PUC RIO

Cecília Kremer Vieira da Cunha

Planejador de Respostas Explicativas Baseado em uma Biblioteca de Esquemas RST

DISSERTAÇÃO DE MESTRADO
Departamento de Informática
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Rio de Janeiro, 5 de Maio de 1997

Rua Marquês de São Vicente, 225 – Gávea
CEP 22453-900 Rio de Janeiro RJ Brasil
http://www.puc-rio.br

Sumário

Lista de Figuras

Lista de Abreviações

EDGE → *Explanatory Discourse Generator*

EES → *Explainable Expert System*

LN → Linguagem Natural

PEA → *Program Enhancement Advisor*

RST → *Rhetorical Structure Theory*

SBC → Sistema Baseado em Conhecimento

1. Introdução

A utilização de sistemas baseados em conhecimento trouxe para a área de design de interfaces o desafio de esclarecer ao usuário o conhecimento previsto pelo sistema. Conhecimento corresponde à informação processada, assim como à forma como ela está representada e como é aplicada (possibilitando inferências e expectativas) para atender aos objetivos finais do usuário [Stefik'95]. Este desafio é de profunda importância no sentido de garantir a usabilidade de um sistema baseado em conhecimento.

Estudos têm sido realizados neste sentido, buscando adaptar sistemas baseados em conhecimento (SBCs) de forma que eles não sobrecarreguem um usuário que tente entender a complexidade inerente ao funcionamento dos mesmos. Surgem então perguntas cruciais a serem respondidas, tais como: Que informações devem estar disponíveis a nível de interface? Quão tutorial o sistema deve ser? Qual o equilíbrio entre informação necessária e quantidade de informação? Que informações adicionais precisam estar representadas na base de conhecimento de forma a atender as necessidades de interface? Quando e como elas devem aparecer? Quanto o contexto de uso deve influenciar o conteúdo e a forma da informação apresentada?

SBCs que têm por objetivo auxiliar usuários a realizar tarefas, sem substituí-los, geralmente possuem um módulo de Explicação. Este módulo é responsável por permitir que usuários entendam o conhecimento usado pelo sistema. Para isto módulos de Explicação normalmente oferecem recursos de interação de forma que um usuário possa realizar perguntas. As respostas para estas perguntas são geradas automaticamente pelo módulo de Explicação. Elas são na verdade uma exposição amigável do que se passa no sistema. A manutenção da fidelidade entre o modelo utilizado pelo sistema e o apresentado ao usuário através das respostas explicativas pode aumentar a usabilidade do sistema e servir como tutorial para um futuro processo de aquisição de conhecimento direto com o usuário. Além disso, é possível também explicitar erros que tenham passado despercebidos no desenvolvimento da aplicação.

O objetivo desta dissertação é propor um Planejador de Respostas Explicativas Baseado em uma Biblioteca de Esquemas RST. A abrangência deste trabalho será descrita nas próximas seções.

1.1 Requisitos do Planejador

Com o objetivo de delimitar o escopo do Planejador de Respostas Explicativas, foi feito o levantamento de quais seriam seus requisitos. Tomamos como ponto de partida os requisitos para um bom sistema de Explicação descritos em [Moore'95], que, para sustentar um sistema de Explicação que possibilite diálogo, são os seguintes:

1. ***Naturalidade.*** Explicações devem ser estruturadas de acordo com padrões de discurso humanos. Ou seja, o sistema deve possuir conhecimento sobre estruturas de discurso e saber aplicar este conhecimento.
2. ***Reatividade.*** O sistema de explicação deve ser capaz de aceitar *feedbacks* do usuário sobre suas explicações, responder perguntas subsequentes ou oferecer explicações alternativas, se o usuário não se mostrar satisfeito. Além disso o usuário deve poder fazer perguntas não muito bem formuladas.
3. ***Flexibilidade.*** Devem existir várias estratégias para se produzir uma explicação - responder a uma pergunta - para os casos em que o usuário não é capaz de entendê-la e para ela poder ser sensível a contexto (conhecimento ou objetivos do usuário).
4. ***Sensibilidade.*** Moore cita a contribuição do trabalho de [Swartout'90] para este e os próximos dois itens. Este item prevê o uso do conceito de contexto - conhecimento ou objetivos do usuário, momento da resolução de um problema ou diálogo precedente. Os algoritmos que produzem explicações devem atender aos seguintes requisitos:
5. ***Fidelidade.*** Uma explicação deve refletir o conhecimento e raciocínio do sistema.
6. ***Suficiência.*** O sistema deve ser capaz de responder ao conjunto de questões que o usuário deseja fazer. Conhecimento e estratégias de raciocínio do sistema devem estar disponíveis para a explicação.
7. ***Extensibilidade.*** A explicação deve ser capaz de responder vários tipos de perguntas, algumas que podem não ter sido previstas na fase de design. O sistema deve poder ser facilmente estendido para responder a novas perguntas.

Em geral sistemas de Explicação disponibilizam um conjunto de perguntas possíveis para o usuário. Ele escolhe a pergunta que deseja fazer e o sistema apresenta a resposta explicativa. O Planejador aqui proposto tem como objetivo planejar uma resposta para uma pergunta de um usuário de um sistema baseado em conhecimento. Como uma abordagem inicial restringimos seu escopo e não é sua função participar de um diálogo continuado com o usuário.

A sua incapacidade de realizar um verdadeiro diálogo é proveniente de um desequilíbrio entre o poder expressivo do gerador de perguntas atual e do gerador de respostas. Entendemos que um diálogo legítimo é um processo de negociação de modelos mentais entre dois agentes comunicantes. Mas, a interação homem-computador pode ser vista como uma comunicação assíncrona entre o usuário e o desenvolvedor do sistema tendo como forma de expressão a linguagem da interface [de Souza'93]. No âmbito do sistema em que concebemos e desenvolvemos este trabalho [Garcia'95], a linguagem disponível para o usuário é bem menos expressiva do que a utilizada pelo gerador de respostas, que conta com a linguagem natural e gráfica. O processo de diálogo será possível quando melhorada a relação de co-expressividade entre as linguagens de entrada e de saída.

Com esta limitação de diálogo, os requisitos para o Planejador proposto constituem um subconjunto dos supracitados. São eles: naturalidade, flexibilidade, sensibilidade quanto ao ponto de resolução de um problema do domínio, fidelidade, suficiência e extensibilidade.

O Planejador proposto produzirá um plano de resposta em linguagem natural (LN) para uma pergunta realizada. Com isso garantimos que a forma de expressão utilizada para veicular os conteúdos da aplicação é a menos restritiva possível [Nadin'88, Brennan'90]. No entanto, nem sempre a LN é o código mais eficiente para se transmitir uma mensagem, o que faz com que muitas vezes recorramos a gestos ou desenhos para nos expressar. O aspecto determinante da eficiência na comunicação está associado ao consumo de memória de trabalho do usuário durante o processo de compreensão de mensagens. Sabemos também que quanto maior for a utilização da memória de trabalho, menores serão os recursos disponíveis para o

processamento do restante da mensagem [Daneman&Carpenter'80, Baddely'86, Scott&de Souza'90a]. Veja um exemplo onde uma linguagem gráfica pode ser mais eficiente do que a LN:

> Avalie-se este trecho que descreve para um ouvinte, ao telefone, o trajeto entre o ponto onde ele está e aquele onde quer chegar:
> "Saindo de onde você está, vire na segunda rua à esquerda, depois na próxima à direita, seguindo em frente e virando na terceira direita, onde você verá uma farmácia na esquina. Depois, vire à esquerda, à direita e então à esquerda novamente. Você estará então na rua da minha casa que fica na próxima esquina, do lado direito."

O uso da LN neste caso exige do receptor da mensagem um alto grau de memorização. Já a linguagem gráfica do desenho abaixo sintetiza a sequência de instruções do falante em papel (memória auxiliar).

