e Realizações de Sistemas Baseados em Agentes

O estudo dos agentes de software tem resultado em diversas visões e realizações de sistemas. Uma delas focaliza-se na construção de um agente especializado que possa assistir ao usuário através da execução de alguma tarefa específica para ele. Como vemos na Figura 3.1, o agente assiste e coopera de forma pessoal com o usuário na realização de uma tarefa [Rie 94]. Neste caso, dizemos que o agente é autônomo se for capaz de ter flexibilidade e adaptabilidade em estabelecer seus próprios objetivos, de acordo com seus interesses, e se for capaz de executá-los de forma eficiente.

Uma segunda área de estudo ocupa-se da integração de um conjunto de agentes especializados (sistemas multi-agentes) que cooperam entre si com a finalidade de

[4]Agentes autônomos são agentes capazes de operar sem a intervenção constante de outros e de ter controle sobre suas metas e estado interno [Woo 95].

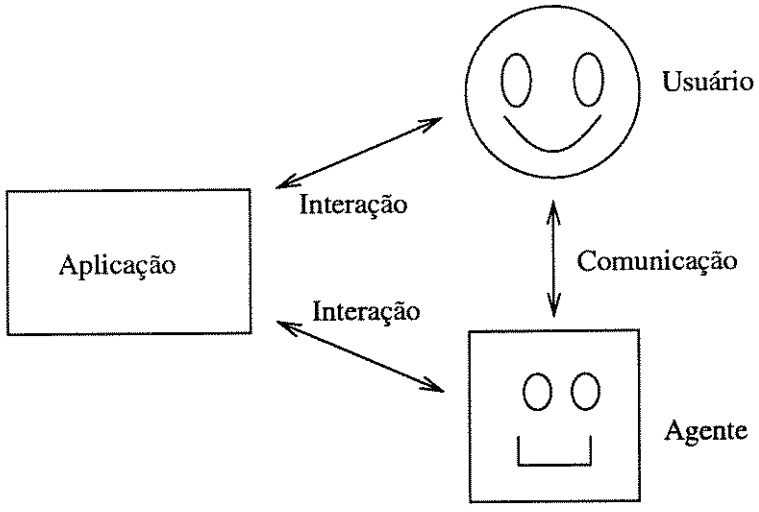

Figura 3.1: Agente Autônomo [Mae 94]

atingir um objetivo global. Neste caso, como vemos na Figura 3.2, os agentes trabalham em conjunto para solucionar um problema. Nessa cooperação, o usuário participa na busca da solução. Os agentes por sua vez adaptam seu comportamento de acordo ao *feedback* do usuário. Isto faz parte da aprendizagem dos agentes com o objetivo de aprimorar a busca da solução.

No caso específico da área de interfaces de usuário, pode-se usar a visão de agentes autônomos para implementar um estilo complementar de interação (*indirect management* [Mae 94]), onde o usuário é colocado em um processo cooperativo no qual ele próprio, junto com o agente, iniciam a comunicação, monitoram eventos e realizam tarefas. A metáfora utilizada é a de um assistente pessoal, aquele que colabora com o usuário em algum ambiente de trabalho, como por exemplo agentes que assistem ao usuário na utilização de alguma outra aplicação, agentes de *help* sensíveis ao contexto, etc. Tais assistentes, quando inseridos em ambientes de trabalho que usam técnicas de inteligência artificial, têm aumentado o potencial dos ambientes devido à organização do trabalho [Cro 88].

Não obstante, nesta área, existem controvérsias sobre o uso de agentes. Um agente é pior ou melhor de acordo com sua habilidade de responder ao usuário. Alguns agentes possuem responsividade explícita, isto é, o usuário e o sistema se comunicam através de uma série de transações explícitas, altamente restritas. A responsividade requer que o agente tenha acesso ao modelo dinâmico do usuário, ou pelo menos, anote sua experiência em uma aplicação particular com regras para interpretar aquela experiência quando formula ações com o objetivo de apresentar um comportamento inteligente [Lau 90].

A segunda abordagem, consiste em produzir agentes que apóiem um conjunto relevante de funções para um processo quando considerado de várias perspectivas. Algo mais próximo da Figura 3.2. Neste contexto, é importante notar que um sistema multi-agentes pode ser muito complexo [Edm 94].

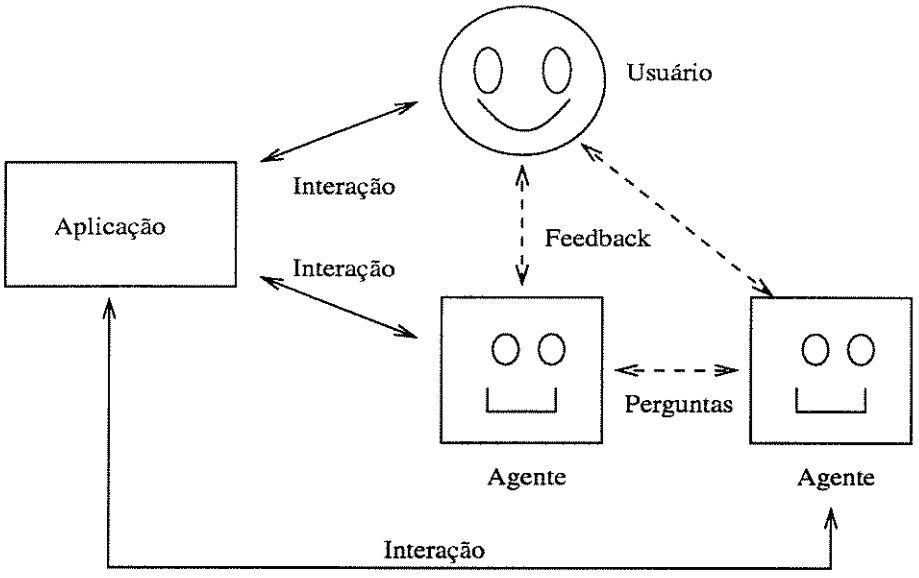

Figura 3.2: Sociedade de Agentes [Mae 94]

O sentido de se pesquisar sistemas multi-agentes está em mostrar o quanto estes sistemas podem tornar os computadores muito mais úteis do que atualmente para uma diversidade de usuários. Conseguir isto depende também do elemento humano e da confiança que os usuários são capazes de depositar nos agentes [Nor 94].

3.4 Conclusão

Atualmente tem havido um grande interesse em sistemas multi-agentes. Fundamentalmente porque o modelo multi-agente substitui o servidor seqüencial da abordagem comum por uma organização de atores cooperando em paralelo. As arquiteturas propostas para sociedades de agentes são úteis porque provêem um arcabouço conceitual e definem as características gerais dos agentes.

No caso específico da área de interfaces, duas abordagens parecem promissoras. Uma, onde os agentes são capazes de realizar tarefas para seus usuários, como agendar compromissos, mandar mensagens via e-mail, etc., e outra, onde os agentes apóiam o usuário em uma série de funções trabalhando conjuntamente na resolução de tarefas. Em ambas as abordagens, o paralelismo e a distribuição de tarefas são condições necessárias para se levar em consideração os métodos de resolução de problemas aplicados pelos usuários. A repartição permite uma grande modularidade e a abertura a processos psicologicamente distribuídos.

O problema que se apresenta é justamente o da identificação dos agentes que farão parte do ambiente.

4

Ambiente Multi-Agentes para Projeto de Interfaces

Baseados nos modelos multi-agentes descritos no capítulo anterior, este capítulo apresenta uma proposta de aproveitamento destes sistemas para assistir o projetista de interfaces homem-computador em sua tarefa. O objetivo deste capítulo é descrever os agentes básicos constituintes do ambiente e propor uma arquitetura interna para os agentes, que será refinada no próximo capítulo.

4.1 Requisitos para o Ambiente Multi-Agentes para Projeto de Interfaces

Como visto anteriormente, o modelo multi-agentes consiste de uma organização de atores cooperando em paralelo. Os agentes têm grande modularidade. É, portanto, possível modificar localmente o comportamento de um agente sem a necessidade de alterar o funcionamento do conjunto.

O agente define a unidade de execução. Ele pode ser executado sob um computador fisicamente diferente do lugar de ativação. Esta é uma característica essencial nos novos ambientes de desenvolvimento. Outra característica dos agentes é que estão compostos de conhecimento, que dirigem suas funções. Este conhecimento é descrito na forma de desejos e processos que definem os objetivos para o agente e a forma na qual ele executará suas tarefas [Kin 95].

Estas características levaram à identificação de algumas questões fundamentais

a serem tratadas por este trabalho. A saber [Ari 97a]:

1. **Explorar o projeto colaborativo entre os projetistas**
   Projeto colaborativo é uma atividade de grupo, complexa, envolvendo participantes com habilidades heterogêneas. Os agentes são viáveis para tarefas de projeto [Edm 94]. O objetivo é produzir agentes que apóiem um conjunto relevante de tarefas;

2. **Permitir o caráter multi-disciplinar**
   A competência multi-disciplinar é uma das exigências do domínio de interface homem-computador. Cada um dos agentes pode ter funcionalidades diferentes, trabalhando em conjunto na solução de um problema;

3. **Documentar o processo de desenvolvimento do projeto**
   Documentar o processo do projeto consiste em registrar o desenvolvimento, indicando quais agentes participam da solução do problema, o tempo de desenvolvimento da solução e as decisões de projeto que conduziram às novas versões;

4. **Determinar um arcabouço conceitual para o projeto**
   Definir um *framework* que proveja uma forma de pensar e modelar a estrutura da interface de forma a permitir uma maior integração entre as diversas atividades realizadas no ambiente;

5. **Organizar os componentes de um projeto de interface**
   Definir uma arquitetura adequada que permita a identificação dos componentes abstratos da interface, do controle do diálogo, do estilo de apresentação e das especificações de projeto;

6. **Permitir alternativas de projeto**
   O ambiente multi-agentes pode apresentar exemplos de outros projetos, de *guidelines* de usabilidade, estilos de organização da informação, etc., que auxiliem o projetista no processo de especificação.

Um ambiente inspirado nestes requisitos é proposto e apresentado na próxima seção.

4.2 Descrição do Ambiente Proposto

A arquitetura do ambiente é uma coleção de agentes autônomos e independentes com uma relação de tipo "paternal" com um único coordenador que chamaremos aqui de agente *kernel*. O agente *kernel* tem tarefas específicas como inicialização dos agentes, inclusão dos agentes na sociedade, entre outras.

O ambiente pode conter qualquer número de agentes e diferentes funcionalidades. O objetivo é ter agentes especializados que auxiliem diferentes usuários do

ambiente de acordo com sua área de atuação. Mais ainda, que permitam que profissionais com perfis tão diferentes como engenheiros de fatores humanos, documentadores técnicos, especialistas em educação, representantes de marketing, projetistas gráficos e industriais, trabalhem num mesmo projeto tendo em consideração um aspecto específico do projeto.

Para conseguir que várias pessoas trabalhem com vários agentes, determinamos que o projeto da interface seja tratado também como uma sociedade formada por múltiplos agentes. Assim, o arcabouço conceitual para o projeto é uma sociedade multi-agentes, onde o projeto da interface está composto por agentes que chamaremos de **agentes da interface da aplicação** (AIA)[1]. Desta forma, os agentes do ambiente trabalham junto aos diferentes usuários para definir um projeto de interface formado pelos AIAs. No ambiente, o projeto que está sendo desenvolvido pelos agentes é chamado de **idéia**. A idéia contém todas as informações relativas à especificação da interface da aplicação, como veremos no capítulo 6.

Um ambiente assim definido pode atingir grandes dimensões. Definir todos os agentes constituintes do ambiente não é a motivação principal deste trabalho, principalmente porque ferramentas existentes atualmente para projeto de interfaces, como por exemplo TRIDENT [Van 93] ou DON [Chu 93], poderiam ser transformadas em agentes utilizando a ferramenta UPShell ([Oli 93];[Oli 93a];[Oli 93b]), a qual permite transformar um sistema baseado em conhecimento em um agente. Assim, ferramentas baseadas em conhecimento podem passar a fazer parte do ambiente.

Devido a essa facilidade, focalizamos o nosso maior interesse em definir os agentes necessários para o funcionamento do ambiente de apoio à definição do projeto da interface, deixando claro para os trabalhos futuros a tarefa de incorporar novos agentes ao ambiente na forma de ferramentas específicas.

Baseados em diferentes trabalhos ([Hix 94];[Chu 93];[Sch 94];[Van 93]) e visando atingir o objetivo afirmado anteriormente, definimos os seguintes agentes, mostrados na Figura 4.1[Ari 96]:

1. **Agente especialista em análise de sistemas**
   Recebe do projetista a descrição da análise e gera uma análise de requisitos;

2. **Agente especialista na gramática do ambiente**
   Proporciona para os demais agentes uma tradução de alguma linguagem fonte para a gramática do ambiente;

3. **Agente de interfaces**
   Identifica os **agentes da interface da aplicação** que irão compor a interface;

4. **Agente da interação da interface**
   Identifica o controle do diálogo da interface;

5. **Agente de prototipagem**
   Define um primeiro cenário da apresentação e mostra de maneira gráfica a aparência do projeto da aplicação;

[1]Os agentes do ambiente gerador do projeto chamaremos simplesmente agentes.

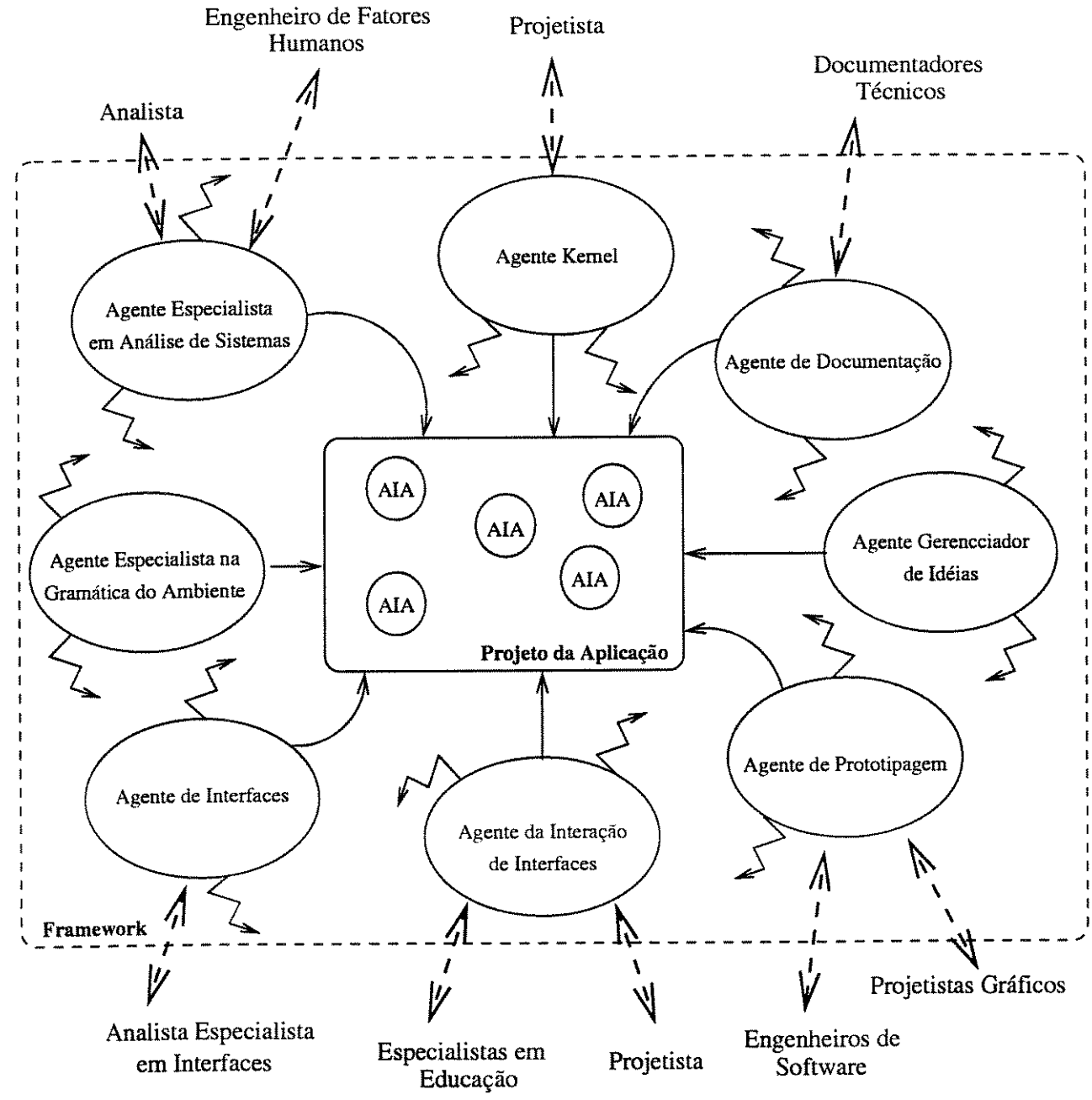

Figura 4.1: Agentes do Framework

6. **Agente gerenciador de idéias**
   Facilita a reutilização, consulta e modificação da **idéia** por parte dos outros agentes;

7. **Agente de documentação**
   Apresenta a documentação formal do projeto da interface;

8. **Agente *kernel***
   Gerencia o trabalho dentro do ambiente de agentes.

A interação entre os agentes é bem mais complexa do que apresentada na Figura 4.1. Eles podem se comunicar entre si consultando dúvidas, transmitindo resultados, etc. O tipo de relação que estabelecem depende da funcionalidade de cada agente, como veremos na próxima seção.

4.3 Agente Especialista na Gramática do Ambiente

Uma característica desejável dos agentes é terem uma linguagem expressiva que permita trocar dados ou informação lógica, comandos individuais e *scripts* [Gen 94]. Como todos os agentes se comunicam com a mesma linguagem, a uniformidade é uma vantagem do ambiente; isto permite também eliminar as inconsistências e variações arbitrárias de notação entre eles. Assim, a linguagem[2] própria do ambiente seria utilizada pelos agentes para:

Figura 4.2: Agente Especialista na Gramática do Ambiente

- Comunicar-se entre si;
- Documentar seu trabalho, permitindo proporcionar explicações aos outros agentes sobre seu comportamento.

[2]Existem duas abordagens para o projeto da linguagem. Uma procedimental, baseada no conceito de que a comunicação pode ser bem modelada como o intercâmbio de diretivas procedimentais (ex. TCL, etc.), e outra declarativa, baseada no conceito de que a comunicação deve ser projetada como o intercâmbio de sentenças declarativas (definições, suposições, e outras). A linguagem declarativa deve ser suficientemente expressiva para comunicar uma ampla gama de informação incluindo procedimentos. Esta última é a sugerida neste trabalho.

O objetivo de existir um agente especializado neste tipo de função dentro do ambiente é justamente para facilitar o trabalho dos demais agentes do ambiente que não necessitam conhecer a gramática (Figura 4.2). Outra vantagem é na incorporação de novos agentes ao ambiente. Neste caso, um pequeno tradutor seria associado ao novo agente de tal forma que possa trabalhar sem necessidade de ser reescrito.

Determinar uma linguagem que possa conter todo o significado das diferentes especificações na sua semântica não é tarefa fácil, principalmente com a existência de tradutores de uma linguagem para outra. Problemas como perda de informação na tradução são riscos que mostram a necessidade de um estudo delicado na definição da gramática do ambiente. Este agente não será detalhado neste trabalho.

4.4 Agente Especialista em Análise de Sistemas

O agente especialista em análise de sistemas é o agente encarregado de obter a especificação da interface que passará a constituir parte da idéia (Figura 4.3).

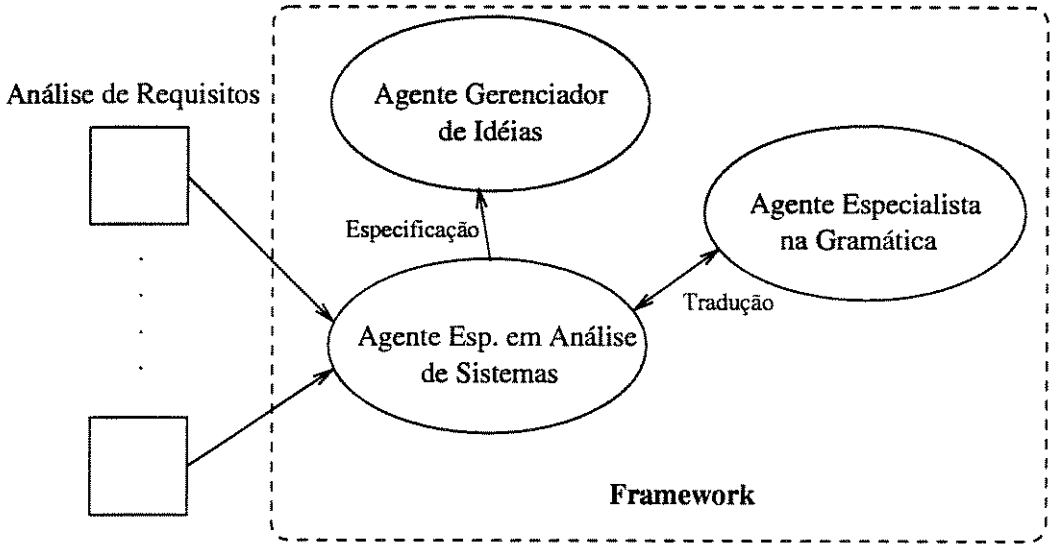

Figura 4.3: Relação do Agente Especialista em Análise de Sistemas com Outros Agentes

Para realizar a especificação, o agente recebe do projetista diversas informações que propõem-se como sendo aquelas descritas em [Hix 93][3]:

1. **Análise das necessidades**;
2. **Análise do usuário**;

[3]Uma descrição mais detalhada se encontra no capítulo 2, seção 2.2.2.

3. **Análise das tarefas**;

4. **Análise funcional**;

5. **Alocação de tarefa-função.**

No capítulo 7 é apresentada com um exemplo a descrição criada pelo agente especialista em análise de sistemas contendo este tipo de informações.