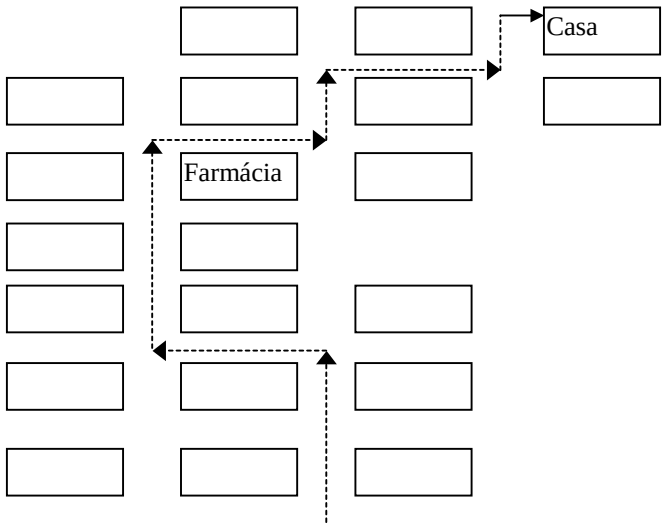


Figura 1: Utilização de Linguagem Gráfica

Existem domínios cujas representações se adequam mais a uma linguagem gráfica [Ioerger'94]. Por isso, o Planejador poderá também comandar outros recursos de interface que estejam disponíveis tornando-os coerentes e cooperativos com o conteúdo do texto em LN.

Assim a resposta gerada tira vantagem de uma multimodalidade de interface que possa estar presente [Maybury'93].

Como requisitos para as respostas explicativas a serem geradas, nossa fonte de estudo foi o trabalho de [Paris'91]. Segundo a autora uma resposta deve ser:

- ***Informativa.*** Ela deve conter informações que o usuário desconhece.
- ***Coerente.*** Ela deve estar organizada de forma coerente.
- ***Compreensível.*** Ela deve usar termos que o usuário conhece e conter informações que ele seja capaz de assimilar.
- ***Relevante.*** Ela deve prover informações que ajudarão os usuários a atingirem seus objetivos.
- ***Correta quanto ao registro e tom conversacional.*** O tom da resposta não deve parecer frio, severo, sarcástico ou desagradável.

Como este Planejador não possui modelo de usuário, não é possível satisfazer ao primeiro requisito, no entanto todos os outros foram perseguidos. Os motivos pelo qual o Planejador não contempla modelo de usuário serão expostos na seção 2.2 (Semiótica Computacional) e na seção 2.3 (Sistemas de Explicação).

1.2 Resumo do Trabalho

Para atender aos requisitos do Planejador foram estudadas as áreas de Design de Sistemas Centrado no Usuário, Semiótica Computacional, Sistemas de Explicação, *Rhetorical Structure Theory* e Geração de Linguagem Natural.

Estes estudos nos levaram a propor o Planejador detalhado no capítulo 3 (Solução). Em linhas gerais podemos dizer que o Planejador, a partir de uma pergunta de um usuário, consegue planejar uma resposta explicativa que confere organização e estruturação retórica ao conteúdo da mensagem que se quer transmitir. A principal forma de expressão utilizada é a linguagem

natural que pode ser apoiada por outros recursos expressivos. A implementação atual do Planejador prevê a utilização de recursos gráficos.

O desenvolvimento desta dissertação foi acompanhado do desenvolvimento de um sistema real baseado em conhecimento, que contribuiu para demonstrar que nossa solução é computável. Os trabalhos [de Souza&Garcia'94, Garcia&de Souza'95 e Garcia&de Souza'97] detectaram a necessidade e possíveis benefícios de se adicionar estruturação e organização retórica às respostas explicativas do módulo de Explicação de um *Active Design Document* (ADD) [Garcia'92].