4.5 Agente de Interfaces

Uma vez concluída a fase de análise, esta provê os requisitos para a fase seguinte, a de projeto. Esta fase é certamente a atividade mais criativa e individualizada do ciclo de vida do software [Hix 93].

O projeto conceitual é de alto nível e diz respeito à síntese dos objetos e operações. Nele se descrevem os AIAs com seus atributos e funcionalidade, provenientes diretamente das tarefas do usuário. A identificação dos AIAs parte da decomposição de alto nível das tarefas da aplicação interativa e faz parte da descrição da idéia como vemos na Figura 4.4.

O AIA assim identificado executará as funções previamente determinadas na alocação de tarefa-função e terá um comportamento adequado a sua forma de ser (funcionalidade) dependendo de com quem ele trabalhe, usuário e outros AIAs.

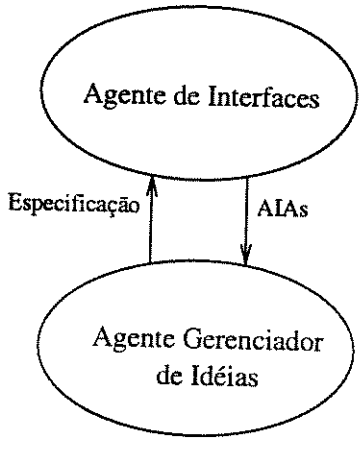

Figura 4.4: Agente de Interfaces

Propõe-se que para organizar o trabalho de definição dos AIAs, o agente de interfaces deveria executar as seguintes atividades:

1. **Identificação dos AIAs e das atividades que cada AIA realiza**;

2. **Identificação das mensagens a serem recebidas e transmitidas pela interface dos AIAs e o papel do usuário com os AIAs.**

Para facilitar o desenvolvimento dessas atividades, propõe-se que o agente de interfaces seja incrementado com um dicionário sobre um domínio particular, chamado de KE (*Knowledge Enterers*) [Guh 94]. Ele permite a utilização de um vocabulário relacionado com o domínio do problema.

4.6 Agente de Interação da Interface

Este agente identifica como acontece a interação entre o usuário e o sistema, chamada de controle do diálogo. Existem diversas metodologias para descrever o controle do diálogo. Muitas vezes umas metodologias completam a informação não existente em outras.

Este agente deverá suportar a descrição do diálogo através de metodologias conhecidas como CLG, GOMS, UAN ou DTEs (Diagramas de Transição de Estados) (Figura 4.5). No capítulo 7, é apresentado como diversas metodologias necessárias na descrição do projeto da interface são suportadas pela descrição dos AIAs.

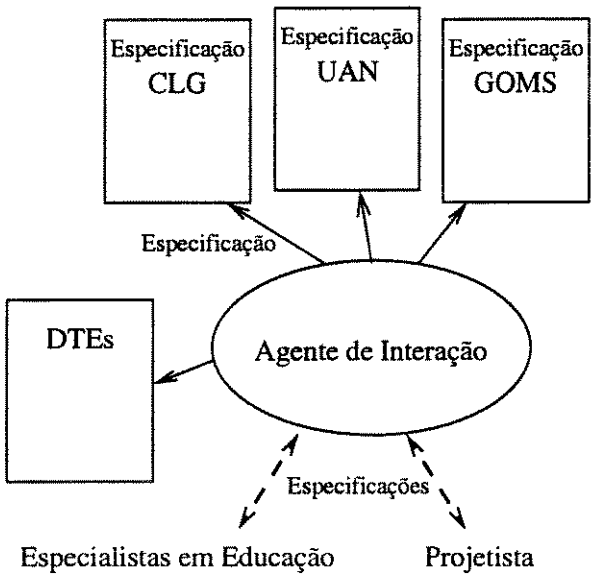

Figura 4.5: Agente de Interação da Interface

4.7 Agente de Prototipagem

Este agente é o encarregado de realizar o projeto detalhado, Figura 4.6, o qual começa com um cenário inicial. Alguns desenhos da tela, objetos, menus, botões, ícones, rótulos de funções e notas sobre o comportamento dos AIAs. Posteriormente, determinam-se as palavras das mensagens, rótulos e escolhas de menus como também a aparência na tela, navegação entre telas, etc.

Este agente mostra de maneira gráfica os AIAs, seguindo um estilo particular (Motif, OpenLook, etc.), permitindo que o projetista ou usuário possa avaliar, modificar e fazer uma análise cognitiva simples (por exemplo, determinar que classes de memórias são impostas aos usuários) no protótipo da interface. Modificações no projeto inicial são geradas justamente deste tipo de análise. Como dito anteriormente, o ambiente pode incorporar ferramentas de prototipação, e por essa razão

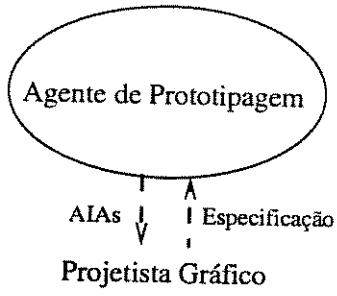

Figura 4.6: Agente de Prototipagem

não será detalhado neste trabalho.

Propõe-se como trabalho futuro que este agente utilize *guidelines* com técnicas de localização, colocação ergonômica, etc. Brown [Bro 89] classifica 302 *guidelines* que podem ser facilmente traduzidos em regras de produção do tipo: **Se** *situação corrente* **Então** *ações*, podendo existir dois tipos de regras: de informação e de modificação.

As regras são de informação quando a ação da regra é simples e informa algum detalhe de projeto. Por exemplo:

Regra: **Se**
está indicando a seqüência de interação
Então
mostre como continuar na seqüência

As regras são de modificação, quando a ação da regra é complexa e indica uma modificação no projeto, por exemplo:

Regra: **Se**
existe cursor(AIA, Área) definido
e
movimento(AIA, Área, Valor)
e
Valor é maior que a média
Então
acione regra de usabilidade(minimize_o_movimento_do_cursor)

Esta classificação não será tratada neste trabalho.

4.8 Agente Gerenciador de Idéias

O agente gerenciador de idéias controla o acesso à idéia, a qual pode ser tratada pelos agentes de forma seqüencial ou paralela. Dessa forma, uma idéia pode estar

sendo editada ao mesmo tempo por um projetista gráfico, por um engenheiro de fatores humanos e por um redator encarregado de determinar as informações que farão parte da apresentação.

O agente gerenciador de idéias também trabalha como um consultor sobre diferentes projetos (tipo de soluções encontradas nos projetos), facilitando a reutilização das especificações, consulta e modificação das mesmas.

Também é possível saber em que situação se encontra o desenvolvimento de determinada idéia, que agente está trabalhando com ela e que atividades faltam serem desenvolvidas.

Uma vez que os agentes terminam um trabalho específico sobre uma idéia, informam aos outros agentes da finalização, pois algum outro agente pode estar esperando a conclusão dessa atividade.

Este tipo de gerenciador de versões existe e pode ser incorporado ao ambiente e não será detalhado neste trabalho.

4.9 Agente de Documentação

Este agente apresenta ao projetista ou ao usuário a documentação formal do projeto da interface. Para isto utiliza diversas informações contidas na idéia. Propõe-se que sejam apresentadas duas descrições textuais principais, a saber:

- Projeto funcional: descreve uma visão geral das tarefas da aplicação e de suas características;
- Projeto detalhado: é uma expansão do projeto funcional formada pela descrição dos AIAs, funcionalidade, controle do diálogo, apresentação, etc.

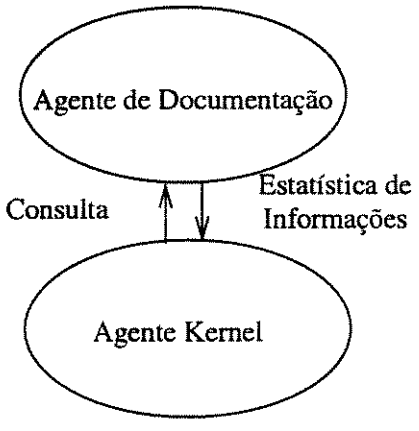

Figura 4.7: Agente de Documentação

Este agente mantém informações relativas a quantos agentes participaram da solução e quantas vezes determinada documentação foi revisada. As informações são consultadas pelo agente *kernel* para ter conhecimento do desenvolvimento do ambiente como um todo e inferir novas regras do comportamento da sociedade, baseado em experiências passadas (Figura 4.7). Este agente deve poder ser configurado para cada instalação de forma a incorporar características específicas de documentação de cada empresa e não será detalhado neste trabalho.

4.10 Agente *Kernel*

O agente *kernel* é o responsável por facilidades relativas à tomada de decisões, coordenação e aconselhamento aos agentes. Entre suas tarefas específicas podemos propor[4]:

- Inicialização de novos agentes no ambiente;
- Eliminação de algum agente do ambiente;
- Inibição do trabalho de algum agente por algum tempo para fins de ordenar o trabalho no ambiente;
- Orientação para o agente no caso explícito de solicitação por parte de um agente;
- Coordenação do trabalho entre os agentes no ambiente, evitando conflitos de atuação, e organização do trabalho do ambiente em geral;
- Informação sobre o desempenho do ambiente como um todo, como quantidades de especificações, modificações, etc.;
- Informação sobre o desempenho de cada agente: tempo que gasta no tratamento de uma solução entre outros;
- Requisição de soluções ao usuário, sobre a necessidade de alguma definição a nível gerencial que o agente *kernel* não tenha conseguido solucionar.

O agente *kernel* deveria ter um controle descentralizado sobre os demais agentes no ambiente com a política de "persuadir" [5] um agente a atuar de uma determinada forma. Dizemos "persuadir" porque a tomada de decisão sempre surge da discussão entre qualquer agente no ambiente e o agente *kernel*. Se o agente *kernel* não consegue convencer a mudar de comportamento, o agente poderá continuar trabalhando como até então.

Para facilitar a implementação da comunicação entre os agentes, propõe-se que este agente conte com a colaboração de um agente chamado de facilitador [Gen 94]. Os agentes usam a linguagem para comunicar suas necessidades e habilidades ao facilitador, que por sua vez se encarrega de transmitir as mesmas da origem para o destino[6].

O agente *kernel* ajuda na tomada de decisões através de um mecanismo global de coordenação de atividades. Decisões erradas tomadas pelo ambiente serão

[4]Algumas destas tarefas são específicas de sistemas multi-agentes e outras são específicas da área de projeto.

[5]São utilizados protocolos de persuasão do tipo apresentado na seção 3.2, página 28.

[6]Os agentes podem estar sendo executados em processadores distintos ou simplesmente em uma estação de trabalho com processos que se comunicam (*pipes*, *streams*, etc.).

corrigidas pelo usuário num diálogo de correção. Este diálogo permite ao agente conhecer algum detalhe a mais sobre a solução de um determinado problema. O agente é capaz de armazenar esse tipo de informação recém aprendida com a participação do usuário (novos planos).

Determinar a forma de como será este diálogo entre o agente *kernel* e o usuário é um dos problemas da *human-agent interaction* [Nor 94]. Esta questão tem diferentes componentes, incluindo como se deve instruir e controlar o agente, a natureza do *feedback* do agente, e a maneira pela qual os agentes oferecem auxílio e informação para o usuário. O agente *kernel* não será detalhado nesta pesquisa, dado que este trabalho concentra-se no problema da descrição do projeto da interface a partir da descrição dos AIAs.

Cada um dos agentes definidos tem funcionalidade diferente mas possuem características comuns em relação a sua arquitetura interna. Na próxima seção é apresentada uma visão geral desta arquitetura que será expandida no próximo capítulo. A definição da arquitetura provê o fundamento do funcionamento e das ferramentas que possam ser construídas, assim como uma estrutura e forma universal para os AIAs.

4.11 Agentes: Arquitetura Interna

Para poder refletir a capacidade de se comunicar, transmitir resultados, poder aprender de resultados anteriores e apresentar um comportamento inteligente, a arquitetura dos agentes proposta foi definida tendo em consideração aspectos provenientes do funcionamento do sistema nervoso central humano [Guy 86].

Este paralelo com o sistema biológico facilita a identificação dos agentes como unidades autônomas dentro da sociedade e permite definir um comportamento próprio para cada agente, seguindo características como memorização, comportamento reativo, aprendizagem, etc.

Assim, uma arquitetura interna é proposta para os agentes do *framework* e para a estrutura dos AIAs. Esta arquitetura está dividida em três partes principais mostradas na Figura 4.8 [Ari 98]:

1. **Sistema central**
   É a parte da arquitetura encarregada do comportamento. Esta parte foi proposta tendo em consideração a analogia do sistema nervoso central, com o objetivo de simular um comportamento "humano" na ação dos agentes. O sistema central pode ser analisado em quatro grandes aspectos:

   - **Método de comportamento**
     Referente ao comportamento do agente, descreve procedimentos para realizar um objetivo (planos), tomada de decisões, gerenciamento das tarefas, aprendizagem, gerenciamento do comportamento direcionado a metas, etc.;

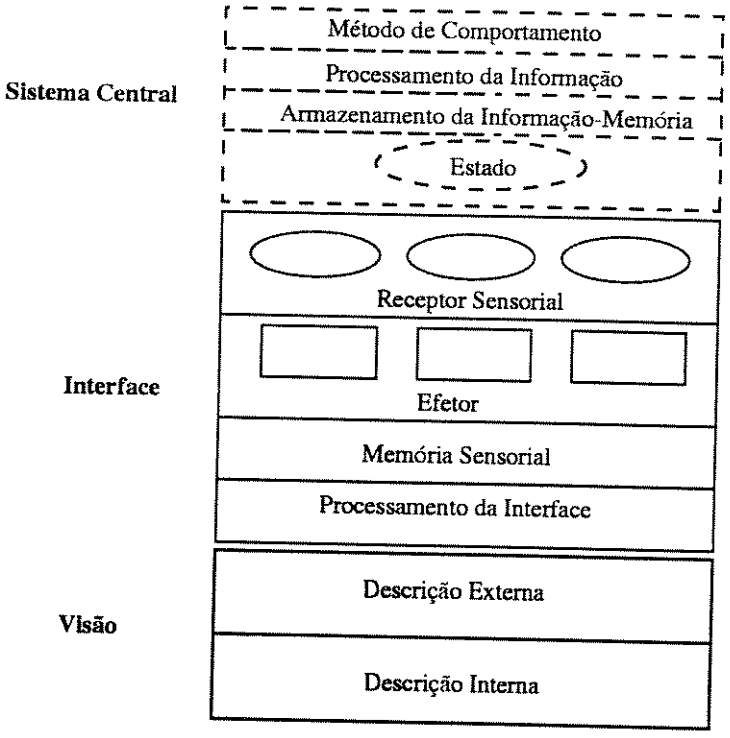

Figura 4.8: Arquitetura Interna do Agente

- **Processamento da informação**
  Referente à estrutura de controle e à análise e classificação da informação recebida pela interface do agente de tal forma que se produzam respostas apropriadas;
- **Armazenamento da informação-memória**
  Referente ao armazenamento da informação que o agente utiliza no seu trabalho;
- **Estado**
  Indica a(s) atividade(s) que está(ão) sendo realizada(s) pelo sistema central. Ele proporciona a todo instante uma imagem da evolução do agente;

2. **Interface**
   É a parte do agente encarregada de perceber o que acontece no ambiente e de se comunicar com o sistema central. Esta parte foi proposta tendo em consideração o sistema sensorial humano e está subdividida em quatro partes principais:

   - **Receptor sensorial**
     Recebe as mensagens (informação) do exterior, de outros agentes ou do usuário[7];

[7]Chamadas também de entrada.

- **Efetor**
  Comunica a resposta para quem solicitou ou transmite mensagens para quem o agente deseja se comunicar[8];
- **Memória sensorial**
  Armazena durante algum tempo pequenas quantidades de informação. Esta memória guarda as soluções encontradas para as últimas mensagens recebidas, como memória de curto prazo [Thi 90]. Facilita o UNDO de vários níveis pois guarda o(s) estado(s) anterior(es) de um agente;
- **Processamento da interface**
  Permite tomar decisões rápidas sem necessidade de intervenção do sistema central.

Na interface, as informações recebidas são classificadas. Assim, uma vez que chega uma mensagem, a interface a classifica em um determinado receptor. Esta pré-classificação existe porque algumas mensagens são tratadas de forma diferente segundo a sua natureza. A Figura 4.9 mostra a arquitetura multi-segmentar[9] da interface, onde cada um dos segmentos, em analogia com a medula espinhal do ser humano, é encarregado de receber um tipo de informação (receptor) e executar um tipo de ação (efetor);

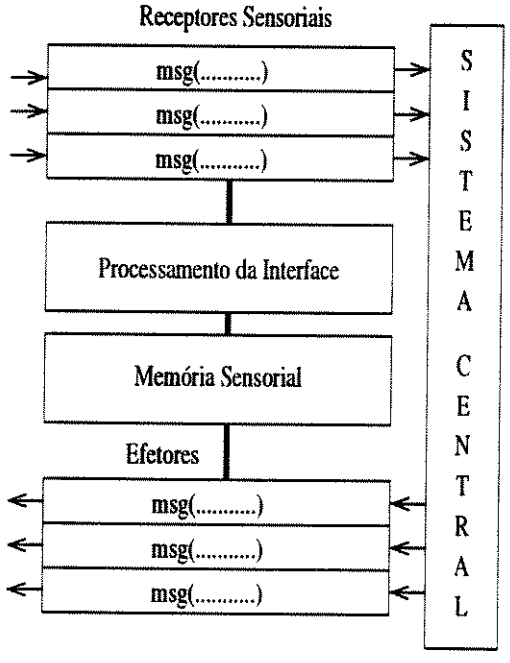

Figura 4.9: Arquitetura Multi-segmentar da Interface

[8]Chamadas também de saída.
[9]Pode também ser chamada de multi-classes.

3. **Visão do agente**
   São as informações sobre si próprio, sobre os demais agentes da sociedade e sobre o tipo de sociedade que ele compõe. Esta parte foi definida baseado na bibliografia de agentes [Ber 92]. Estas informações encontram-se armazenadas em duas descrições principais:

   - **Descrição externa do agente**
     Representação compacta das capacidades do ambiente, dos agentes do ambiente e dos usuários que interatuam com o agente. Corresponde a uma descrição limitada de suas capacidades intrínsecas: "como o agente sabe quais são as capacidades e/ou habilidades dos outros membros da sociedade?";
   - **Descrição interna**
     Representação do funcionamento interno do agente. Os agentes são descritos em termos da sua respectiva arquitetura interna: "como o agente se comporta internamente?".

   O conteúdo destas descrições será detalhado nos próximos capítulos.

Esta arquitetura combina propriedades de técnicas de especificação baseada em regras e descrição de sistemas especialistas. Maiores detalhes de seus componentes são apresentados no próximo capítulo.

4.12 Conclusão

Baseado na abordagem multi-agentes, definimos um conjunto mínimo de agentes para o funcionamento do *framework* de apoio a projeto de interfaces. Cada um dos agentes trabalha para o desenvolvimento do projeto conceitual e detalhado da interface.

A interface da aplicação é especificada como uma sociedade multi-agentes. Ambiente e interface da aplicação são descritas sob a mesma abordagem. A unificação do modelo vem ao encontro da possibilidade de especificar ambientes e interfaces sob a mesma concepção e comprovar a facilidade de utilização da mesma.

Neste capítulo, foram apresentadas as características gerais da arquitetura interna dos agentes. A mesma definição é utilizada para a composição interna dos AIAs. No capítulo 5 são apresentados maiores detalhes da arquitetura dos agentes do ambiente e no capítulo 7 os detalhes de como é construída a arquitetura dos AIAs.

5

Arquitetura Interna dos Agentes

O capítulo anterior descreveu a composição do ambiente e a arquitetura interna geral dos agentes. Este capítulo propõe os diferentes componentes da arquitetura interna de um agente a fim de que ele tenha um comportamento inteligente, isto é, a capacidade de armazenar e manipular representações de conhecimento, ter autonomia e habilidade social, usar e atualizar seu conhecimento, realizar inferências, planejar suas ações e aprender comportamentos para situações novas.

A definição da arquitetura utiliza conceitos da IA como representação de conhecimento, raciocínio, aprendizado e aspectos provenientes do sistema nervoso central humano.

5.1 A Interface do Agente

Na configuração dos agentes, não existe entre eles memória compartilhada [Ish 94]. A comunicação entre agentes é através de mensagens a partir da arquitetura multi-segmentar. Este trabalho não define a sintaxe das mensagens nem a abordagem do protocolo de cooperação que os agentes utilizam. Existem muitos trabalhos que focalizam este estudo. Podemos citar o artigo de Burmeister, Haddadi & Sundermeyer [Bur 95]. Nele, os autores estabelecem a sintaxe e semântica das mensagens, primitivas do protocolo e até uma forma de representação gráfica do protocolo.

Nesta seção, analisamos como as mensagens são tratadas pelos módulos da interface e como se integram com o sistema central. A Figura 5.1 mostra a relação

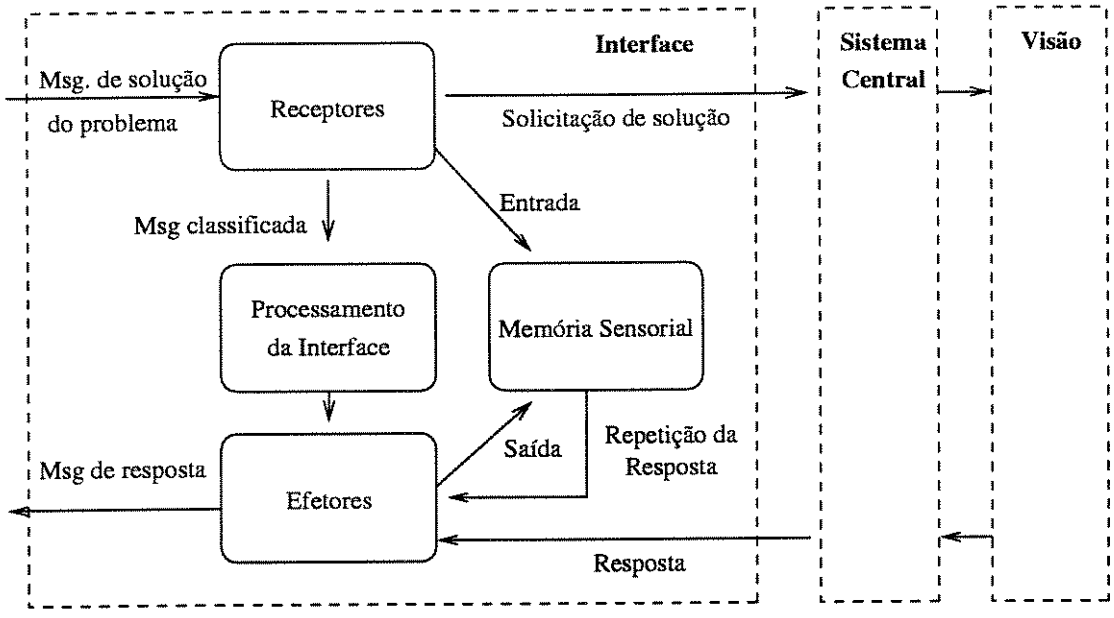

Figura 5.1: Componentes da Interface

entre os componentes da interface.

5.1.1 Receptores e Processamento da Interface

Como apresentado no capítulo anterior, a interface recebe mensagens. A informação contida na mensagem é chamada de "informação sensorial" para diferenciá-la das outras informações que são tratadas dentro das bases de conhecimento e que se referem ao que o agente "sabe" em contraposição ao que o agente "percebe" do exterior.

Uma vez recebida uma mensagem, ela é classificada, isto é, enquadrada em um receptor. Podem existir várias classes de receptores (também chamadas de segmentos) dependendo das necessidades funcionais do agente. Contudo é possível descrever quatro grandes classes:

1. Classes relacionadas com a comunicação: enviam e recebem mensagens do agente *kernel*, dúvidas de outros agentes, pedido de informações sobre a visão do agente, etc.;

2. Classes relacionadas com o conhecimento do agente: são informações relacionadas à execução de tarefas pelo agente;

3. Classes relacionadas com aprendizagem: enviam ou recebem novos planos para tratamento de soluções;

4. Classes relacionadas com a cooperação: enviam ou recebem solicitações de cooperação na solução de problemas.

A mensagem já classificada é enviada pelo receptor ao sistema central para seu tratamento. Contudo, existem mensagens que são tratadas diretamente pela interface (processamento da interface) sem necessidade da chamada ao sistema central. O processamento da interface possui métodos relacionados com a aparência da interface que são executados em forma rápida e independente dos métodos do sistema central.

A interface deve guardar a informação do instante em que uma mensagem foi recebida e tratada. Esta característica é avaliada pelo agente *kernel* para determinar o desempenho do agente.

5.1.2 Efetores e Memória Sensorial

O sistema central, a partir de seu comportamento (módulo de comportamento), estado de trabalho e conhecimento, indicará ao efetor a ação correspondente. Existe então uma relação entre o receptor e o efetor contendo cada um a entrada e a saída do agente. Esta informação assim já processada permite obter informações sobre o conhecimento do agente na solução do problema e é armazenada temporariamente na memória sensorial; isto porque a mensagem pode vir a se repetir outra vez em um curto intervalo de tempo. Neste caso, não se precisa mandar a solicitação de resposta para o sistema central e a formulação da ação é realizada mais rapidamente. Mas este tipo de memorização é temporária. Passado um intervalo de tempo, a interface pode esquecer as informacões e requerer o procedimento normal [Ari 98].

5.2 O Sistema Central do Agente

Como vimos no capítulo anterior, o sistema central é a parte do agente encarregada de coordenar as diversas atividades de tratamento da solução, aprendizagem, alteração de prioridades, delegação de tarefas a outros agentes e troca de diretivas com o agente *kernel*. Ele está formado por quatro grandes aspectos: método de comportamento, processamento da informação, armazenamento da informaçãomemória e o estado. Cada um destes aspectos é tratado por um módulo especializado.

O método de comportamento é o módulo crítico [Mul 95]; por isto, diversas arquiteturas foram estudadas a fim de estabelecer os módulos especializados constituintes do método de comportamento ([Hua 95];[Dun 95];[Jen 95];[Mul 95]). Os seguintes módulos são propostos:

1. **Módulo de inferência**
   Composto por uma máquina de inferência que contém regras de inferência declarativas e genéricas. Ela aplica o conhecimento para a solução do problema e infere novos dados;

2. **Módulo de comportamento**
   Encarregado do comportamento reativo do agente, ele responde às mudanças

no módulo de armazenamento da informação-memória: chegada de novas metas, mensagens de novos resultados, etc.;

3. **Módulo de conhecimento estratégico de comportamento**
   Possui conhecimento sobre as metas correntes dos agentes, raciocínios do agente e ajuda a organizar as metas;

4. **Módulo de avaliação da situação**
   Decide quais atividades serão executadas localmente e quais serão delegadas.

A Figura 5.2 mostra estes módulos juntamente com o módulo de processamento da informação, armazenamento da informação-memória[1] (mostrada como memória de longo e curto prazo) e o estado.

Na busca da solução, o módulo de comportamento é quem recebe uma solicitação da interface e informa ao módulo de avaliação da situação, o qual decide (segundo sua carga de trabalho, distribuição da sociedade e habilidades) se a tarefa será realizada ou não pelo agente. Se a tarefa não foi delegada, o módulo de processamento da informação classifica a informação e, junto com a máquina de inferência, decidem o plano a seguir. O módulo de armazenamento da informação-memória é quem armazena o conhecimento e os dados temporários gerados pelo módulo de inferência, pelo módulo de comportamento e pelo módulo de processamento. Se o módulo de inferência criou um novo plano na solução do problema, ele é aprendido pelo módulo de conhecimento estratégico de comportamento e a solução obtida é enviada para a interface. Na próximas seções, analisamos cada um destes módulos em mais detalhe.

5.2.1 Módulo de Inferência

Para este módulo, propõem-se duas atividades principais:

1. **Tomada de decisão**
   Entre caminhos alternativos de ação é eleito um deles, baseado em padrões específicos de busca. Consiste de uma série de regras de seleção dos planos. Como os planos estão associados a metas, eles podem ser combinados para obter a solução;

2. **Gerenciamento das tarefas**
   Consiste de regras para decomposição das tarefas que compõem a busca de uma solução. Este módulo leva uma relação das tarefas, das metas e do tipo de inferência que está sendo aplicada a cada meta.

[1] Maiores detallhes na seção 5.2.2.

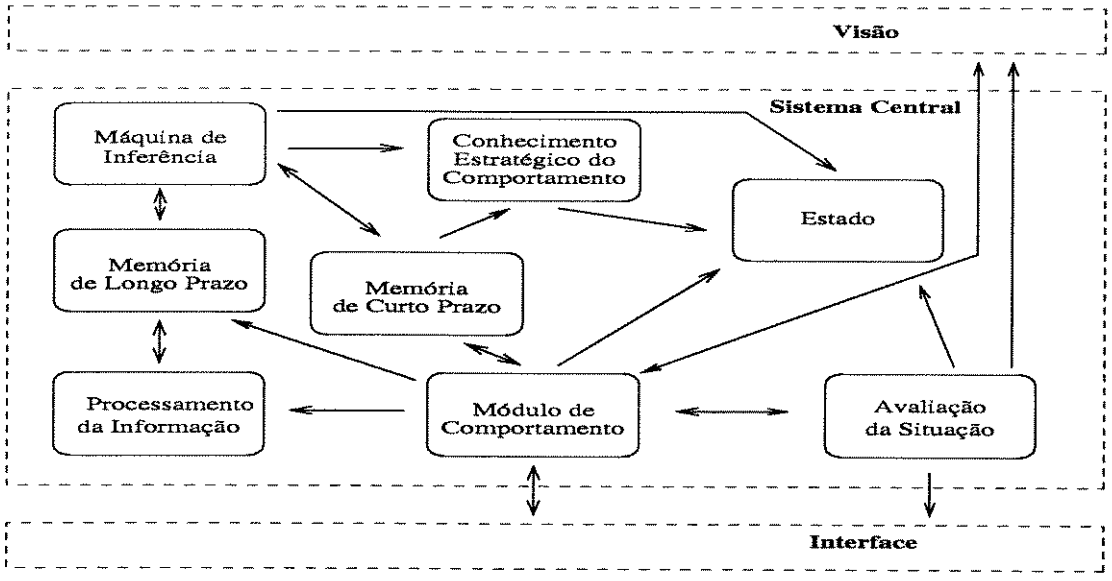

Figura 5.2: Componentes do Sistema Central

5.2.2 Módulo de Armazenamento da Informação-Memória

Baseados na composição da memória humana [Guy 86] propõe-se para cada agente a mesma subdivisão: memória sensorial, memória de curto prazo e a memória de longo prazo. A memória sensorial já foi definida anteriormente e não faz parte do sistema central mas da interface. Embora esta seção trate dos módulos do sistema central, colocamos novamente a definição desta para poder identificar em conjunto as diferentes memórias que fazem parte do agente:

- **Memória sensorial**
  É aquela que retém informações sensoriais durante um curto intervalo de tempo. O objetivo dela é facilitar a resposta do agente para mensagens muito freqüentes. As mensagens armazenadas na memória sensorial dão uma idéia clara de quais elementos são mais comuns no trabalho do agente;

- **Memória de curto prazo**
  É uma memória que contém informações, fatos, números, entre outros, por um curto intervalo de tempo. Ela também é chamada de memória de trabalho porque contém o conhecimento sobre o problema que está sendo resolvido;

- **Memória de longo prazo**
  É a memória que contém o conhecimento do agente e é utilizada para armazenar as novas informações aprendidas pelo agente. O armazenamento de informação na memória de longo prazo[2] pode estar:

[2]Também chamada de base de conhecimentos.

- **Na memória secundária**
  Onde a informação é armazenada com peso fraco ou apenas moderadamente forte, o que caracteriza os tipos de informações não muito utilizados. Os pesos na classificação da informação são um recurso para a busca rápida da mesma;
- **Na memória terciária**
  Informação com peso forte. Fatos ou informações muito utilizadas.

A necessidade destes tipos diferentes é devido ao fato de que o conhecimento do agente não é estático. Ele pode incorporar novos conhecimentos que inicialmente são verificados quanto à sua validade. A variação do peso indica um aprendizado mais seguro. Assim, informações na memória secundária podem passar à memória terciária com a repetição do seu uso, o que faz o peso dela passar de fraco ou moderadamente forte para forte.

Durante o processo de clasificação da informação (seção 5.2.3), é recuperada informação semelhante a partir da memória de longo prazo, a qual é usada para processar a nova informação permitindo, assim, a integração dos módulos de processamento da informação e armazenamento da mesma.

5.2.3 Módulo de Processamento da Informação

O processamento da informação consiste na análise da informação através de um processo de classificação de acordo com suas propriedades. Quando a informação é recebida pelo sistema central, o primeiro passo é transmitir uma chamada a partes diferentes, seletivamente identificadas, da memória de longo prazo. Nela, as novas informações sensoriais são comparadas imediatamente às experiências precedentes de tipos iguais ou semelhantes. Identificam-se, então, os padrões que mais se relacionam com a informação recebida. A informação nova e a velha são comparadas quanto às semelhanças e diferenças, estabelecendo associações com outras informações do mesmo tipo (*case-based reasoning* [Bar 81]). Como resultado obtém-se uma classificação da mesma.

A nova informação é classificada no que chamamos de "classes de informação". Baseando-se nessa classificação, o módulo de inferência busca as respostas à informação recebida.

5.2.4 Módulo de Comportamento

O módulo de comportamento tem que coordenar a atuação dos outros módulos, reagir a eventos, etc. Para este módulo propõem-se os seguintes objetivos:

1. **Monitorar eventos**
   Tratamento rápido de determinados eventos, como interrupção, pelo sistema central de forma diferente;

2. **Coordenar a atuação dos outros módulos**
   Mecanismo de coordenação global dos diferentes módulos do sistema central;

3. **Inferir onde determinada informação será armazenada**
   Determinar, segundo a sua importância, se uma informação será armazenada com peso fraco, com peso moderadamente forte ou com peso forte na memória do agente.

5.2.5 Módulo de Conhecimento Estratégico de Comportamento

Este módulo proporciona os planos que dirigem a busca da solução. Alguns planos são pré-definidos, outros podem ser criados durante o trabalho do agente. Eles são baseados nos resultados bem sucedidos das experiências anteriores. Portanto, para este módulo, propõem-se as seguintes tarefas:

1. **Supervisão dos planos**
   Planos com metas concretas ou metas aproximadas (incerteza);

2. **Aprendizagem**
   Armazenamento de planos comprovadamente úteis. Apóia o módulo de inferência na geração de novos comportamentos.

5.2.6 Módulo de Avaliação da Situação

Para este módulo, propõe-se o gerenciamento de cooperação. O agente mantém referência das tarefas atribuídas a outros agentes e busca algum agente da sociedade que execute uma tarefa específica. Esta atividade é fundamental porque decide sobre os possíveis conflitos que possam existir numa solicitação de cooperação (nenhuma proposta, várias propostas e subdivisão de tarefas).

Este módulo decide, sabendo as descrições externas dos agentes da sociedade e das suas próprias possibilidades, quais tarefas serão realizadas por ele. Assim, propusemos que cada agente comunique ao agente *kernel* os seus desejos de trabalhos e espere a atribuição de permissão.

O agente *kernel*, que vê o desempenho geral da sociedade (estado da rede, quantidades de tarefas por agente, resultados satisfatórios de um agente em relação a outro, etc.), pode negociar com os agentes outro tipo de distribuição das tarefas.

5.2.7 Estado do Sistema Central

Os estados do sistema central dependem da funcionalidade do agente. Um gerador de estados indica o estado do sistema central segundo a atividade que está sendo desenvolvida e que módulos são os responsáveis por esse trabalho.

5.3 Visão do Agente

O componente de visão de um agente está formado por tudo o que ele conhece. Propõe-se que este componente esteja formado por informações referentes a três níveis, a saber:

- Nível do *kernel*;
- Nível de sí mesmo;
- Nível de relacionamento com os outros.

Como apresentado no capítulo anterior, as informações do agente *kernel* e dos outros agentes da sociedade estão na descrição externa. A descrição interna contém as informações sobre as funções internas do agente.

5.3.1 Descrição externa do agente

Está formada pelas seguintes informações:

- Representação do ambiente em que vive;
- Dispositivos físicos que utiliza;
- Habilidades, descrevendo o que este agente pode oferecer aos outros;
- Representação das habilidades dos outros agentes;
- Representação das habilidades do agente *kernel*;
- Descrição dos tipos de usuários que interagem com o agente.

5.3.2 Descrição interna

Está formada pelas seguintes informações:

- Representações dos problemas que o agente pode resolver;
- Representações de como o agente manipula o conhecimento;
- Planos para tratamento da solução.

Contudo, estas descrições não são estáticas; elas evoluem ao longo do tempo. As descrições indicam o uso do ambiente como um todo. Os agentes podem ajustar suas regras de atuação e comportamento baseado no amadurecimento do seu trabalho na sociedade, levando a mudanças nas informações contidas nas descrições correspondentes.

Como o "estado do mundo real" é reportado ao agente através dos receptores, modificações percebidas pelo agente são, segundo a sua importância, colocadas nas descrições.

O agente consulta as descrições com diversos objetivos tais como conhecer sobre o que os outros agentes sabem, ter informação de como pedir auxílio ao agente *kernel*, etc.

Esta é uma característica fundamental que o diferencia de outras arquiteturas, como por exemplo o ambiente CHIRON [Tay 95], onde um arquivo de configuração define quais agentes e quantas instâncias desses agentes devem ser invocados inicialmente. Nesta proposta de arquitetura, os agentes definem a configuração inicial através da consulta da descrição externa dos agentes (troca de informações sobre sua visão), permitindo assim, determinar que agentes se encontram habilitados a trabalhar nesse momento no ambiente.

5.4 Conclusão

Modelar o comportamento humano é um objetivo sempre presente na área de inteligência artificial. Neste trabalho, além deste enfoque, é identificada a possibilidade de estabelecer um paralelo de atuação entre o ambiente e o projetista. O que se pretende é chegar a uma mesma base de comportamento onde agentes e projetistas trabalhem de certa forma de maneira similar, isto é, com conhecimento, regras de atuação, aprendizagem, memorização em intervalos curtos de tempo, etc.

O capítulo anterior concentrou-se na arquitetura interna no aspecto geral. Este capítulo mostra os componentes da arquitetura que servem de orientação e fundamento para o comportamento inteligente dos agentes. O próximo capítulo descreve o funcionamento dos agentes do ambiente enquanto definem o projeto da interface da aplicação.

6

Funcionamento dos Agentes do Framework

Este capítulo propõe detalhes do funcionamento dos agentes do *framework* enquanto um projeto da interface da aplicação está sendo construído. O enfoque principal é dado para aqueles agentes que têm relação direta com a definição do projeto da interface: agente especialista em análise de sistemas, agente de interfaces, agente de interação da interface e agente de prototipagem. Na última seção, é apresentado o conjunto de descrições que fazem parte da idéia e que mostram o resultado do trabalho dos agentes.

6.1 Agente Especialista em Análise de Sistemas

Este agente proporciona um contexto adequado ao analista a fim de documentar a análise das necessidades, análise do usuário, análise das tarefas, alocação de tarefa-função e análise funcional. Veremos a seguir como cada uma dessas atividades é feita:

1. **Análise das necessidades**:
   É uma narrativa que inclui uma sentença simples sobre os objetivos e necessidades da aplicação, conforme [Hix 93]. O agente especialista na análise de sistemas deve colocar à disposição do analista um editor de texto que permita documentar a análise, como vemos na Figura 6.1. Esta narrativa é armazenada (na idéia) com um identificador do projeto chamado de *IDProject*.

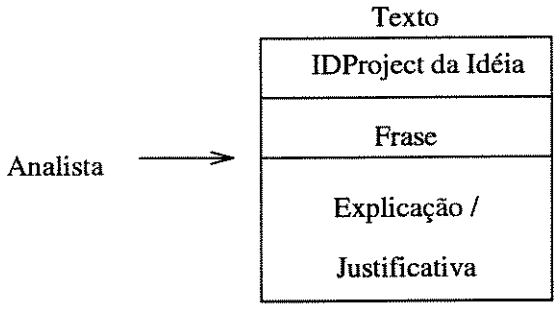

Figura 6.1: Análise de Necessidades feita pelo Analista

2. **Análise do usuário**:
O analista ou engenheiro de fatores humanos deverá ter entrevistado e examinado previamente usuários potenciais para determinar o tipo de classificação psicológica dos usuários. O agente especialista na análise de sistemas deve mostrar ao analista um *menu* com as diferentes características de usuários e sua definição. O analista deve selecionar as características que mais se adequam aos usuários potenciais do sistema. Essas características incluem ([Hix 93];[Ari 97c]):

   (a) Entendimento geral sobre computadores;

   (b) Nível de conhecimento (usuário novato, freqüente, ou não freqüente);

   (c) O hardware com o qual estão mais acostumados (ex: Mac, PC, *workstation*, *mainframe*);

   (d) O software com o qual estão mais acostumados (ex: processadores de texto, bancos de dados, planilhas eletrônicas, etc.)

   (e) Necessidade de acesso a informações relacionadas com o trabalho do usuário (ex: sumário versus informação detalhada);

   (f) Experiência com aplicações relacionadas ou similares;

   (g) Habilidades básicas (ex: digitação)

   (h) Nível de educação;

   (i) Conhecimento e/ou experiência sobre os conhecimentos específicos da organização em que o usuário trabalha;

   (j) Capacidade de memorização do diálogo.

   O agente deve gerar um resumo (texto) das características indicadas e deduzir uma classificação do tipo de usuário (*user_type*). Este resumo poderia gerar um direcionamento com relação ao tipo mais apropriado de interface segundo o tipo do usuário, podendo-se encaixar nas formas que Newman [New 95] descreve como "estilos de interação".

Baseados nas *guidelines* propostas por Brown [Bro 89], a classificação do usuário (*user_type*) pode ser utilizada pelo agente para selecionar, de dentro da base de conhecimentos geral de *guidelines*, aqueles mais adequados para esse projeto. A máquina de inferência do agente selecionaria as regras da base segundo as características dos usuários e estabeleceria uma nova base a ser empregada para esse projeto. Esta base será utilizada pelo agente de prototipagem para validar ou auxiliar o analista durante o projeto de apresentação.

A análise do usuário pode fazer com que sejam introduzidas modificações na análise das necessidades, criando novas características. Neste caso, o analista deverá editar novamente o texto da análise das necessidades;

3. **Análise das tarefas**:
   No nosso trabalho, a análise de tarefas corresponde à descrição da árvore de tarefas em níveis mais abstratos de decomposição (até o terceiro nível).

   A análise de tarefas da aplicação é feita através de uma descrição gráfica onde, por exemplo, o analista facilmente descreve as tarefas e subtarefas (estrutura lógica) envolvidas no uso do novo sistema, segundo a visão do usuário. Esta análise é feita através de uma estrutura em árvore como mostrada na Figura 6.2.

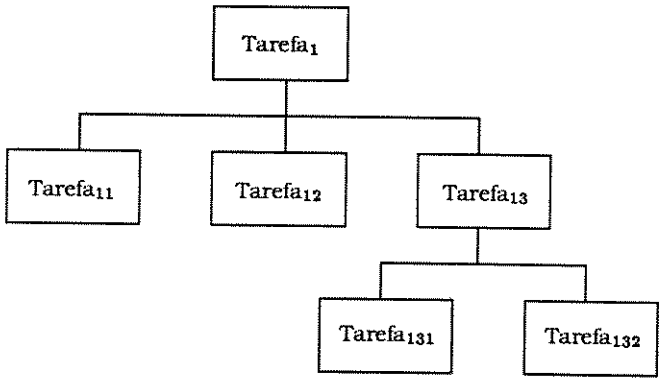

Figura 6.2: Árvore de Tarefas em um Alto Nível de Decomposição

4. **Alocação de tarefa-função e análise funcional**:
   No nosso trabalho, a análise funcional começa com a identificação de quais tarefas na árvore de tarefas são manuais (M), quais são automáticas (A) e quais são participativas (usuário e sistema - P) [Ari 97d].