ADD é um modelo para desenvolvimento de sistemas inteligentes que auxiliam um processo de design gerando simultaneamente sua documentação. A documentação gerada consiste não só dos dados do projeto mas também do *rationale* do designer.

Um documento ativo de design contém um modelo inicial do artefato a ser desenvolvido e é capaz de gerar expectativas sobre as decisões do usuário. Ele acompanha os passos do designer e sempre que estes atendem suas expectativas o ADD consegue gerar automaticamente explicações para os mesmos. Assim o ADD consegue documentar as decisões de projeto sem nenhuma sobrecarga para o designer. No entanto, nem sempre as expectativas do sistema são atendidas. Neste caso, o ADD solicita ao designer que forneça, ele mesmo, as bases de explicação para a sua decisão. Isto pode ser feito de duas formas: ou atualizando o conhecimento da base do ADD, o que permite que a decisão seja posteriormente explicada pelo próprio sistema; ou fazendo uma anotação em texto livre. Desta forma, como a decisão não possuirá uma explicação automática, o sistema se atém a mostrar a nota escrita pelo designer, no lugar de uma explicação completa.

O modelo ADD trata trabalhos individuais. Recentemente foi desenvolvida uma extensão a este modelo para tratamento de design cooperativo em grupo. Esta extensão corresponde ao MultiADD [Vivacqua&Garcia'96a, Vivacqua&Garcia'96b], uma arquitetura multiagente e

multiusuário sobre a qual foi desenvolvida a aplicação ADDProc[1]. O Planejador proposto se insere no desenvolvimento desta aplicação que foi desenvolvida para a Petrobrás (CENPES) através de um projeto do [ADDLabs].

O Planejador se encontra atualmente em fase de testes. Algumas avaliações de ordem subjetiva já foram obtidas e estão descritas no capítulo 3 (Solução). Uma discussão do trabalho é realizada no capítulo 5 (Discussão) e os trabalhos futuros de continuidade ao aqui proposto estão previstos no capítulo 6 (Conclusão).

1.3 Apresentação do Trabalho

Esta dissertação será apresentada em mais 5 capítulos, organizados da seguinte forma:

- O Capítulo 2 apresenta o estudo realizado sobre trabalhos relacionados que caracterizam a perspectiva adotada neste trabalho.

- O Capítulo 3 explica a solução adotada.

- O Capítulo 4 descreve a aplicação desta solução em um sistema real baseado em conhecimento.

- O Capítulo 5 faz uma avaliação do Planejador proposto discutindo resultados.

- Finalmente, o Capítulo 6 conclui a dissertação e aponta para os trabalhos futuros previstos.

[1]O ADDProc é um produto desenvolvido pelo ADDLabs [ADDLabs]. Sua proposta e especificação encontram-se documentadas nos relatórios de projeto, dentre os quais destacamos [Garcia'95].

2. Referenciais Teóricos

2.1 Design de Sistemas Centrado no Usuário

A área de Design de Sistemas Centrado no Usuário [Norman&Draper'86] contribuiu para um maior entendimento sobre como se dá a interação homem-computador e os processos mentais envolvidos nas ações e reações dos usuários. A partir desta compreensão é possível desenvolver interfaces mais satisfatórias.

Como parte do módulo de explicação o Planejador está relacionado à interface que será provida, pois é ele o responsável pela determinação: do conteúdo e da forma da resposta em LN a ser gerada; do relacionamento desta resposta com os outros recursos de interface disponíveis; assim como da atualização destes últimos.

[Kammersgaard'88] apresentou quatro possíveis perspectivas sobre a interação homem-computador: *systems perspective*, *dialogue partner perspective*, *tool perspective* e *media perspective*. Na primeira perspectiva toda interação é vista como transmissão de dados entre um humano e um computador. A maior preocupação consiste em tornar esta transmissão efetiva e eficiente, ou seja possibilitar que o usuário aja de forma similar aos componentes automáticos, como se fosse mais uma engrenagem da máquina.