   Uma vez determinados os tipos de tarefas, o engenheiro de software descreve na árvore as funções associadas às tarefas, como vemos na Figura 6.3, realizando assim as alocações de tarefa-função.

   Para as tarefas manuais, um editor deverá ser habilitado a fim de realizar uma descrição das "funções" manuais do usuário.

As tarefas participativas serão mapeadas em AIAs, as tarefas automáticas serão mapeadas em agentes com a mesma arquitetura interna dos AIAs, que para diferenciá-los são chamados de **agentes da função da aplicação** (AFA).

Para as tarefas automáticas e participativas, o agente deverá possuir o suporte de ferramentas da engenharia de software, como por exemplo um editor de diagramas de fluxo de dados (DFD) para especificar as funções. O importante é obter uma descrição com as seguintes informações:

(a) Nome da função;

(b) Descrição textual da função;

(c) Entradas;

(d) Saídas;

(e) Restrições ou limitações.

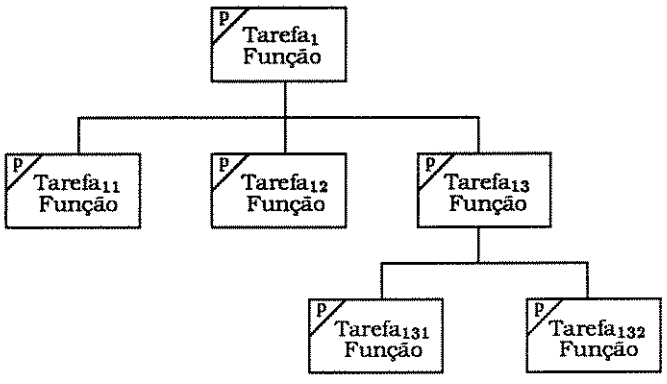

Figura 6.3: Associação entre as Tarefas e as Funções

As tarefas participativas são especiais no contexto deste trabalho, uma vez que indicam exatamente o que é trabalho do usuário e o que é trabalho do sistema. Assim, propõe-se que a informação do que é entrada/saída do usuário e do que é entrada/saída do sistema sejam apresentadas através da notação mostrada na Figura 6.4. Esta notação é uma extensão ao trabalho de Schreiber [Sch 94] através de seus dois modelos.

No modelo das funções (Figura 6.4), são definidas graficamente as entradas e saídas através de elipses, as funções da aplicação como retângulos e as condições como quadrados arredondados. Schreiber descreve as funções indicando os parâmetros de entrada, entrada/saída ou saída e as pré-condições, se existirem. Acrescentamos a esta notação a diferenciação entre quais entradas são do usuário (elipses sombreados) e quais entradas são do sistema. Esta extensão introduz a identificação exata da origem ou destino da informação.

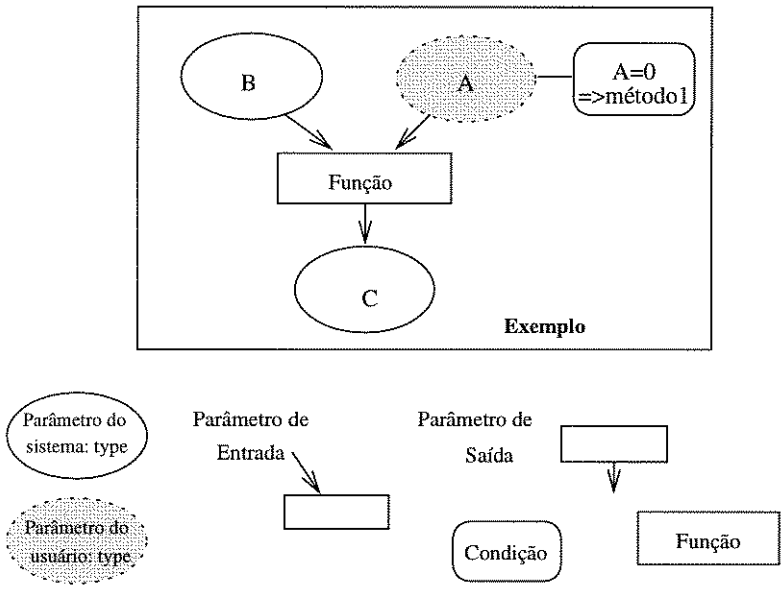

Figura 6.4: Notação e Exemplo do Modelo das Funções

O modelo semântico de dados (Figura 6.5) indica os dados e o tipo de dados. Eles podem ser pré-definidos como tipos de dados simples: *number*, *enumeration*, *string* ou tipo de dados complexo, como *records*, *lists*, *trees*, *tables*, *graphs* ou *sets* [Sch 94].

6.2 Agente de Interfaces

Este agente, baseado na descrição da árvore de tarefas gerada pelo agente de análise, deve gerar automaticamente os AIAs que compõem a solução. Inicialmente, propõe-se que este agente defina, para cada tarefa, um AIA até o terceiro nível, conforme [Hix 93]. A árvore gerada pelo agente é como mostrada na Figura 6.6[1].

Sendo flexível o ambiente definido propõe-se, para o futuro, que exista um especialista que possa selecionar os AIAs baseado em características determinadas para cada tarefa-função. Este especialista estabeleceria quais tarefas são suficientemente autocontidas para serem executadas por uma unidade independente como é um AIA. Um AIA é uma entidade capaz de executar uma atividade específica (funcionalidade), capaz de responder e se comunicar e de lembrar ações passadas. A eleição dos AIAs será baseada nos seguintes conceitos [Ari 97b]:

1. **Independência**
   A tarefa eleita é o suficientemente independente para ser executada por um único AIA?;

[1]Note que as tarefas participativas são mapeadas em AIAs.

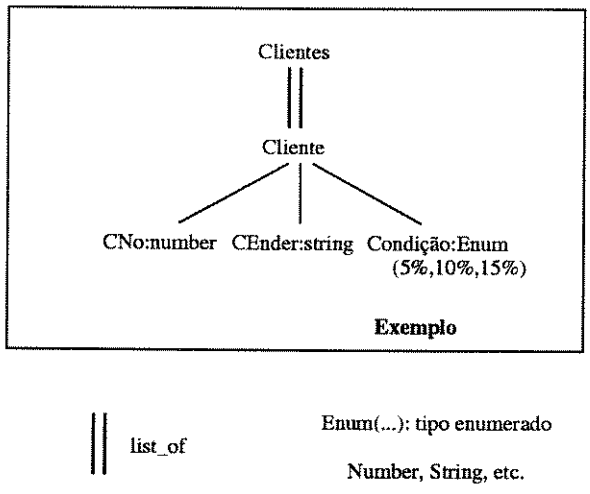

Figura 6.5: Notação e Exemplo do Modelo Semântico de Dados [Sch 94]

2. **Funcionalidade**
   A tarefa-função eleita é o suficientemente complexa a ponto de requerer a definição de um AIA?;

3. **Ergonomia**
   A tarefa-função requer, por questões ergonômicas, ser considerada um AIA?;

4. **Estruturação**
   A tarefa-função requer, por motivos de estruturação da aplicação, ser um AIA?.

Esta análise poderia reestruturar automaticamente a árvore dos AIAs podendo gerar uma árvore diferente da árvore de tarefas. Esta reestruturação, baseada em conhecimentos, não será abordada por este trabalho.

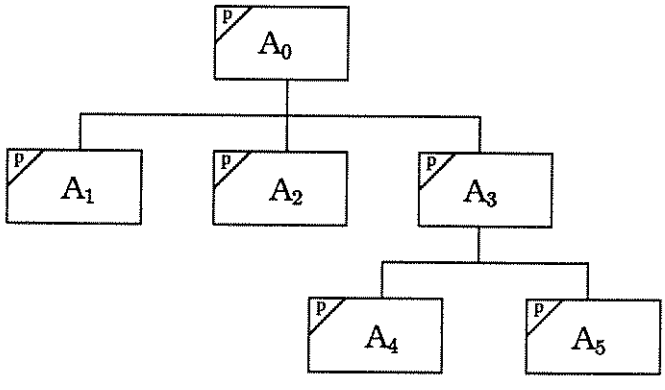

Figura 6.6: Árvore dos AIAs Gerados pelo Agente

6.3 Agente de Interação da Interface

Este agente é o encarregado da descrição da interação do usuário com o sistema. Como vimos no capítulo 2, existem diversas metodologias que podem ser aplicadas para descrever a interação do ponto de vista do usuário. Propomos que este agente apresente ao especialista em interação homem-computador um *template* para cada nó da árvore de tarefas. Este *template* tem o objetivo de expandir a análise de tarefas em um nível mais baixo. O *template* contém informação sobre:

1. **Diálogo**:
   Indicando o comportamento do sistema e do usuário. Este diálogo deverá suportar especificações tipo UAN ou similar, detalhando as ações do usuário, a resposta da interface, o estado da interface e a conexão com as funções da aplicação;

2. **Espaço interativo**:
   Indicando a estrutura através da qual o usuário deve comunicar seus comandos ao sistema. É definido um espaço interativo ou "janela virtual" onde são descritos os objetos e ações sobre os objetos que compõem o ambiente interativo do(s) AIA(s). Este espaço interativo deverá suportar descrições como a especificação a nível sintático da CLG;

3. **Regras de seleção**:
   Indicando as estruturas de controle para escolher um caminho apropriado quando existam alternativas. As regras de seleção deverão suportar descrições como em GOMS;

4. **Estados**:
   Indicando o mapeamento das seqüências de diálogo em estados que a interface pode percorrer e transições que levam a esses estados [Was 85]. Os diagramas de transição de estados ou *statecharts* [Har 87] tornam as descrições do *template* mais completas.

6.4 Agente de Prototipagem

Este agente deve poder gerar um cenário inicial da interface dos AIAs. Este agente pode ser formado de uma ou mais das diferentes ferramentas de prototipação existentes, como por exemplo o *hypercard* que é uma ferramenta de manipulação direta muito utilizada para prototipação [New 95].

O resultado da prototipação deverá ser validado fazendo uso da base de conhecimentos de *guidelines* gerada pela análise do usuário.

6.5 Composição da Idéia

À medida que indicamos o trabalho de cada agente na definição do projeto, são feitas referências sobre como a idéia está sendo formada. Nesta seção, apresentamos, em conjunto, as informações que fazem parte da idéia. Inicialmente, definimos o que na realidade é a idéia. Ela é uma memória de trabalho que contém as informações do projeto. Cada agente (análise, interação, interfaces e prototipagem, etc.) gera uma estrutura na idéia. A máquina de inferência de cada agente é responsável por verificar a consistência e realizar inferências sobre o conteúdo da memória.

A seguir, vemos o tipo de informação que é adicionada por cada agente e a notação da mesma.

O agente de análise armazena, na idéia, o resumo da **análise das necessidades** em um arquivo com o texto da descrição e num fato da seguinte forma:

project(IDProject, archive_name).

onde archive_name é a indicação ao texto editado.

O agente de análise armazena também o resultado da **análise do usuário** em fatos do seguinte tipo:

user_description(IDProject, archive_description).
user_class(IDProject, user_type).

onde archive_description é uma indicação ao texto do resumo das características dos usuários do projeto, e user_type é uma indicação que será utilizada para habilitar as diferentes partes das bases de conhecimentos sobre *guidelines*.

Na **análise das tarefas**, o resultado da edição gráfica da árvore de tarefas são novas informações geradas pelo agente e armazenadas na idéia. Elas têm a seguinte forma:

tree(IDProject, Node, ..., Node).

onde tree indica os diferentes nós folhas da árvore de acordo com uma certa convenção dada.

Quando o analista determina que tarefas são manuais, automáticas ou participativas, o agente gera uma nova informação semântica. Esta informação é adicionada à idéia com fatos do seguinte tipo:

task_type(Node, type).

onde type pode ser M, A ou P.

O resultado da **alocação de tarefa-função** são novas informações semânticas adicionadas à idéia sobre as associações entre as tarefas e as funções do sistema:

allocation(Node_tree, Name_function).

onde existem vários fatos de allocation indicando o nome do nó da árvore e a função associada.

Na **análise funcional**, o projetista deverá fornecer as informações sobre o nome da função, descrição, restrições e limitações, que são armazenados como:

function(Name_function, archive_name_function).

onde archive_name_function é uma indicação ao texto que contém essas descrições.

Para completar esta informação e definir as entradas/saídas das funções e o que é entrada/saída do usuário e o que é entrada/saída do sistema o agente deve gerar, a partir da Figura 6.4, informações com a seguinte sintaxe:

function_inputs(Name_function, origin(name_subfunction)).
function_outputs(Name_function, origin(name_subfunction)).
arguments_type(name_subfunction, data_type).

onde origin indica se a subfunção é do usuário (user) ou do sistema (system) e data_type o tipo da subfunção descrita no modelo semântico de dados.

Do modelo semântico de dados (Figura 6.5) são geradas as seguintes informações:

data_model(data_type, type).

onde data_type é o nome do tipo de dados e type é o tipo de dados simples ou complexo. Para o caso de type ser um tipo de dados complexo, nova informação é adicionada:

simple_data_type(type, simple_type).

onde simple_type pode ser: number, enumerate ou string.

As condições, equivalentes às pré-condições do modelo de Schreiber, são descritas através de regras de produção da forma:

Name: Se condition
Então action

onde name indica o nome da função ou entrada/saída a quem corresponde a condição.

O agente de interfaces deve gerar a árvore com os AIAs e nova informação semântica é adicionada à idéia com regras do seguinte tipo:

tree_of_AIA(Node, A_i).

que estabelecem a relação entre as tarefas e os AIAs.

O agente da interação da interface deve gerar, na idéia, as informações relativas ao **diálogo**, com a seguinte sintaxe:

dialog(AIA, Task, User_action, Feedback, State, Application).

onde AIA indica o AIA associado a essa descrição, Task, indica o nome da tarefa, User_action, indica as ações do usuário, o Feedback a resposta do sistema, State indica o estado da interface e Application indica a conexão com a aplicação. Estas informações são análogas às existentes na metodologia UAN e foram eleitas por representarem um conjunto completo necessário para a especificação do diálogo. O importante é que, para as descrições das ações do usuário (entradas), existam as respostas do sistema (*feedback*) que, por sua vez, podem estar ligadas com as funções da aplicação e conectadas com um estado em particular.

O agente da interação da interface também deve gerar, na idéia, as informações do **espaço interativo**, o qual é formado por objetos de informação, ações sobre os objetos, indicação das variáveis de estado que são utilizadas como argumentos e comandos de acesso. Nessas descrições, existe uma indicação de a que AIA pertencem essas informações, o que identifica o componente de apresentação de cada AIA. As informações têm a seguinte sintaxe:

virtual_interactive_objects(A_i(object, ..., object), ...).

virtual_interactive_actions(A_i(action, ..., action), ...).

virtual_interactive_state_variables(A_i(name, ..., name), ...).

virtual_interactive_access(A_i(command, ..., command), ...).

O agente de interação deve gerar as **regras de seleção**, definidas pelo projetista, com a seguinte sintaxe:

identification: Se condition
Então action
Senão action

onde condition ou action podem ser um conjunto de fatos associados de forma disjuntiva (conectivo "ou") ou conjuntiva (conectivo "e"). A associação das regras com a tarefa para a qual foi definida a regra é armazenada na idéia em fatos do seguinte tipo:

rules(task, identification).

O agente de interação também deve gerar os **estados** do DTE, através de regras que indicam a mudança de um estado para outro. Nelas, existe uma indicação de a que AIA pertencem:

DTE_A_i: Se condition
Então action

Finalmente, o agente de prototipagem armazena na idéia fatos do seguinte tipo:

presentation(IDProject, archive_layout).

onde archive_layout é uma indicação aos arquivos onde se encontra a prototipação do cenário.

No próximo capítulo, serão apresentadas, através de um exemplo, as diversas informações que compõem a idéia com a sintaxe definida neste capítulo.

6.6 Conclusão

Este capítulo mostra como os diversos agentes contribuem na definição da análise das necessidades, dos usuários e das tarefas, conforme a literatura de interfaces homem-computador. É apresentada uma extensão ao trabalho de Schreiber [Sch 94] que permite identificar, na especificação das funções, quais entradas/saídas são do/para o usuário. Esta extensão facilita documentar a origem ou destino da informação. A alocação de tarefa-função permite a identificação dos AIAs que compõem a interface da aplicação. Os AIAs organizam na sua estrutura as funções, dados, condições, espaço interativo e controle do diálogo como veremos no próximo capítulo.

7

Exemplo de Aplicação

Este capítulo apresenta as informações necessárias que o projetista deve fornecer para descrever a interface da aplicação. Estas informações dão a base para que possam ser aplicadas as diferentes metodologias de projeto de interfaces. Através de um exemplo, vemos como estas informações se relacionam entre si e como é possível utilizar as vantagens de diversas metodologias para o projeto completo da interface.

7.1 Exemplo Proposto

O exemplo mostrado a seguir é uma descrição de requisitos de informação de um sistema que configura e desenha a face de um relógio. Este exemplo é uma adaptação daquele originalmente proposto por Johnson em seu livro *Human Computer Interaction* [Joh 92].

Através do exemplo, serão focalizadas as diferentes atividades realizadas pelos agentes durante o desenvolvimento da interface da aplicação, conforme seqüência abaixo:

1. Análise das necessidades;
2. Análise do usuário;
3. Análise das tarefas;
4. Alocação de tarefa-função e análise funcional;
5. Identificação dos AIAs;
6. Identificação do diálogo;

7. Identificação do espaço interativo;
8. Identificação das regras de seleção;
9. Fluxo de controle do diálogo;
10. Construção dos AIAs.

Veremos como estas atividades, junto com o ciclo de vida do software da interface, produzem o projeto da interface para o problema do relógio.

7.2 Análise das Necessidades

Segundo vimos no capítulo anterior, a análise das necessidades está formada por uma frase simples e uma descrição das características. Para este exemplo, o projetista fornece:

Frase: Configurar relógio

Características:

- O relógio pode ter diferentes faces, diferentes cores e as horas podem ser indicadas de diferentes maneiras;
- Configurar significa habilitar ao usuário um programa para definir a face, a cor e os ponteiros do relógio, segundo a escolha do usuário.

Esta informação é armazenada na idéia no seguinte fato:

project(ProjetoNro1, relógio).

onde relógio é a indicação ao texto editado.

7.3 Análise do Usuário

A partir de uma lista de características do usuário, foram selecionadas pelo projetista como verdadeiras para o exemplo:

- Acostumado com interfaces de manipulação direta;
- Não muito experiente;
- Com habilidades de manipulação direta, menus, etc.

Esta informação é armazenada na idéia nos seguintes fatos:

user_description(ProjetoNro1, usuário_relógio).

user_class(ProjetoNro1, usuário_tipo_medio).

onde usuário_relógio é uma indicação ao texto do resumo das características dos usuários do projeto. E usuário_tipo_medio é uma classificação obtida pelo ambiente que indica um direcionamento ao tipo de *guidelines* que serão utilizados pelo agente de prototipação.

7.4 Análise das Tarefas

Distingue o tipo de tarefa que é realizada do ponto de vista do usuário. As tarefas definidas pelo projetista são as apresentadas na Figura 7.1.

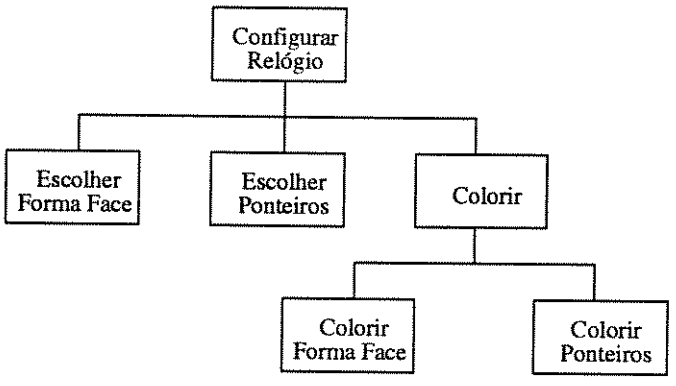

Figura 7.1: Árvore de Tarefas para o Relógio

A árvore da Figura 7.1 é armazenada na idéia com uma certa convenção:

tree(ProjetoNro1, configurar relógio, escolher forma face, escolher ponteiros, colorir, colorir forma face, colorir ponteiros).

7.5 Alocação de Tarefa-Função e Análise Funcional

Todas as tarefas são definidas pelo projetista como tipo "participativas" (Figura 7.2), isto é, tarefas que são realizadas pelo usuário e pelo sistema.

Esta informação é adicionada à idéia incrementando fatos do seguinte tipo:

task_type(configurar relógio, p).

task_type(escolher forma face, p).

task_type(escolher ponteiros, p).

task_type(colorir, p).

task_type(colorir forma face, p).

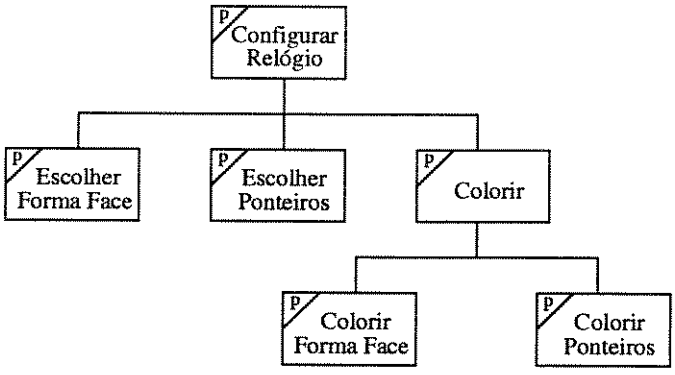

Figura 7.2: Classificação das Tarefas

task_type(colorir ponteiros, p).

A partir dos conceitos de engenharia de software, o projetista determina, no nível mais alto, as funções da aplicação, como vemos na Figura 7.3. São elas:

1. preview_face(face, cor_face);
2. preview_ponteiros(ponteiro, cor_ponteiro);
3. colorir(x, cor);
4. iniciar(face, cor_face, ponteiro, cor_ponteiro);
5. desenha_face(face, cor_face);
6. atualiza_ponteiros;
7. desenha_ponteiros_hora(ponteiro, cor_ponteiro, hora_atual);
8. desenha_ponteiros_minuto(ponteiro, cor_ponteiro, hora_atual);
9. desenha_ponteiros_segundo(ponteiro, cor_ponteiro, hora_atual);
10. calcula_angulo_hora(hora_atual);
11. apaga_ponteiro_hora(hora_anterior).

A alocação de tarefa-função é armazenada na idéia como:

allocation(escolher forma face, preview_face(face, cor_face)).

allocation(escolher forma face, desenha_face(face, cor_face)).

allocation(escolher ponteiros, preview_ponteiro(ponteiro, cor_ponteiro)).

allocation(colorir, colorir(x, cor))., etc.

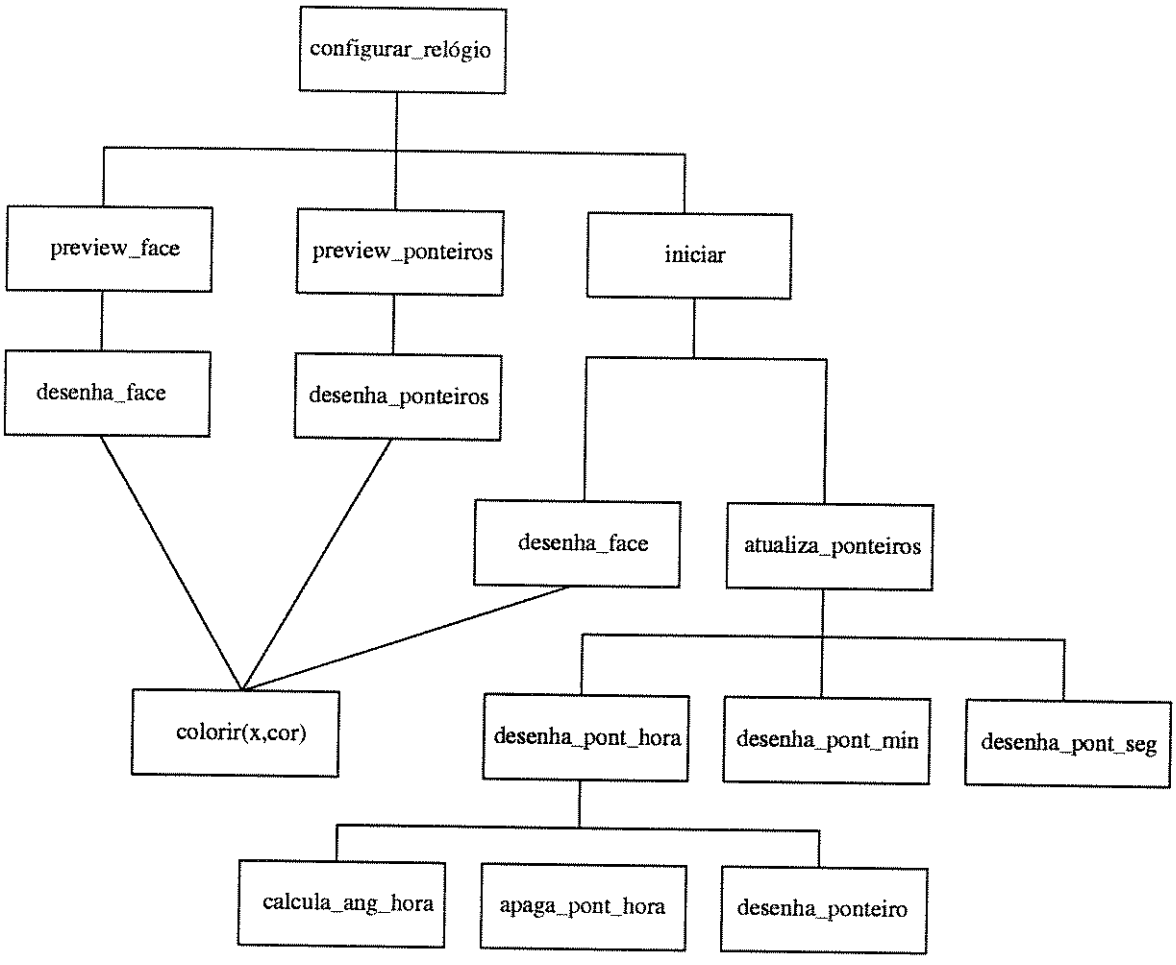

Figura 7.3: Algumas Funções do Software para o Relógio

Conforme a engenharia de software, para cada função, o projetista deverá fornecer as seguintes informações: nome da função, descrição textual da função, entradas, saídas e restrições ou limitações.

No nosso caso serão descritas somente o "nome", a "descrição textual" e as "restrições e limitações". As entradas e saídas serão descritas em forma gráfica como veremos na Figura 7.5.

Para o exemplo, a descrição de algumas das funções é a seguinte:

Função 1:

- Nome: preview_face(face, cor_face);
- Descrição: apresenta um *preview* da face eleita ou de uma face *default* e permite ao usuário mudar a cor da face;
- Restrições: a face deve pertencer ao conjunto de faces e a cor_face deve pertencer ao conjunto de cores.

que são armazenadas na idéia como:

function(preview_face(face, cor_face), archive_preview_face).

onde archive_preview_face é uma indicação ao texto que contém a descrição da função e das restrições.

Função 2:

- Nome: preview_ponteiros(ponteiro, cor_ponteiro);
- Descrição: apresenta um *preview* do ponteiro eleito ou de um ponteiro *default* e permite ao usuário mudar a cor;
- Restrições: a forma do ponteiro deve pertencer ao conjunto de ponteiros e a cor_ponteiro deve pertencer ao conjunto de cores.

que são armazenados na idéia como:

function(preview_ponteiros(pont, cor_pont), archive_preview_ponteiros).

Função 3:

- Nome: iniciar(face, cor_face, ponteiro, cor_ponteiro);
- Descrição: desenha a face selecionada com a cor eleita e registra uma função de *callback* para atualizar os ponteiros a cada segundo;
- Restrições: sem restrições.

que são armazenados na idéia como:

function(iniciar(face, cor_face, pont, cor_pont), archive_iniciar).

Função 4:

- Nome: desenha_face(face, cor_face);
- Descrição: esta função desenha a face e chama a função colorir(x, cor);
- Restrições: sem restrições.

que são armazenados na idéia como:

function(desenha_face(face, cor_face), archive_desenha_face).

Função 5:

- Nome: colorir(x, cor);
- Descrição: pinta o objeto x com a cor eleita;

- Restrições: a cor deve pertencer ao conjunto de cores.

que são armazenados na idéia como:

function(colorir(x, cor), archive_colorir_x).

Para definir as entradas e saídas das funções, utilizamos a extensão ao trabalho de Schreiber [Sch 94] onde, além de indicar as entradas do usuário e do sistema, especificam-se as condições associadas com as entradas, as saídas e as funções.

A especificação das estruturas de dados para o exemplo do relógio são descritas pelo projetista em forma gráfica como apresentado na Figura 7.4[1].

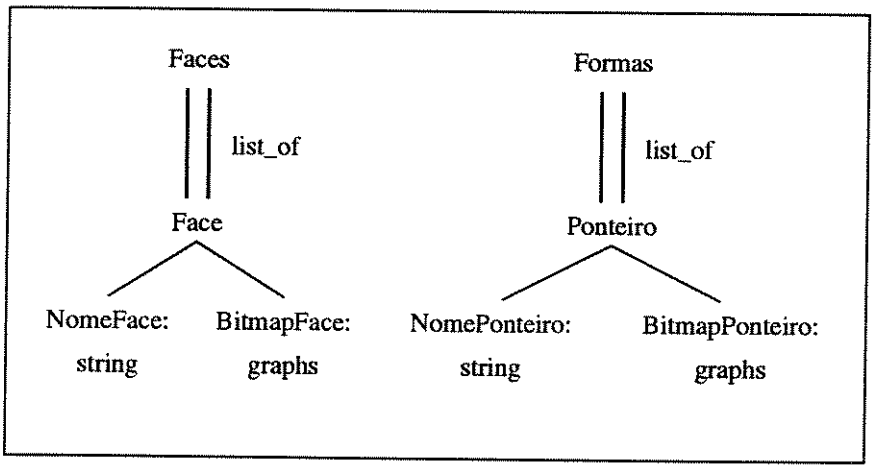

Figura 7.4: Modelo Semântico de Dados para o Exemplo do Relógio

Estas informações semânticas são armazenadas na idéia como:

data_model(faces, list_of(nome_face, bitmap_face)).

data_model(formas, list_of(nome_ponteiro, bitmap_ponteiro)).

simple_data_type(nome_face, string).

simple_data_type(bitmap_face, graphs).

simple_data_type(nome_ponteiro, string).

simple_data_type(bitmap_ponteiro, graphs).

A Figura 7.5 mostra o modelo das funções para o exemplo. Estas informações semânticas são armazenadas na idéia como:

function_inputs(preview_face, user(face, cor_face), system(face_default, cor_default)).

[1]Notação mostrada no capítulo anterior.

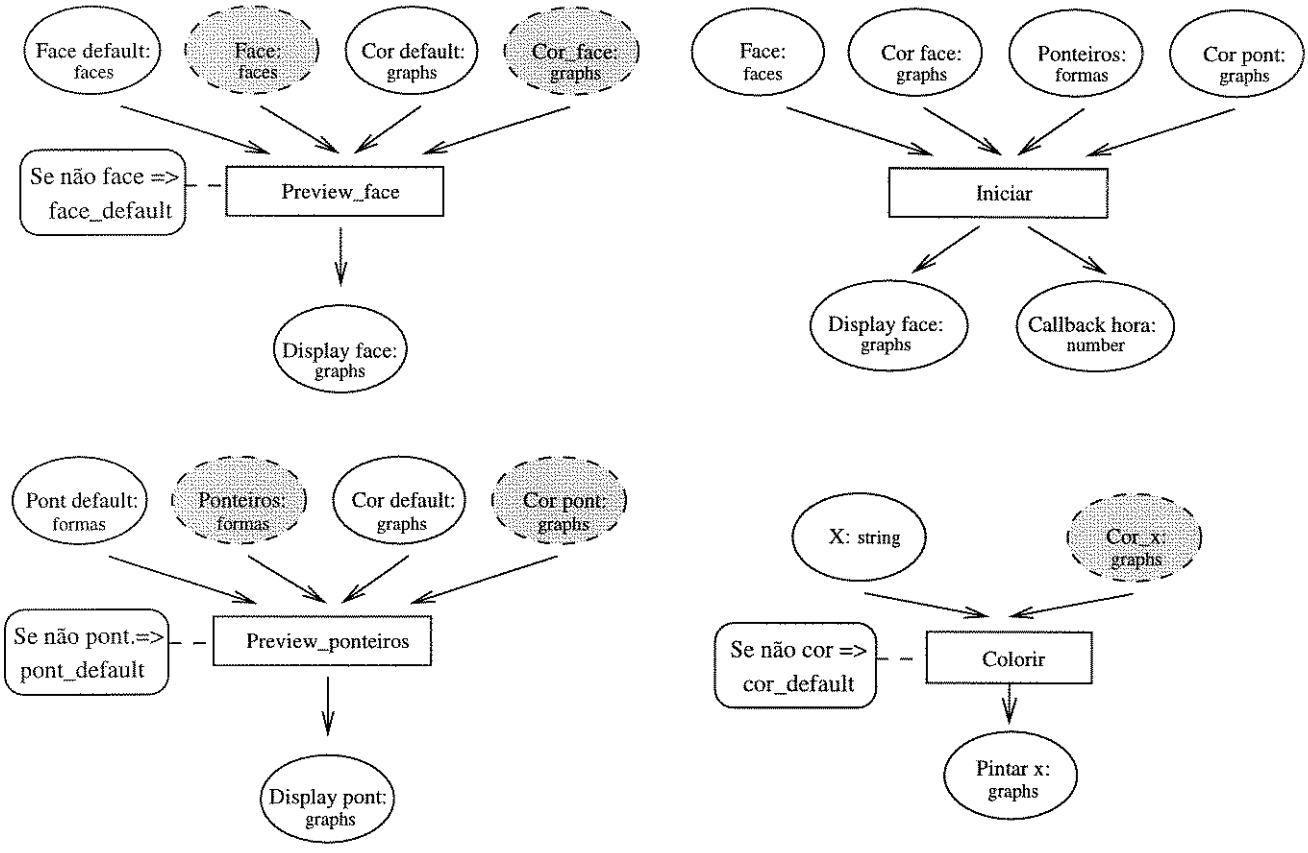

Figura 7.5: Modelo das Funções para o Exemplo do Relógio

function_inputs(preview_ponteiros, user(ponteirosçor_ponteiros), system (ponteiros_default, cor_default)).

function_inputs(iniciar, system(face, cor_face, ponteiros, cor_ponteiros)).

function_inputs(colorir, user(cor_x), system(x)).

function_outputs(preview_face, system(display_face)).

function_outputs(preview_ponteiros, system(display_ponteiros)).

function_outputs(iniciar,system(display_face, registra_callback)).

function_outputs(colorir, system(pintar_x)).

arguments_type(face, faces).

arguments_type(cor_face, graphs).

arguments_type(face_default, faces).

arguments_type(cor_default, graphs).

arguments_type(ponteiros, formas).

arguments_type(cor_ponteiros, graphs).

arguments_type(ponteiros_default, formas).

arguments_type(cor_default, graphs).

arguments_type(cor_x, graphs).

arguments_type(x: string).

arguments_type(display_face, graphs).

arguments_type(display_ponteiros, graphs).

arguments_type(registra_callback, number).

arguments_type(pintar_x, graphs).

preview_face:	Se	não face eleita
	Então	mostrar face_default
preview_ponteiros:	Se	não ponteiros eleitos
	Então	mostrar ponteiros_default
colorir:	Se	não cor eleita
	Então	mostrar cor_default

7.6 Identificação dos AIAs

O agente de análise, como explicado no capítulo anterior, associa automaticamente AIAs às tarefas de níveis mais alto como mostrado na Figura 7.6.

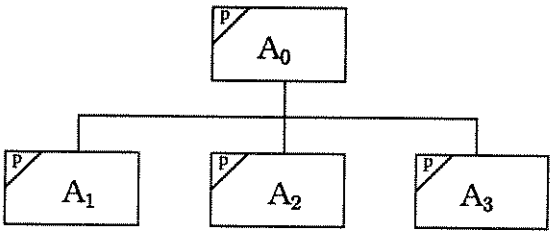

Figura 7.6: Árvore dos AIAs para o Exemplo do Relógio

Nova informação semântica é adicionada à idéia com regras do seguinte tipo:

tree_of_AIA(configurar_relógio, A_0).

tree_of_AIA(escolher_forma_face, A_1).

tree_of_AIA(escolher_ponteiros, A_2).

tree_of_AIA(colorir, A_3).

que estabelecem a relação entre as tarefas e os AIAs.

7.7 Identificação do Diálogo

O projetista deve fornecer a descrição do diálogo para cada tarefa/subtarefa conforme a Figura 7.2.

Tarefa 0: Configurar relógio - AIA associado: A_0

1. Escolher a forma da face ou
2. Escolher a forma dos ponteiros ou
3. Colorir

Tarefa 1: Escolher a forma da face - AIA associado: A_1

1. Determinar a forma da face;
   (a) Selecionar a forma da face.

Tarefa 2: Escolher a forma dos ponteiros - AIA associado: A_2

1. Determinar a forma dos ponteiros;
   (a) Selecionar a forma dos ponteiros;

Tarefa 3: Colorir - AIA associado: A_3

1. Colorir a face do relógio ou
   (a) Selecionar a cor da face;
2. Colorir os ponteiros do relógio;
   (a) Selecionar a cor dos ponteiros.

Neste ponto verificamos que é necessário detalhar:

- Selecionar forma da face;

- Selecionar forma dos ponteiros;
- Selecionar cor.

elas são funções semelhantes que podem ser instanciações da seguinte tarefa:

Tarefa 4: Selecionar objeto

1. Mover o cursor para o objeto a ser selecionado;
2. Pressionar e soltar o botão esquerdo do mouse; o objeto selecionado será destacado.

Ao descrever as tarefas 1, 2, 3 e 4 o agente automaticamente deve completar a Figura 7.2 gerando a Figura 7.7.

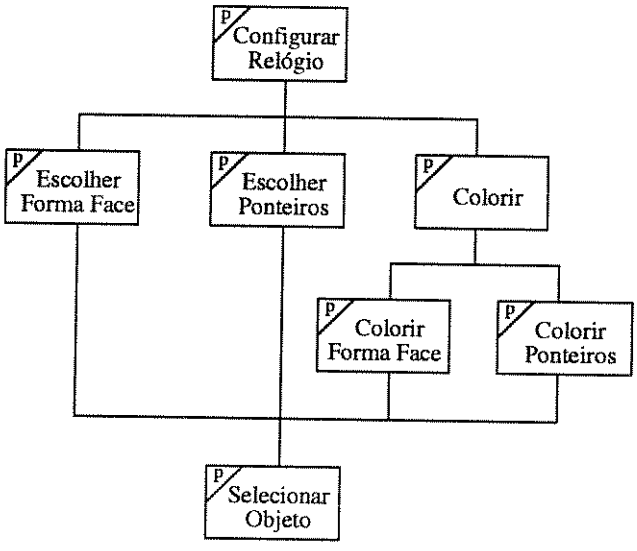

Figura 7.7: Árvore Completa com as Tarefas de Nível mais Baixo

O diálogo será descrito através das seguintes informações:

1. Nome do AIA;
2. Tarefa associada ao AIA;
3. Ações do usuário para realizar a tarefa;
4. *Feedback* do sistema;
5. Estado que o sistema alcança com as ações do usuário;
6. Função da aplicação associadas às ações do usuário.

com a seguinte sintaxe:

dialog(AIA, tarefa, ações, *feedback*, estado, função).

Nas tabelas seguintes vemos as informações necessárias para as tarefas do exemplo descritas na notação UAN.

Tarefa: Configurar relógio;			
ações do usuário	**feedback**	**estados**	**função**
escolher forma face |	tabela faces !	face selec.	preview_face
escolher ponteiros |	tabela ponteiros !	ponteiro selec.	preview_ponteiros
colorir	tabela cores		colorir(x, cor)

Tarefa: Escolher a forma da face			
ações do usuário	**feedback**	**estados**	**função**
selecionar face	relógio com nova face		preview_face(face, cor_face)

Tarefa: Escolher a forma dos ponteiros			
ações do usuário	**feedback**	**estados**	**função**
selecionar ponteiro	nova forma ponteiro		preview_ponteiros(pont, cor)

Tarefa: Colorir			
ações do usuário	**feedback**	**estados**	**função**
Colorir a face |			
Colorir os ponteiros			

Tarefa: Colorir os ponteiros do relógio;			
ações do usuário	**feedback**	**estados**	**função**
selecionar cor	pintar o ponteiro		colorir(ponteiro, cor)

Tarefa: Colorir a face do relógio;			
ações do usuário	**feedback**	**estados**	**função**
selecionar cor	pintar face		colorir(face, cor)

Tarefa: Selecionar Objeto;			
ações do usuário	**feedback**	**estados**	**função**
~[X];Mv	X!		
M^			

onde ~[X] indica mover o cursor (~) até onde está o objeto ([X]), Mv significa pressionar o botão do mouse, M^ significa soltar o botão do mouse e X! significa destacar com uma mudança de cor o objeto (X) utilizando a notação UAN [Hix 93][2].

7.8 Identificação do Espaço Interativo

O espaço interativo é uma forma de descrever "telas" onde o projetista deverá indicar os objetos de informação (*virtual_interactive_objects*) que são apresentados ao

[2]A metodologia UAN descreve as tarefas de mais alto nível na forma de nomes de tarefas associadas por relações temporais (escolha (|), independência (&), etc.). As tarefas de nível mais baixo como a tarefa "selecionar objeto" descrevem movimentos do usuário acionando dispositivos de entrada/saída através de uma notação própria [Hix 93].

usuário, as ações (*virtual_interactive_actions*) requeridas sobre esses objetos, dar indicação das variáveis de estado (*virtual_state_variables*) que são utilizadas como argumentos *default* entre as ações, e dar indicação de uma classe especial de comandos que são requeridos para acessar a configuração do relógio e que são chamados de comandos de entrada (*virtual_interactive_access*).

O tipo de informação do espaço interativo é facilmente descrito pela metodologia CLG na descrição a nível sintático. Para o exemplo do relógio, temos os seguintes objetos, ações, variáveis de estado e comandos de entrada, mostrados na tabela com a notação de CLG.

Configurar=	(a command context		
	State_variables=(set:	face	
		cor_face	
		ponteiros	
		cor_ponteiros	**variáveis**
		face_default	**de estado**
		ponteiros_default	
		cor_default	
		hora	
	Display_areas=(set:	area_preview_face	
		area_preview_ponteiros	
		tabela_faces	**objetos**
		tabela_ponteiros	
		tabela_cores)	
		iniciar	
	Commands=(set:	selecionar_face_na_tabela	
		selecionar_ponteiro_tabela	
		selecionar_cor_da_face	**ações**
		selecionar_cor_dos_ponteiros	
		selecionar_iniciar)	
	Entry_commands=(set:	configurar_relógio))	**acessar**

que serão guardados na idéia com a seguinte sintaxe:

virtual_interactive_objects(A_0(iniciar), A_1(area_preview_face, tabela_faces), A_2 (area_preview_ponteiro,tabela_ponteiros),A_3(tabela_cores, area_ preview_face, area_preview_ponteiros)).

virtual_interactive_actions(A_0(selecionar_iniciar), A_1 (selecionar_face_na_tabela), A_2 (selecionar_ponteiro_tabela), A_3(selecionar_cor_da_face, selecionar_cor_dos_ponteiros)).

virtual_interactive_state_variables(A_0(hora), A_1(faceçor_face, face_default), A_2 (ponteiros, cor_ponteiros,ponteiros_default),A_3(cor_default)).

virtual_interactive_access(A_0(configurar_relógio)).

onde existe uma indicação explicita de a que AIA pertencem as informações, o que estabelece a ligação entre a apresentação e o software da interface encapsulado pelo AIA.