A *dialogue partner perspective* corresponde ao oposto, onde humanos e computadores são vistos como parceiros em um diálogo. A interação é vista como uma comunicação entre os dois onde o computador deve ser capaz de mostrar comportamento comunicativo e cognitivo similar ao do seu parceiro humano.

As outras duas perspectivas influenciaram o desenvolvimento do Planejador. O Planejador se encaixa na *media perspective* no sentido de que o computador é um meio de comunicação entre o desenvolvedor da aplicação e seus usuários. Já focando na interação usuário-computador (e não desenvolvedor-usuário), o Planejador pode ser visto como uma ferramenta que auxilia o usuário a executar uma tarefa ("*tool perspective*"). As respostas explicativas colaboram com o usuário na realização das seguintes atividades:

- validação do sistema. As respostas explicativas retratam fielmente o modelo subjacente e como consequência podem refletir erros nos mesmos. Isto possibilita que o usuário confirme ou não a validade do sistema.
- uso do sistema. As respostas explicativas têm a função de esclarecer para o usuário o funcionamento do sistema. Um usuário que compreende o sistema que utiliza é capaz de aproveitar toda a sua funcionalidade com desenvoltura podendo chegar até a fazer um uso criativo do mesmo, não antevisto pelo desenvolvedor.
- manutenção do sistema. Quanto mais o usuário entender o sistema maiores são as chances de que ele consiga ser objetivo nos pedidos de alteração ou na definição de novos requisitos, incluindo aqui também o caso de aquisição automática de conhecimento.

Em [Norman&Draper'86] são apresentados estudos que visam o maior entendimento e melhoria no processo de interação homem-computador.

A Engenharia Cognitiva [Norman'86] abordou o processo de interação homem-computador dividindo-o nos golfos de avaliação e execução. No golfo de execução um usuário transforma seus objetivos e intenções em ações ou passos de interação. No golfo de avaliação ele constata se as ações tomadas atenderam suas expectativas e observa que outros recursos de interação estão disponíveis, inferindo para que servem. O entendimento destes golfos e das atividades do usuário relacionadas, como mostrado na figura 2, geraram uma série de diretrizes para o desenvolvimento de interfaces.

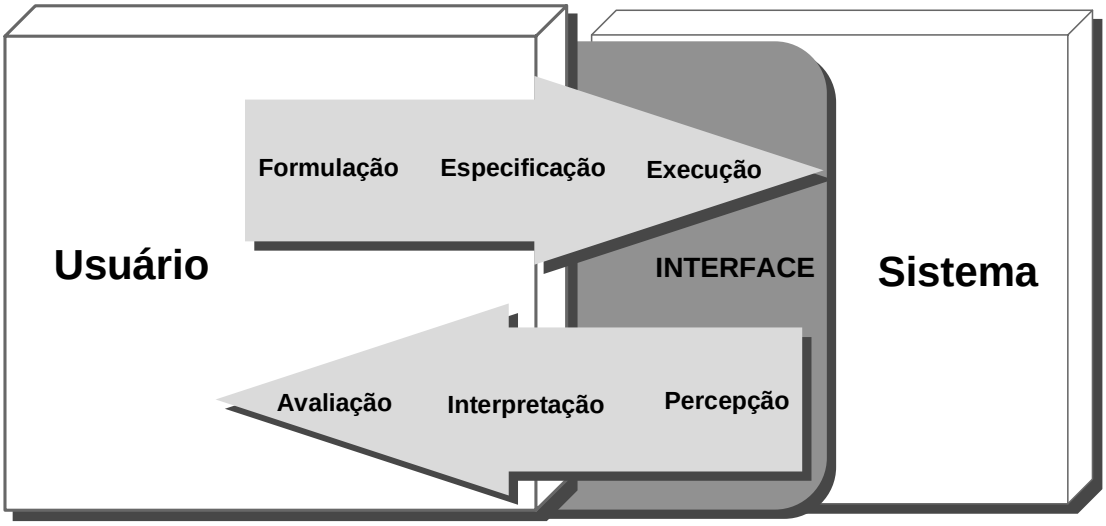


Figura 2: Golfos de Avaliação e Execução [de Souza&Leite'97]