A Figura 7.8 apresenta um protótipo da aparência da tela de configuração do relógio. A construção de protótipos envolve a produção de versões simplificadas do sistema que ilustrem as características especificadas no espaço interativo. Esta tela pode ser gerada pelo agente de prototipagem fazendo uso de alguma ferramenta de prototipagem como por exemplo o *Hypercard*. A tela é armazenada na idéia como:

presentation(ProjetoNro1, protótipo_tela).

onde protótipo_tela é uma indicação ao arquivo onde se encontra esta descrição.

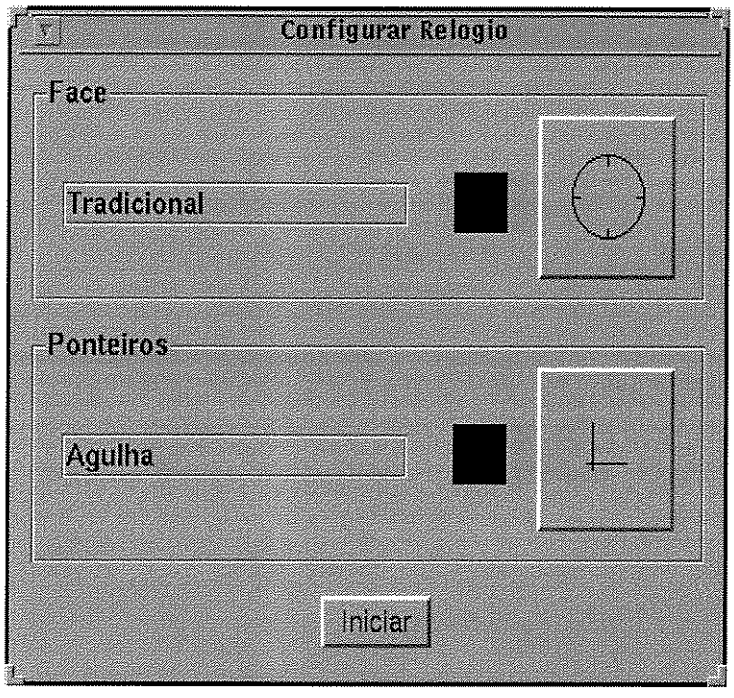

Figura 7.8: Protótipo da Tela para Configurar o Relógio

As informações do espaço interativo e o protótipo especificam a apresentação das telas da interface da aplicação.

7.9 Identificação das Regras de Seleção

As regras de seleção são importantes na descrição de eventos gerados que não fazem parte da descrição da tarefa. A metodologia UAN não possui ferramentas para esta descrição. A metodologia GOMS é, neste caso, mais apropriada. A descrição usando a metodologia GOMS para o exemplo do relógio é a seguinte:

```
Goal: configurar_relógio
        Goal: escolher_forma_face
                Goal: preview_face
                        desenhar_face na area_preview_face
                        colorir_face_cor
        Goal: escolher_ponteiros
                Goal: preview_ponteiros
                        desenhar_ponteiros na area_preview_ponteiro
                        colorir_pont_cor
        Goal: iniciar
                Goal: desenhar_face_cor
                        atualiza_ponteiros
```

e as regras de seleção definidas para este exemplo são:

```
R1:  Se      eleição de configurar relógio
     Então   chamar programa de configurar relógio
     Senão   nada acontece

R2:  Se      eleição de iniciar
     Então   chamar a desenha_face_cor_face
     e       atualiza_ponteiros cada seg.
     Senão   continue

R3:  Se      não é o desenho inicial do relógio
     Então   apaguem_ponteiros
     e       desenha_ponteiros
     Senão   desenha_ponteiros
```

Uma vez definidas as regras de seleção, o projetista deve indicar a que tarefa elas correspondem. Esta informação é armazenada na idéia nos seguintes fatos:

```
rules(configurar relógio, R1).

rules(configurar relógio, R2).

rules(escolher ponteiros , R3).
```

As regras[3] de seleção são armazenadas na idéia no mesmo formato, isto é, como regras de produção.

[3]Este tipo de descrição pode ser amplamente explorada determinando, por exemplo, regras que configurem a cor do relógio de forma *default* segundo as cores que o usuário estabeleceu no seu *rootwindow* ou outro tipo de informação com o objetivo de aprimorar o programa de configuração do relógio.

7.10 Fluxo de Controle do Diálogo

Na literatura, utilizam-se amplamente os DTEs para descrever o fluxo de controle do diálogo. Eles favorecem a especificação seqüencial do comportamento e permitem completar a descrição prevista pela UAN[4]. Contudo, existem diferentes críticas com relação à complexidade de um DTE de todo o sistema. Por isso, optamos por DTEs específicos para cada AIA, como mostrado na Figura 7.9, reduzindo a explosão de estados.

O projetista deve fornecer os estados e as transições para cada AIA que compõe a interface que está sendo projetada. Por exemplo, para o AIA A_0 temos:

DTE_A_0 : Se pressionar botão esquerdo mouse sobre iniciar
e EstadoAnterior = neutro
Então início mostrar o relógio e a hora
e EstadoAtual = mostrar-hora

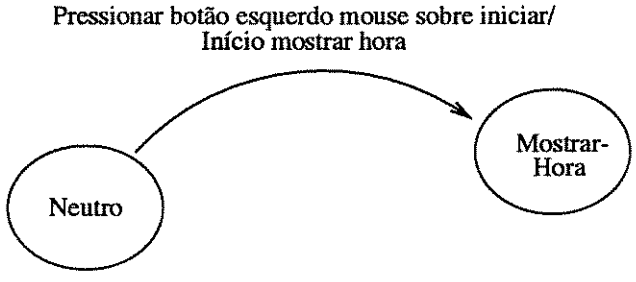

Figura 7.9: DTE para A_0

Para o AIA A_1 temos (Figura 7.10):

DTE_A_1 : Se mover o cursor para eleger a face
e EstadoAnterior = neutro
Então início destaque do nome da face
e EstadoAtual = mostrar-tabela

DTE_A_1 : Se pressionar botão esquerdo mouse
e EstadoAnterior = mostrar-tabela
Então início preview face-eleita
e EstadoAtual = face-eleita

DTE_A_1 : Se soltar o botão esquerdo
e EstadoAnterior = face-eleita
Então finaliza preview
e EstadoAtual = neutro

[4]Através da UAN é possível ter uma noção dos estados pelos quais uma tarefa passa. Mas, não existe relação com o estado anterior da tarefa.

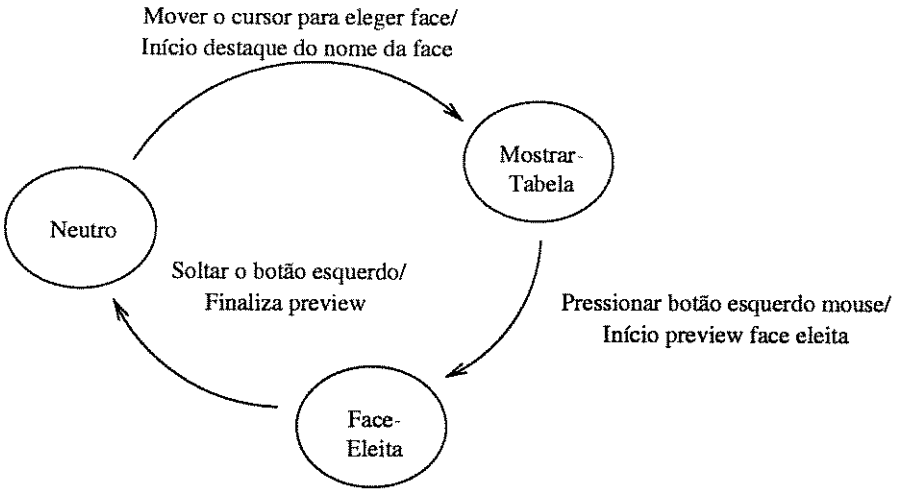

Figura 7.10: DTE para A_1

Para o AIA A_2 temos (Figura 7.11):

DTE_A_2 : Se mover o cursor para eleger os ponteiros
e EstadoAnterior = neutro
Então início destaque do nome dos ponteiros
e EstadoAtual = mostrar-tabela

DTE_A_2 : Se pressionar botão esquerdo mouse
e EstadoAnterior = mostrar-tabela
Então início preview ponteiros-eleitos
e EstadoAtual = ponteiro-eleito

DTE_A_2 : Se soltar o botão esquerdo
e EstadoAnterior = ponteiro-eleito
Então finaliza preview
e EstadoAtual = neutro

Para o AIA A_3 temos (Figura 7.12):

DTE_A_3 : Se mover o cursor para eleger a cor da face
e EstadoAnterior = neutro
Então início destaque cor
e EstadoAtual = mostrar-cor-face

DTE_A_3 : Se pressionar botão esquerdo mouse
e EstadoAnterior = mostrar-cor-face
Então início colorir-face
e EstadoAtual = cor-eleita

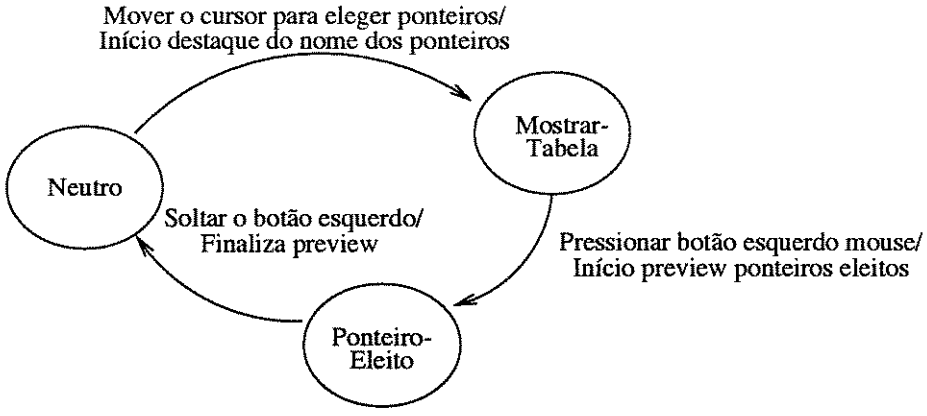

Figura 7.11: Diagrama de Transição de Estados para A_2

DTE_A_3 : Se soltar o botão esquerdo
e EstadoAnterior = cor-eleita
Então finaliza selecionar cor
e EstadoAtual = neutro

DTE_A_3 : Se mover o cursor para eleger a cor dos ponteiros
e EstadoAnterior = neutro
Então início destaque cor
e EstadoAtual = mostrar-cor-ponteiros

DTE_A_3 : Se pressionar botão esquerdo mouse
e EstadoAnterior = mostrar-cor-ponteiros
Então início colorir-ponteiros
e EstadoAtual = cor-eleita

DTE_A_3 : Se objeto eleito
e EstadoAnterior = neutro
Então início colorir-objeto
e EstadoAtual = pinta-objeto

DTE_A_3 : Se objeto colorido
e EstadoAnterior = pinta-objeto
Então finaliza colorir
e EstadoAtual = neutro

7.11 Resumo do Conteúdo da Idéia

Nas seções anteriores, foram definidas as informações fornecidas pelo projetista e armazenadas na idéia com seus diferentes formatos. Estas informações serão

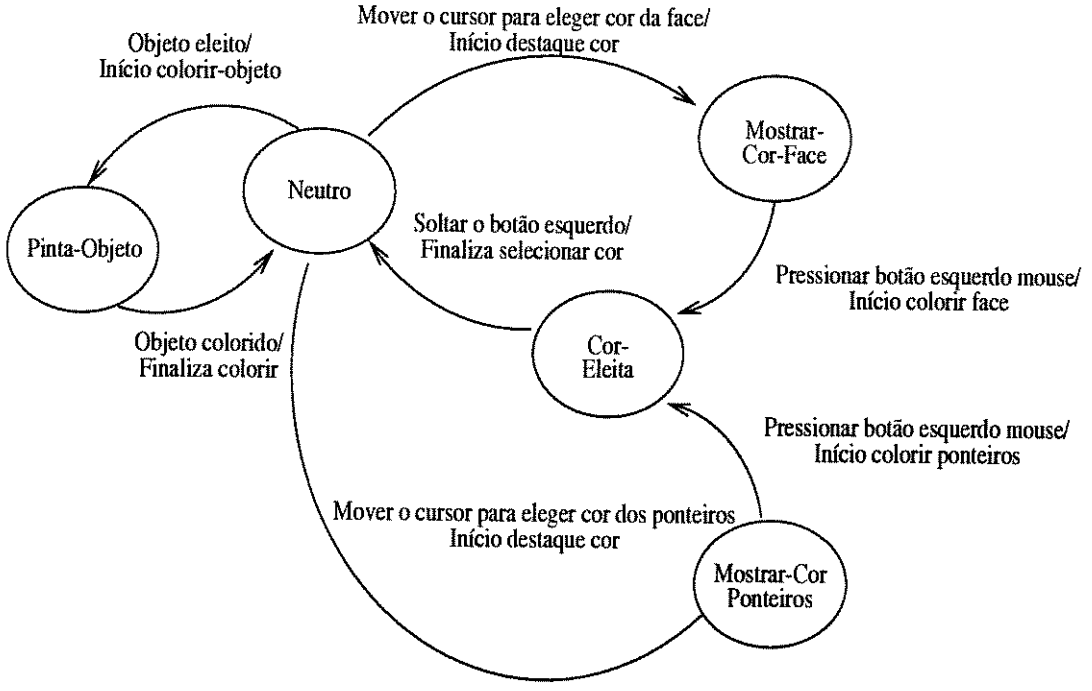

Figura 7.12: Diagrama de Transição de Estados para A_3

tratadas pelo ambiente a fim de gerar o conteúdo dos AIAs. Na próxima tabela existe um resumo de todas as informações fornecidas pelo projetista e que são armazenadas na idéia.

Análise das Necessidades
project(ProjetoNro1, relógio).
Análise do Usuário
user_description(ProjetoNro1, usuário_relógio).
user_class(ProjetoNro1, usuário_tipo_medio).
⋮
Análise das Tarefas
tree(ProjetoNro1, configurar relógio, . . .).
⋮
Alocação de Tarefa-Função
task_type(configurar relógio, p).
⋮
allocation(colorir, colorir(x, cor)).
⋮

Análise Funcional
function(colorir(x, cor), archive_colorir_x).
⋮
function_inputs(colorir, user(cor_x), system(x)).
⋮
function_outputs(colorir, system(pintar_x)).
⋮
colorir: Se não cor eleita Então ...
⋮
arguments_type(cor_x, graphs).
⋮
data_model(faces, list_of(nome_face, bitmap_face)).
⋮
simple_data_type(nome_face, string).
⋮

Identificação dos AIAs
tree_of_AIA(configurar_relógio, A_0).
⋮

Identificação do Diálogo
dialog(A_0, configurar_relógio, ...).
⋮

Identificação do Espaço Interativo
virtual_interactive_objects(A_1(area_preview_face, ...),...).
⋮
virtual_interactive_actions(A_1(selecionar_face_na_tabela),...).
⋮
virtual_interactive_state_variables(A_0(hora),...).
⋮
virtual_interactive_access(A_0(configurar_relógio)).
⋮

Regras de Seleção
R1: Se eleição de configurar relógio Então ...
⋮
rules(configurar relógio, R1).
⋮

Regras de Controle do Diálogo
DTE_A_1: Se pressionar o botão esquerdo do mouse Então ...
⋮
Cenários
presentation(ProjetoNro1, protótipo_tela).
⋮

Vemos que as diferentes metodologias contribuem para uma descrição mais completa e formal do projeto de uma interface com o usuário. Isto evidencia a necessidade de um ambiente aberto como o proposto, onde podem ser colocados novos agentes cada um deles direcionados para o suporte das diversas metodologias.

7.12 Construção dos AIAs

Nesta seção, vemos como é possível obter facilmente o conteúdo dos AIAs a partir das descrições contidas na idéia. As informações são colocadas adequadamente e permitem ter uma especificação completa das tarefas que cada AIA executa. A importância da descrição dos AIAs está no fato de poder organizar e reunir as diversas especificações de tal forma a mostrar a interface como um conjunto integrado de informações que guiam seu desenvolvimento posterior.

Para cada AIA definido na etapa de identificação, é definida a sua arquitetura interna. Conforme definido no capítulo 4, a arquitetura está formada por um sistema central que é o comportamento ou funcionalidade do AIA, a interface que indica tudo aquilo que o AIA recebe ou envia ao exterior e a parte da visão que é a documentação sobre a funcionalidade do AIA, como veremos a seguir.

7.12.1 Sistema central

1. **Método de comportamento**
   O ambiente deve colocar automaticamente como conteúdo do método de comportamento de cada AIA todas as referências às funções que estabelecem a finalidade do AIA. Estas referências surgem das descrições de allocation contidas na idéia.

   Por exemplo, a função preview_face corresponde ao AIA A_1, associado à tarefa "escolher forma face" e descrito por:

   allocation(escolher forma face, preview_face(face, cor_face)).

   function(preview_face(face, cor_face), archive_preview_face).

   Todas as funções relativas à tarefa "escolher forma face" pertencem ao que chamamos de "método de comportamento" do AIA A_1;

2. **Processamento da informação**
No processamento da informação, deve-se considerar dois tipos de informação: a informação relativa à aplicação e a informação relativa à estrutura de controle do AIA. A primeira é obtida de forma automática pelo ambiente a partir das regras de seleção descritas na seção 7.9. Estas regras permitem escolher entre métodos de comportamento, quando existe mais de um método para realizar uma tarefa ou objetivo, como por exemplo:

```
R3: Se      não é o desenho inicial do relógio
    Então   apaguem_ponteiros
    e       desenha_ponteiros
    Senão   desenha_ponteiros
```

A informação relativa à estrutura de controle é colocada pelo projetista de forma não automática. Para isto, é necessário que o projetista defina aquelas regras necessárias para estabelecer o fluxo de controle do AIA. Por exemplo, o AIA A_2 possui uma função chamada de colorir(ponteiros, cor_ponteiros) que, na realidade, é a função de outro AIA (A_3) que é colorir(x, cor). A regra definida para mostrar este fluxo de controle é:

```
RC: Se      colorir(ponteiros, cor_ponteiros)
    Então   acionar efetor para A3
```

3. **Armazenamento da informação-memória**
Esta parte do sistema central contém o modelamento conceitual das estruturas de dados da aplicação para a interface do usuário. Concordamos com Schreiber [Sch 94] na importância, para a descrição da interface, da especificação das estruturas de dados e das funções relevantes.

A partir das descrições feitas no modelo semântico e modelo das funções mostrado na seção 7.5, nas Figuras 7.4 e 7.5, o ambiente deve gerar automaticamente o tipo de dados manipulados pelo AIA. Por exemplo, para o AIA A_1 temos:

```
data_model(faces, list_of(nome_face, bitmap_face)).

function_inputs(preview_face, user(face, cor_face), system(face_default,
cor_default)).

function_outputs(preview_face, system(display_face)).
```

Estas descrições complementam o projeto da interface e permitem examinar os dados dentro do contexto dos AIAs em estágios iniciais, sendo uma referência ao projeto que será tratado pelos programadores;

4. **Estado**
   Esta parte do sistema central permite ao projetista definir o fluxo de controle do diálogo na interface. Para este tipo de descrições, o ambiente associa automaticamente as regras dos diagramas de transição de estados para cada AIA (seção 7.10). A descrição dos estados permite comprovar se não existem lacunas no comportamento de cada AIA.

7.12.2 Interface

1. **Processamento da interface**
   O processamento da interface existe para permitir tomar decisões rápidas que podem ser feitas pela própria interface sem necessidade de intervenção do sistema central. Neste caso, é necessário considerar dois tipos de fontes de informação: um semi-automático que provém da descrição de *dialog* para as tarefas de nível mais baixo (Figura 7.7). Isto é, no caso do exemplo, a partir da tarefa "seleciona objeto", traduzida para a seguinte regra pelo projetista:

   RPI_1 Se mover o cursor até X e pressionar botão mouse
   Então mudança de cor no objeto X

   A outra parte da idéia que indica processamento da interface surge do modelo semântico de dados (condições na Figura 7.5) em forma automática. Por exemplo:

   preview_face: Se não face eleita
   Então mostrar face_default

   Esta regra indica que se for o início, a interface mantém informação de qual é a face *default* a mostrar e não necessita ser buscada no sistema central através de um procedimento normal;

2. **Receptores**
   Os receptores são a parte da interface encarregada de reconhecer as entradas. Como dito no capítulo 4, os receptores estão divididos em classes que facilitam o tratamento das entradas. O ambiente utiliza a informação da descrição do espaço interativo (*virtual_interactive_actions*) para identificar as classes de receptores em forma automática, isto é, as ações requeridas sobre os objetos da "janela virtual" permitem definir as classes de entradas recebidas pelos AIAs. Por exemplo, para o AIA A_0 temos:

Classe de Receptores
Selecionar iniciar

   Para o AIA A_1 temos:

Classe de Receptores
Selecionar face na tabela

Para o AIA A_2 temos:

Classe de Receptores
Selecionar ponteiros tabela

Para o AIA A_3 temos:

Classe de Receptores
Selecionar cor da face
Selecionar cor dos ponteiros

3. **Efetores**
Os efetores são as ações do AIA e estão também divididas em classes. Para definir as classes de efetores, o ambiente utiliza a informação da descrição do espaço interativo (*virtual_interactive_objects*), isto é, os objetos de informação que são apresentados ao usuário na "janela virtual" permitem definir as classes de saídas em forma automática. Por exemplo, para o AIA A_0 temos:

Classe de Efetores
Iniciar

Para o AIA A_1 temos:

Classe de Efetores
Area preview face
Tabela faces

Para o AIA A_2 temos:

Classe de Efetores
Area preview ponteiros
Tabela ponteiros

Para o AIA A_3 temos:

Classe de Efetores
Tabela cores
Area preview face
Area preview ponteiros

4. **Memória sensorial**
   É a parte da interface que armazena, durante algum tempo, pequenas quantidades de informação. Esta característica deve aumentar a facilidade na recuperação de erros (UNDO). Em geral é desejável que o sistema da aplicação proveja um mecanismo para desfazer algumas ações. Para isto, a memória sensorial guarda o estado anterior do AIA, as entradas e saídas a fim de retornar a um estado anterior diretamente.

   A memória sensorial também serve para implementar o REDO. Greenberg & Witten [Thi 90] investigaram o uso dos mecanismos de história e viram que o usuário resubmete uma porção substancial dos comandos. Assim, a idéia de prover comandos de REDO é empiricamente bem fundamentada, e não meramente uma possibilidade técnica.

   O projetista pode então definir o mecanismo de UNDO e REDO através da descrição de procedimentos explícitos que fazem parte da memória sensorial. É uma tarefa que não pode ser automatizada pois depende da aplicação, como por exemplo, não se deve ter UNDO em um jogo de xadrez.

7.12.3 Visão do agente

É a parte do AIA encarregada de documentar a capacidade do AIA. Esta parte não é totalmente automatizada. O ambiente deve mostrar um sistema de configuração que apresenta as informações da idéia para serem configuradas pelo projetista. O objetivo principal da parte de visão é documentar as funcionalidades externas que o AIA oferece (descrição externa) e o funcionamento interno de cada AIA (descrição interna).

Para a descrição externa de A_0 a informação contida na identificação do espaço interativo (*virtual_interactive_actions*) que descreve quais são as ações permitidas para o AIA seria selecionada pelo projetista. Para os demais AIAs, o projetista seleciona a descrição das funções associadas a cada AIA. Por exemplo, para o AIA A_0:

- Selecionar iniciar.

Para o AIA A_1:

- Mostrar preview da face eleita;
- Mostrar preview da face *default*.

Para o AIA A_2:

- Mostrar preview dos ponteiros eleitos;
- Mostrar preview dos ponteiros *default*;
- Desenhar ponteiros;

- Apagar ponteiros;
- Calcula o ângulo da hora.

Para o AIA A_3:

- Colorir a face eleita ou a face *default*;
- Colorir os ponteiros eleitos ou os ponteiros *default*.

A descrição interna dos AIAs, deve ser gerada automaticamente a partir da análise funcional, isto é, o conteúdo de *allocation* permite conhecer que funções fazem parte de cada AIA. Por exemplo, para o AIA A_0:

- iniciar(face, cor_face, ponteiro, cor_ponteiro).

Para o AIA A_1:

- preview_face(face, cor_face).
- desenha_face(face, cor_face).

Para o AIA A_2:

- preview_ponteiros(ponteiro, cor_ponteiro).
- desenha_ponteiros(ponteiro, cor_ponteiro, hora_atual)., etc.

Para o AIA A_3:

- colorir(x, cor).

A parte da visão permite conhecer todas as funções externas e aquelas que pertencem ao desenvolvimento interno.

7.12.4 Exemplo do Funcionamento dos AIAs

Nesta seção vemos o funcionamento dos AIAs no tratamento das seguintes tarefas do usuário:

1. Escolher a forma da face;
2. Escolher os ponteiros do relógio;
3. Mudar a cor dos ponteiros.

mostradas nas Figuras 7.13, 7.14 e 7.15.

Na Figura 7.13, o AIA A_1 recebe uma ação do usuário através do receptor (classe - selecionar face tabela). O estado do AIA A_1 passa de "neutro" para "mostrar-tabela". O processamento da interface destaca a tabela de faces (efetor da classe - tabela faces). É acionado o método de comportamento preview_face. A função preview_face faz chamadas à função desenha_face e colorir(face, cor).

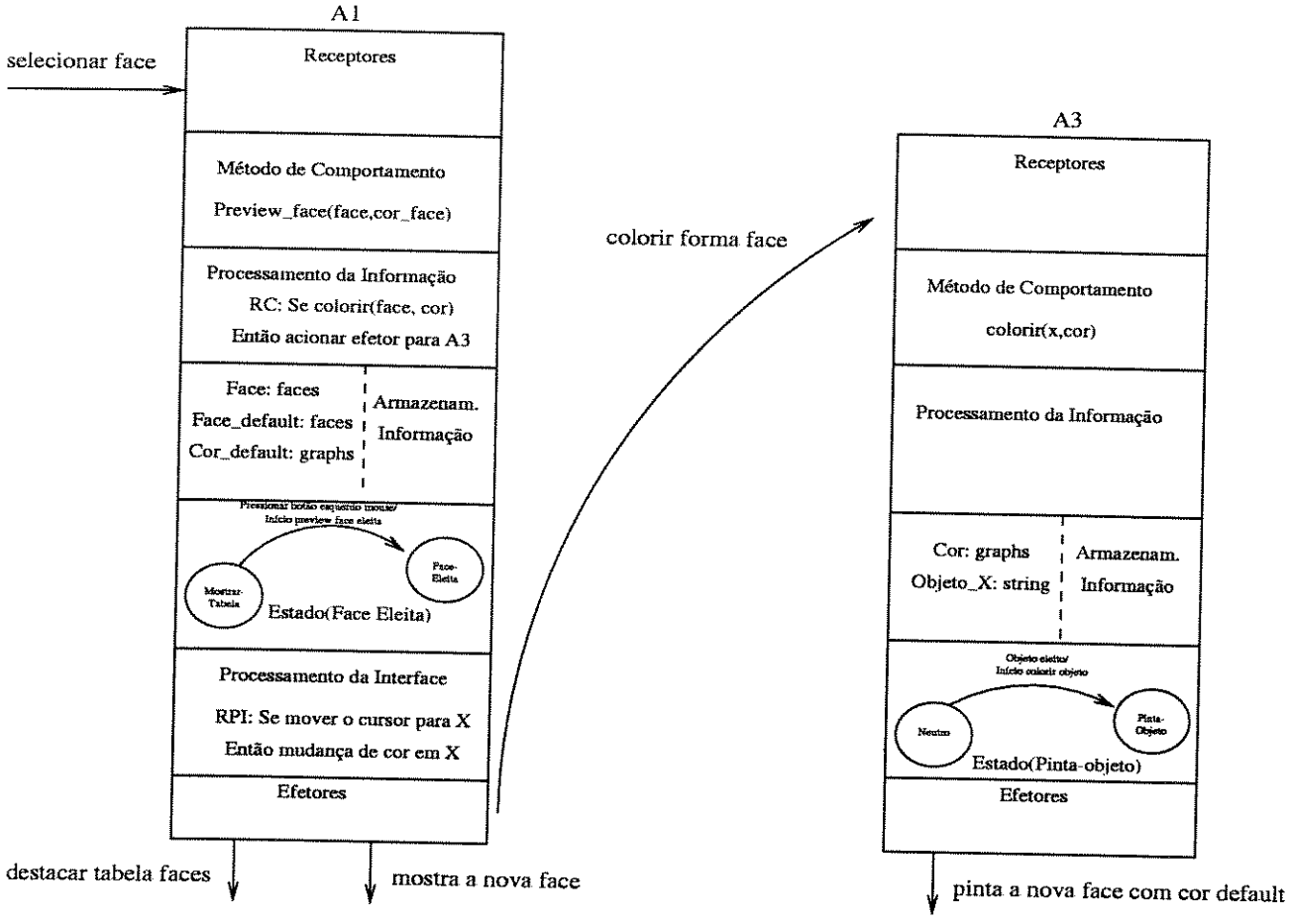

Figura 7.13: Tarefa 1: Escolher Forma da Face

Pela seguinte regra de processamento da informação:

RC: Se colorir(face, cor)
Então acionar efetor para A_3

é acionado uma chamada para A_3. O AIA A_3 recebe a chamada através do receptor (classe - selecionar cor da face) e aciona o método de comportamento colorir(x, cor).

O efetor de A_1 (classe - area preview face) mostra a nova face do relógio e o efetor de A_3 (classe - area preview face) pinta a nova face com a cor *default*. O estado do AIA A_1 passa de "mostrar-tabela" para "face-eleita". O estado do AIA A_3 passa de "neutro" para "pinta-objeto".

Na Figura 7.14, o AIA A_2 recebe uma ação do usuário através do receptor (classe - selecionar ponteiros tabela). O estado do AIA A_2 passa de "neutro" para "mostrar-tabela". O processamento da interface destaca a tabela de ponteiros (efetor da classe - tabela ponteiros). É acionado o método de comportamento preview_ponteiros, o qual faz chamadas à função desenha_ponteiros e colorir(ponteiros, cor). Pela seguinte regra de processamento da informação:

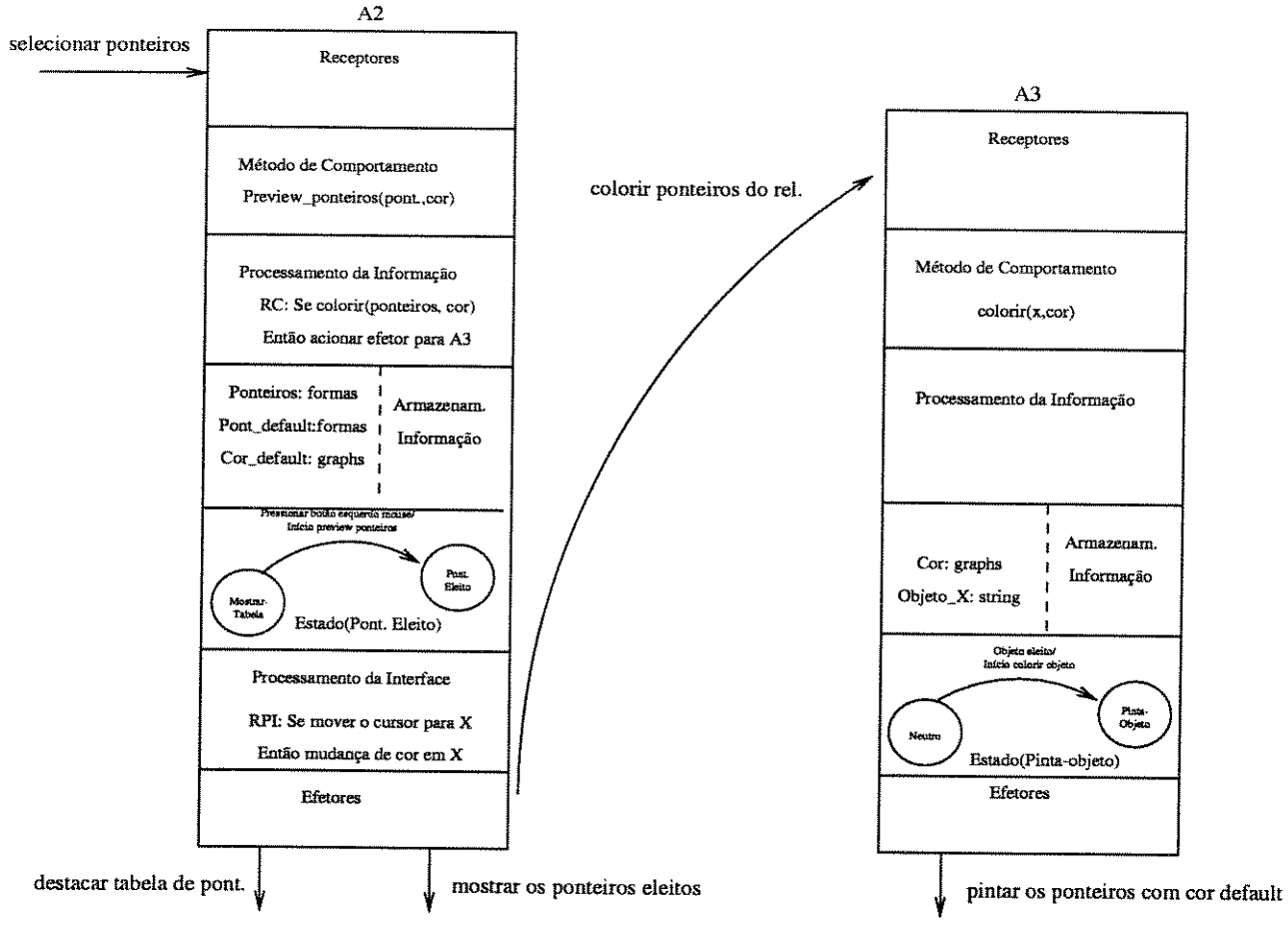

Figura 7.14: Tarefa 2: Escolher os Ponteiros do Relógio

RC: Se colorir(ponteiros, cor)
Então acionar efetor para A_3

é acionada uma chamada para A_3. O AIA A_3 recebe a chamada através do receptor (classe - selecionar cor dos ponteiros) e aciona o método de comportamento colorir(x, cor).

O efetor de A_2 (classe - area preview ponteiros) mostra os ponteiros selecionados e o efetor de A_3 (classe - area preview ponteiros) pinta os ponteiros com a cor *default*. O estado do AIA A_2 passa de "mostrar-tabela" para "ponteiro-eleito". O estado do AIA A_3 passa de "neutro" para "pinta-objeto".

Na Figura 7.15, o AIA A_3 recebe uma ação do usuário através do receptor (classe - selecionar cor dos ponteiros). O estado do AIA A_3 passa de "neutro" para "mostrar-cor-ponteiros". O efetor de A_3 (classe - tabela cores) destaca a cor.

É acionado o método de comportamento colorir(x, cor). O estado do AIA A_3 passa de "mostrar-cor-ponteiros" para "cor-eleita". O efetor de A_3 (classe - area preview ponteiros) pinta os ponteiros com a cor eleita.

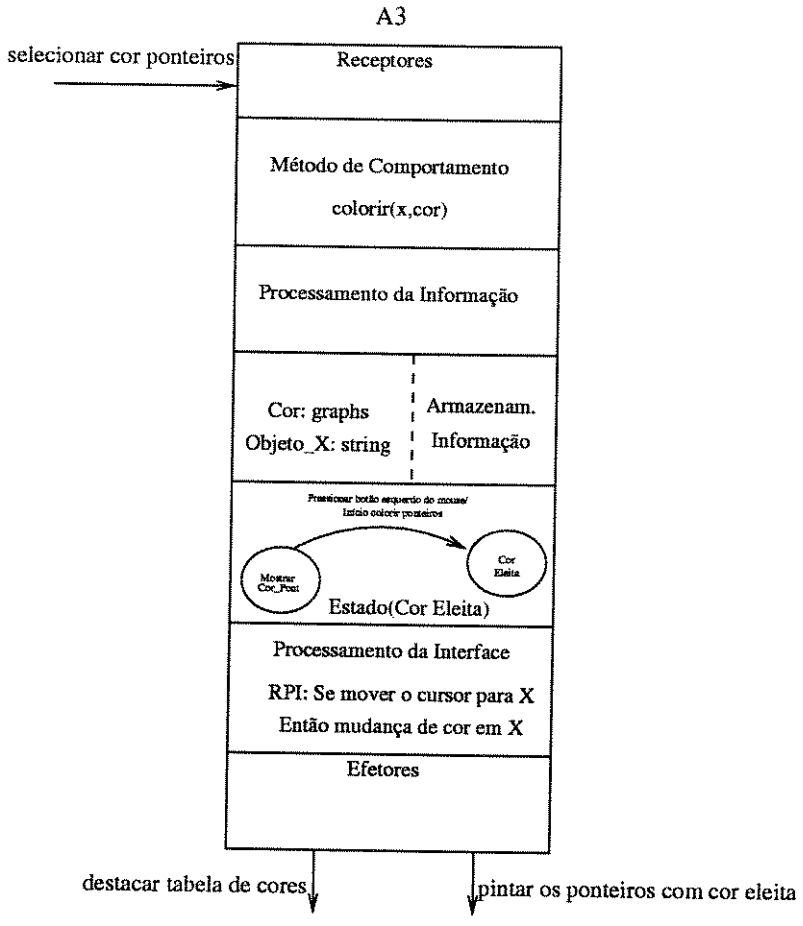

Figura 7.15: Tarefa 3: Mudar a Cor dos Ponteiros

7.13 Conclusão

Através deste exemplo foram apresentadas as diversas informações que são necessárias para a especificação completa de um projeto de sistema interativo. Os AIAs são constituídos a partir das informações contidas na idéia e se apresentam como uma estrutura que condensa a especificação do espaço interativo, do diálogo, das regras de seleção e controle do diálogo, junto com a especificação das funções, entradas/saídas, restrições e condições próprias do software.

As informações contidas na idéia e na descrição dos AIAs suportam o desenvolvimento sistemático das interfaces do usuário cobrindo todas as atividades de análise e projeto, não sempre tratadas ou indicadas em todas as metodologias. Outra vantagem é que o ambiente pode gerar modelos conforme diferentes metodologias, buscando na idéia as informações necessárias.

8

Conclusão

Devido à natureza multi-disciplinar da interação homem-computador, temos investigado abordagens de desenvolvimento de sistemas interativos que garantam a aceitação do software da interface por parte dos usuários. Estas abordagens de desenvolvimento visam, principalmente, permitir que pessoas que não possuam conhecimentos aprofundados de computação sejam capazes de projetar e contribuir com seu conhecimento ao desenvolvimento de interfaces.

Assim, foi definido um *framework* a fim de tornar mais fácil o desenvolvimento multi-disciplinar, isto é, um ambiente onde telas gráficas sejam projetadas com o auxílio do agente de prototipagem, engenheiros de fatores humanos trabalhem com o agente da interface, documentadores técnicos trabalhem com o agente de documentação, etc. Neste trabalho foi definida uma base conceitual onde diversos especialistas trabalham de maneira coordenada no desenvolvimento do projeto. A base conceitual é definida segundo a abordagem multi-agentes, formada por agentes da interface da aplicação (AIA). Tais agentes são definidos a partir das informações que permanecem armazenadas na idéia e que representam o resultado da análise, projeto e prototipagem da aplicação.

Neste trabalho, foram analisados diversos sistemas de apoio ao projeto de interfaces (UIDEs) correspondentes a diversas filosofias. Contudo, as ferramentas ainda precisam ser melhoradas. Elas não suportam aspectos do processo de projeto tal como análise dos usuários e modelamento de cenários, e são restritas para algumas metodologias de análise/projeto de interfaces homem-computador.

O ambiente proposto integra as diferentes metodologias existentes para o projeto de sistemas interativos a fim de produzir um projeto completo da interface. Ele fornece uma estrutura de desenvolvimento adequada para interfaces com uma abordagem de "ondas alternadas" durante o ciclo de vida, permitindo a avaliação nas fases sucessivas, seja por bases de *guidelines* ou feita por especialistas. Consideramos que é necessário estabelecer especificações de usabilidade junto da análise

das necessidades. Isto permite ter uma primeira avaliação de usabilidade já na fase de análise e evitar que problemas se propaguem para o protótipo.

A definição da arquitetura do ambiente foi feita para prover o fundamento do funcionamento e das ferramentas que possam ser construídas, assim como uma estrutura e forma universal para os AIAs. O paralelo com o sistema biológico facilita a identificação dos agentes como unidades autônomas, com um comportamento próprio (método de comportamento), com características como memorização (memória sensorial, etc.), comportamento reativo (processamento da interface), aprendizagem (memória de longo prazo), etc.

Outras vantagens do ambiente surgem da abordagem multi-agentes, como por exemplo a autonomia dos agentes, capacidade de interagir, de entrar ou sair do ambiente, paralelismo, repartição dos trabalhos entre os agentes, etc.

O ambiente formado pelos oito agentes básicos permite a identificação dos componentes abstratos da interface da aplicação, classificação psicológica dos usuários, especificação do controle do diálogo, do software da aplicação, das especificações de projeto formado pelos AIAs, e do espaço interativo.

Dentre as vantagens com relação ao ambiente definido podemos citar [Ari 97b]:

- Permite incorporar outras metodologias, como UAN, na descrição do projeto. Esta é uma vantagem em relação a outros formalismos como HIT [Sch 94], que não possui notação para descrever o diálogo;
- Combina boas características de várias técnicas, como a descrição dos métodos de GOMS, etc.

Dentre as vantagens com relação às descrições contidas na idéia, podemos citar [Ari 97b]:

- Incorpora a definição de tarefas, funções e classificação de usuários, descrições comumente usadas nas diversas metodologias;
- Possui estados que mostram o fluxo de controle do diálogo. Os estados estão associados com os AIAs o que facilita a compreensão por parte dos projetistas;
- Facilita a documentação como um todo e não unicamente dos aspectos implementados em computador. É possível encontrar a descrição das tarefas que o usuário deve desenvolver. Isto é, aquelas que são manuais e fazem parte das tarefas a serem desenvolvidas no domínio completo da aplicação.

Dentre as vantagens da descrição dos AIAs, podemos citar [Ari 97b]:

- Condensa, numa mesma estrutura, a especificação referente à interface e especificação funcional da aplicação. Os AIAs se apresentam como uma forma de resolver o *gap* entre a especificação da interface e da aplicação[1];

[1]Chamados comumente na literatura como "termos psicológicos" e "termos físicos" [Nor 86].

- Descreve aspectos concretos tais como o nome das funções, tipo de dados, e aspectos abstratos como a visão do agente que apresenta uma documentação das tarefas do AIA;

- Organiza e reúne diversas especificações, completando a especificação do projeto, facilitando sua modularização e servindo como guia para seu posterior desenvolvimento;

- Facilita a simulação manual do comportamento dos AIAs. O projeto da interface é considerado um conjunto de AIAs com suas partes bem definidas onde é possível fazer protótipos de papel e compreender o comportamento da interface, como vimos nas Figuras 7.13, 7.14 e 7.15, do capítulo anterior.

O que chamou a atenção no experimento do capítulo anterior foi a necessidade de estabelecer uma relação entre as diversas fases no desenvolvimento do projeto da interface. Muitos autores estabelecem [Hix 93] que estas fases devem ser coordenadas mas não indicam como isto é feito. Esta é uma das grandes vantagens deste trabalho.

8.1 Trabalhos Futuros

Como conseqüência da abrangência do tema, muitas questões associadas surgiram durante o desenvolvimento desta tese. Com relação ao ambiente destacamos as seguintes extensões:

1. Determinar a linguagem do ambiente a ser utilizada pelo agente especialista na gramática, que possa conter todo o significado das diferentes especificações na sua semântica, para permitir que os agentes heterogêneos possam se comunicar trocando informação sobre algum projeto na mesma linguagem;

2. Determinar um gerenciador de versões (agente gerenciador da idéia) para controle dos agentes que estão trabalhando no projeto;

3. Determinar o agente de documentação que possa ser configurado conforme determinadas características/exigências de documentação da aplicação particular;

4. Determinar algum protocolo de comunicação entre os agentes e como serão os comportamentos sociais entre eles (cohabitação, cooperação, colaboração, etc.);

5. Estudar a incorporação de ferramentas de prototipagem que possam fazer uma análise cognitiva simples do protótipo da interface a fim de avaliar a usabilidade do projeto;

6. Estudar como os agentes podem conter mecanismos, e os tipos dos mesmos, para comparar a usabilidade do projeto com as especificações. Identificar as regras de *guidelines* para validar as especificações;

7. Especificar uma ferramenta para a classificação psicológica dos usuários por parte de psicólogos ou especialistas em educação a fim de utilizar este conhecimento na validação das especificações;

8. Estudar a aplicação de *personality traits* no processo de interpretar quando os agentes podem atuar em uma solução, o que pode ser apresentado aos outros agentes, etc. Segundo Carbonell [Car 80] o conhecimento dos papéis e suas personalidades ajuda a interpretar suas ações e induzir seus objetivos. Isto facilita ter sistemas inteligentes capazes de alterar seu comportamento segundo seu *personality trait.*

Com relação à idéia destacamos as seguintes extensões:

1. Identificar as possíveis sociedades, isto é, sub-domínios. Os AIAs podem ser encapsulados em sociedades para facilitar a sua manipulação. A relação entre as sociedades está dada pelas relações temporais entre os AIAs;

2. Investigar a forma de reestruturar automaticamente a árvore dos AIAs, podendo gerar uma árvore diferente de tarefas baseado nos conceitos de independência, funcionalidade, ergonomia e estruturação;

3. Estudar a possiblidade de incrementar outras informações semânticas na idéia provenientes de ferramentas-agentes que possam ser incorporadas no futuro.

Com relação à área de interfaces destacamos as seguintes extensões:

1. Determinar como será o diálogo entre o agente *kernel* e o usuário. Este estudo inclue como se deve instruir e controlar o agente, como o agente aprende com os erros, o *feedback* do agente, e a maneira pela qual os agentes oferecem auxílio e informação ao usuário;

2. Determinar especificações de usabilidade para passos intermediários durante o projeto. Consideramos que certas características de usabilidade podem ser avaliadas em fases anteriores e assim diminuir os erros encontrados no projeto detalhado;

3. Formalizar, baseado no exemplo do capítulo 7, uma metodologia que englobe toda a informação necessária para a análise e projeto de interfaces.

Com relação aos AIAs destacamos as seguintes extensões:

1. Explorar a parte da memória sensorial da arquitetura com o objetivo de descobrir as suas facilidades na resolução dos problemas de UNDO/REDO.

Finalmente, pode ser aplicada a definição dos AIAs na especificação de bases de dados comerciais como ORACLE. A metodologia atual para especificação do software da interface não consegue relacionar, de forma satisfatória, a especificação de páginas WWW com o modelo entidade-relacionamento dos dados tratados pela página [Bar 92]. Os AIAs são uma abstração que representam a ligação entre a interface (objetos do espaço interativo) com o modelo de dados, o que resolve o *gap* entre a interface e o banco. É possível estudar como a estrutura dos AIAs pode ser colocada no banco de dados e utilizar a especificação dos AIAs para geração do código da interface.

Consideramos que os objetivos estabelecidos para esta tese foram alcançados e que as principais constribuições obtidas foram:

1. Estudo da multi-disciplinariedade do projeto, onde foi proposto um ambiente que integra as diversas disciplinas no desenvolvimento de sistemas interativos;

2. Estudo de diversas metodologias para o projeto de interfaces, onde foi proposta uma base conceitual que permite incluir as principais técnicas nas descrições do projeto;

3. Estudo do processo de desenvolvimento de sistemas interativos, onde foi necessário estabelecer as diversas fases do ciclo de vida e as conexões entre as fases, resultando assim, na descrição de uma metodologia que será formalizada em trabalhos futuros.

Os resultados obtidos fornecem base suficiente para trabalhos futuros que venham a implementar, validar e expandir o que aqui foi investigado.

Referências Bibliográficas

[Ari 96] Arias, C. & Daltrini, B., A Multi-Agent Environment for User Interface Design, *Euromicro 96: 22nd Euromicro Conference, beyond 2000 Hardware and Software Design Strategies*, IEEE Computer Society Press, Praga, Czech Republic, Pg. 242-247, 1996.

[Ari 97a] Arias, C. & Daltrini, B., User-Interface Specification with Multi-Agent Systems, Submetido para publicação ao *Communications of the ACM*, ACM, Inc., 1997.

[Ari 97b] Arias, C. & Daltrini, B., Multi-Agent System for the User Interface Design Process, *19th International Conference on Information Technology Interfaces*, ITI 97, Conference Proceedings, Kalpić D. & Dobrić V. (Eds), Pula, Croatia, Pg. 221-226, 1997.

[Ari 97c] Arias, C. & Daltrini, B., Aspectos do Projeto de Interfaces Considerados por um Framework Multi-disciplinar, Submetido para *XI Brazilian Symposium on Software Engineering*, SBES 97, Fortaleza, Brasil, 1997.

[Ari 97d] Arias, C. & Daltrini, B., Diseño de la Interfaz: Modelización de Tareas, Submetido para *XVII International Conference of the Chilean Computer Science Society*, Valparaiso, Chile, 1997.

[Ari 98] Arias, C. & Daltrini, B., Un Ambiente para el Desarrollo de la Interfaz: Descripción de su Arquitectura Interna, *Revista Internacional Información Tecnológica*, La Serena, Chile, Vol. 9, Nro. 1, 1998.

[Bae 87] Baecker, R. & Buxton, W., *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, Morgan Kaufmann Publishers Inc., California, 1987.

[Bar 92] Barker, R. & Longman, C., *CASE*METHOD: Entity Relationship Modelling*, Addison-Wesley Publishing Company, Oracle Corporation UK Limited, 1992.

[Bar 87] Barnard, P., Cognitive Resources and the Learning of Human-Computer Dialogues, *Interfacing Thought: Cognitive Aspects of*

Human-Computer Interaction, Carroll J. (Ed), MIT Press, Pg. 112-158, 1987.

[Bar 81] Barr, A. & Feigenbaum, E., *The HandBook of Artificial Intelligence*, Addison-Wesley, Inc., Massachusetts, Vol. I e II, 1981.

[Bat 94] Bates, J., The Role of Emotion in Believable Agents, *Communications of the ACM*, ACM, Inc., Vol. 37, Nro. 7, Pg. 122-125, 1994.

[Ber 92] Berthet, S., Demazeau, Y. & Boissier, O., Knowing Each Other Better, *11th International Workshop on Distributed Artificial Intelligence '92*, Glenn Arbor, Michigan, Pg. 23-41, 1992.

[Boi 92] Boissier, O. & Demazeau, Y., A Distributed Artificial Intelligence View on General Purpose Vision Systems, *Decentralized Artificial Intelligence*, Demazeau Y. & Werner E. (Eds), North-Holland, Elsevier, Vol. 3, Pg. 311-330, 1992.

[Bro 89] Brown, C., *Human-Computer Interface Design Guidelines*, Ablex Publishing Corporation, New Jersey, 2nd. ed., 1989.

[Bur 95] Burmeister, B., Haddadi, A. & Sundermeyer, K., Generic, Configurable, Cooperation Protocols for Multi-Agent Systems, *Lecture Notes in Artificial Intelligence*, From Reaction to Cognition, Castelfranchi C. & Müller J. (Eds.), Springer-Verlang, Vol. 957, Pg. 157-230, 1995.

[Car 80] Carbonell, J., Towards a Process Model of Human Personality Traits, *Artificial Intelligence*, North-Holland Publishing Company, Vol. 15, Nro. 1,2, Pg. 49-74, 1980.

[Car 83] Card, S., Moran, T. & Newell A., *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates Publishers, Hillsdale, NJ, 1983.

[Chu 93] Chul, Kim W. & Foley, J., Providing High-level Control and Expert Assistance in User Interface Presentation Design, *Conference on Human Factors in Computing Systems*, INTERCHI, Conference Proceedings, Amsterdam, Pg. 430-437, 1993.

[Cou 91] Coutaz, J., Interfaces Homme-Machine: un Regard Critique, *Technique et Science Informatiques*, Editions Dunod, Vol. 10, Nro. 1, Pg. 53-64, 1991.

[Cou 95] Coutaz, J., Evaluation Techniques: Exploring the Intersection of HCI and Software Engineering, *Lecture Notes in Computer Science*, Software Engineering and Human-Computer Interaction, Taylor R. & Coutaz J. (Eds.), Springer-Verlag Berlin, Vol. 896, Pg. 35-48, 1995.

[Cro 88] Crowston, K. & Malone, T., Intelligent Software Agents, *BYTE*, Vol.13, Nro.13, Pg. 267-271, 1988.

[DeS 89] DeSoi, J., Lively, W. & Sheppard, S., Graphical Specification of User Interfaces with Behavior Abstraction, *Human Factors in Computing Systems*, CHI 89, Conference Proceedings, Bice K. & Lewis C. (Eds), ACM Press, Austin, Texas, Pg. 139-144, 1989.

[Dun 95] Dunin-Keplicz, B & Treur, J., Compositional Formal Specification of Multi-Agent Systems, *Lecture Notes in Artificial Intelligence*, Intelligent Agents: Theories, Architectures and Languages, Springer-Verlag, Vol. 890, Pg. 102-117, 1995.

[Edm 94] Edmonds, E., Candy, L., Jones, R. & Soufi, B., Support for Collaborative Design: Agents and Emergence, *Communication of the ACM*, ACM, Inc., Vol. 37, Nro. 7, Pg. 41-47, 1994.

[Etz 94] Etzioni, O. & Weld, D., A Software-Based Interface to the Internet, *Communications of the ACM*, ACM, Inc., Vol. 37, Nro. 7, Pg. 72-76, 1994.

[Gen 94] Genesereth, M. & Ketchpel, S., Software Agents, *Communications of the ACM*, ACM, Inc., Vol. 37, Nro. 7, Pg. 49-53,147, 1994.

[Gru 89] Grudin, J. & Poltrock, S., User Interface Design in Large Corporations: Coordination and Communication Across Disciplines, *Human Factors in Computing Systems*, CHI 89, Conference Proceedings, Bice K. & Lewis C. (Eds), ACM Press, Austin, Texas, Pg. 197-203, 1989.

[Guh 94] Guha, R. & Lenat, D., Enabling Agents to Work Together, *Communications of the ACM*, ACM, Inc., Vol. 37, Nro. 7, Pg. 127-142, 1994.

[Gut 89] Gutierrez, O., Prototyping Techniques for Different Problem Contexts, *Human Factors in Computing Systems*, CHI 89, Conference Proceedings, Bice K. & Lewis C. (Eds), ACM Press, Austin, Texas, Pg. 259-263, 1989.

[Guy 86] Guyton, A., *Tratado de Fisiologia Médica*, Editora Guanabara S.A., 6ta. ed., 1986.

[Har 87] Harel, D., Statecharts: A Visual Formalism for Complex Systems, *Science of Computer Programming*, Vol. 8, Nro. 3, Pg. 231-274, 1987.

[Har 95] Harrison, M. & Duke, D., A Review of Formalisms for Describing Interactive Behaviour, *Lecture Notes in Computer Science*, Software Engineering and Human-Computer Interaction, Taylor R. & Coutaz J. (Eds.), Springer-Verlag Berlin, Vol. 896, Pg. 49-75, 1995.

[Har 89] Hartson, H., User-Interface Management Control and Communication, *IEEE Software*, Vol. 6, Nro. 1, Pg. 62-70, 1989.

[Hix 93] Hix, D. & Hartson, H., *Developing User Interfaces: Ensuring Usability Through Product and Process*, John Wiley & Sons, Inc., USA, 1993.

[Hix 94] Hix, D. & Hartson, H., IDEAL: An Environment to Support Usability Engineering, *Lecture Notes in Computer Science*, Human-Computer Interaction, Blumenthal B., Gornostaev J. & Unger C. (Eds.), Springer-Verlag Berlin Heidelberg, Vol. 876, Pg. 95-106, 1994.

[Hua 95] Huang, J., Jennings, N. & Fox, J., An Agent Architecture for Distributed Medical Care, *Lecture Notes in Artificial Intelligence*, Intelligent Agents: Theories, Architectures and Languages, Springer-Verlag, Vol. 890, Pg. 219-232, 1995.

[Hüb 95] Hübner, J., *Migração de Agentes em Sistemas Multi-Agentes Abertos*, Dissertação de Mestrado em Ciência da Computação, Universidade Federal de Rio Grande do Sul, UFRGS, 1995.

[Ish 94] Ishida, T., Parallel, Distributed and Multiagent Production Systems, *Lecture Notes in Artificial Intelligence*, Parallel Distributed and Multiagent Production System, Springer-Verlag, Vol. 878, Pg. 61-85, 1994.

[Jan 93] Janssen, C., Weisbecker, A. & Ziegler, J., Generating User Interfaces from Data Models and Dialogue Net Specification, *Conference on Human Factors in Computing Systems*, INTERCHI, Conference Proceedings, Amsterdam, Pg. 418-423, 1993.

[Jen 95] Jennings, N., Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions, *Artificial Intelligence*, Elsevier Science B. U., Vol. 75, Nro. 2 Pg. 195-240, 1995.

[Joh 95] Johannsen, G., Knowledge-Based Design of Human-Machine Interfaces, *Control Eng. Practice*, Elsevier Science Ltd., UK, Vol. 3, Nro. 2, Pg. 267-273, 1995.

[Joh 92] Johnson, P., *Human-Computer Interaction: Psycology, Task Analysis and Software Engineering*, McGraw-Hill Book Company International Ltd., London, 1992.

[Kie 85] Kieras, D. & Polson, P., An Approach to the Formal Analysis of User Complexity, *International Journal of Man-Machine Studies*, Vol. 22, Pg. 365-394, 1985.

[Kin 95] King, J., Intelligent Agents, *AI Expert*, Miller Freeman Inc., Vol. 10, Nro. 3, 1995.

[Lau 86] Laurel, B., Interface as Mimesis, *User Centered System Design: New Perspectives on Human-Computer Interaction*, Norman D., & Draper S. (Eds.), Lawrence Erlbaum Associates Publishers, Hillsdale, NJ, Pg. 67-85, 1986.

[Lau 90] Laurel, B., *The Art of Human-Computer Interface Design*, Laurel B. (Ed.), Addison-Wesley Publishing Company, Inc., Massachusetts, 1990.

[Luc 92] Lucena, F. *Construção de Interfaces Homem-Computador: o uso de Estadosgramas na Especificação e Implementação de Controle de Interface*, Dissertação de Mestrado em Ciência da Computação, Universidade Estadual de Campinas, UNICAMP, 1992.

[Mae 94] Maes, P., Agents that Reduce Work and Information Overload, *Communications of the ACM*, ACM, Inc., Vol. 37, Nro. 7, Pg. 31-40,146, 1994.

[Mor 81] Moran, T., The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems, *International Journal of Man-Machine Studies*, Vol. 15, Nro. 1, Pg. 3-50, 1981.

[Mul 95] Müller, J., Pischel, M. & Thiel, M., Modeling Reactive Behavior in Vertically Layared Agent Architectures, *Lecture Notes in Artificial Intelligence*, Intelligent Agents: Theories, Architectures and Languages, Springer-Verlag, Vol. 890, Pg. 262-276, 1995.

[Mye 89] Myers, B., Encapsulating Interactive Behaviors, *Human Factors in Computing Systems*, CHI 89, Conference Proceedings, Bice K. & Lewis C. (Eds), ACM Press, Austin, Texas, Pg. 319-324, 1989.

[Mye 95] Myers, B., User Interface Software Tools, *ACM Transactions on Computer-Human Interaction*, ACM Press, Vol. 2, Nro. 1, Pg. 64-103 1995.

[Mye 90] Myers, B., Giuse, D., Dannenberg, R., Vander Zanden, B., Kosbie, D., Pervin, E., Mickish, A. & Marchall, P., Garnet: Comprehensive Support For Graphical, Highly-Interactive Users Interfaces, *IEEE Computer*, Vol 23, Nro 11, Pg. 71-85, 1990.

[New 95] Newman ,W. & Lamming, M., *Interactive System Design*, Addison-Wesley Publishing Company, Inc., UK, 1995.

[Nie 92] Nielsen, J., The Usability Engineering Life Cycle, *IEEE Computer*, Nro. 3, Pg. 12-22, 1992.

[Nor 94] Norman, D., How Might People Interact with Agents, *Communications of the ACM*, ACM, Inc., Vol. 37, Nro. 7, Pg. 68-71, 1994.

[Nor 86] Norman, D., Cognitive Engineering, *User Centered System Design: New Perspectives on Human-Computer Interaction*, Norman D., & Draper S. (Eds.), Lawrence Erlbaum Associates Publishers, Hillsdale, NJ, Pg. 31-61, 1986.

[Oli 93] Oliveira, E. & Mouta, F., A Distributed AI Architecture Enabling Multi-Agent Cooperation, *Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Conference Proceedings, Edinburg, Scotland, 1993.

[Oli 93a] Oliveira, E., Mouta, F. & Rocha, A., Negotiation and Conflict Resolution within a Community of Cooperative Agents, *International Symposium on Automous Decentralized Systems*, Conference Proceedings, Kawasaki, Japan, 1993.

[Oli 93b] Oliveira, E., Mouta, F. & Rocha, A., Cooperation in a Multi-Agent Community, *XIII International Conference on Artificial Intelligence, Expert Systems and Natural Language*, Conference Proceedings, Avignon, France, 1993.

[Pay 86] Payne, S. & Green, T., Task-Action Grammars: A Model of Mental Representation of Task Languages, *Human-Computer Interaction*, Vol. 2, Pg. 93-133, 1986.

[Pow 90] Powell, J., *Designing User Interfaces*, Lance A. Leventhal, Ph.D., 1990.

[Pre 92] Pressman, R., *Software Engineering a Practitioner's Approach*, McGraw-Hill, Inc., New York, 3ra. ed., 1992.

[Rei 81] Reisner, P., Formal Grammar and Human Factors Design of an Interactive Graphics System, *IEEE Transaction on Software Engineering*, Vol. SE-7, Nro. 2, Pg. 229-240, 1981.

[Rie 94] Riecken, D., M: An Architecture of Integrated Agents, *Communications of the ACM*, ACM, Inc., Vol. 37, Nro. 7, Pg. 107-116, 1994.

[Sch 94] Schreiber, S., Specification and Generation of User Interfaces with the BOSS-System, *Lecture Notes in Computer Science*, Human-Computer Interaction, Blumenthal B., Gornostaev J. & Unger C. (Eds.), Springer-Verlag Berlin Heidelberg, Vol. 876, Pg. 107-120, 1994.

[Shn 92] Shneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley Publishing Company, Massachusetts, 2da ed., 1992.

[Sho 92] Shoham, Y., Agent Oriented Programming: An Overview of the Framework and Summary of Recent Research, *11th International Workshop on Distributed Artificial Intelligence '92*, Glenn Arbor, Michigan, Pg. 345-353, 1992.

[Sic 92] Sichman, J., Demazeau, Y. & Boissier, O., When can Knowledge-Based Systems can be Called Agents?, *IX Simpósio Brasileiro de Inteligência Artificial*, SBIA, Rio de Janeiro, Brasil, Pg. 172-185, 1992.

[Sil 95] Silva, E. & Daltrini, B., *Representação em Projeto de Interfaces Homem-Computador: Estudo, Aplicação e Propostas de Estensão do Formalismo UAN*, Dissertação de Mestrado em Engenharia Elétrica, Universidade Estadual de Campinas, UNICAMP, 1995.

[Tay 95] Taylor, R., Nies, K., Alan Bolcer, G., McFarlane, C., Anderson, K. & Johnson, G., Chiron-1: A Software Architecture for User Interface Development, Maintenance, and Run-Time Support, *ACM Transacions on Computer-Human Interaction*, ACM Press, Vol. 2, Nro. 2, 1995.

[Thi 90] Thimbleby H., *User Interface Design*, ACM Press Frontier Series, Wokingham, England, Addison-Wesley Publishing Company, 1990.

[Van 93] Vanderdonckt, J. & Bodart, F., Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection, *Conference on Human Factors in Computing Systems*, INTERCHI, Conference Proceedings, Amsterdam, Pg. 424-429, 1993.

[Was 85] Wasserman, I., Extending State Diagrams for the Specification of Human-Computer Interaction, *IEEE Transaction on Software Engineering*, Vol. 11, Nro. 8, Pg. 699-713, 1985.

[Woo 95] Wooldridge, M. & Jennings, N., Agent Theories, Architectures, and Languages: a Survey, *Lecture Notes in Artificial Intelligence*, Intelligent Agents: Theories, Architectures and Languages, Springer-Verlag, Vol. 890, Pg. 2-39, 1995.

[You 89] Young, R., Green, T. & Simon, T., Programmable User Models for Predictive Evaluation of Interface Designs, *Human Factors in Computing Systems*, CHI 89, Conference Proceedings, Bice K. & Lewis C. (Eds), ACM Press, Austin, Texas, Pg. 15-19, 1989.