Em uma primeira instância percebeu-se que interfaces devem ser de fácil aprendizado e utilização. Elas devem transmitir a sensação de que é o usuário quem está no controle, como se elas fossem ferramentas fáceis de usar, confortáveis e agradáveis. Tal como no uso de boas ferramentas, os usuários devem se engajar na interação de tal forma que se concentrem unicamente em seus objetivos finais, com a impressão de que a interface é "invisível" [Laurel'86]. Na busca pela definição mais precisa destes objetivos e pelo atingimento dos mesmos compreendeu-se que o design de interfaces deve ser tratado como um problema por si só, além do design da aplicação [Norman'86].

Um resultado deste trabalho é a percepção de que interfaces são linguagens através das quais usuários expressam suas intenções e sistemas mostram o que estão ou não habilitados a processar [Hutchins&Hollan&Norman'86]. Duas propriedades de linguagens de interfaces são a distância semântica e a distância articulatória. A distância semântica é a distância entre a intenção de um usuário e os significados disponíveis no sistema. Ou seja, mede se o modelo conceitual do sistema possui o significado (por exemplo objeto ou função) que o usuário procura e se a intenção dele é mapeada para apenas um significado ou para uma série deles. A distância articulatória compreende a distância entre o significado que se quer apresentar em um sistema e a forma como ele é apresentado.

[Hutchins&Hollan&Norman'86] mostram também que existem duas metáforas principais para interfaces: a conversacional e a de maquete. Na metáfora conversacional a interface é um intermediário entre o usuário e um mundo imaginário dentro do computador sobre o qual eles "falam". Ou seja, o usuário pede à interface que ações sejam executadas, esta pede ao computador que as execute e transmite a resposta do computador de volta para o usuário. Já na metáfora de maquete o usuário tem a impressão de que ele próprio atua sobre o mundo no meio computacional sem intermediários (tipicamente interfaces de manipulação direta).

O Planejador faz uso da metáfora conversacional quando utiliza a linguagem natural nas respostas explicativas. A metáfora de maquete também pode ser aplicada de forma a diminuir distâncias articulatórias. Em pequena escala, esta segunda metáfora foi utilizada no sistema MultiADD, em que se insere o Planejador aqui proposto. As duas metáforas juntas compõem uma única resposta explicativa.

Os recursos gráficos que concorrem para o surgimento de uma metáfora de maquete completam o conteúdo do texto em LN (metáfora conversacional). Eles servem como evidência do conteúdo do texto e sintetizam de forma gráfica um conteúdo que em LN seria cognitivamente mais caro. Alguns exemplos são as dependências conceituais e temporais entre alguns componentes da base de conhecimento do MultiADD. O módulo Explicador da interface permite também que algumas perguntas sejam expressas diretamente através de manipulação dos gráficos apresentados permitindo a sensação de maior engajamento e controle no usuário do que se fosse usada a metáfora conversacional. Veja figura 18 no capítulo 4.

Já a metáfora conversacional transmite informações abstratas de difícil mapeamento e expressão no modelo de maquete. Por exemplo, ela veicula a indicação de relevância dos gráficos apresentados, informações tutoriais sobre os mesmos, ou simplesmente textos explicativos que podem mesmo conter sinais de erro no sistema. Veja figura 18 no capítulo 4.

Ao se definirem as linguagens de entrada e saída providas pela interface, além de distâncias semânticas e articulatórias é importante observar a existência de co-referencialidade entre elas

[Draper'86]. Isto significa que elementos que são apresentados ao usuário na saída devem poder ser referenciados na entrada e vice-versa. Quando isto não for possível, esta restrição deve estar claramente sinalizada para que não se criem expectativas inadequadas.

Estas sinalizações são o que [Laurel'86, O'Malley'86 e Owen'86] chamaram de função de revelação, que é responsável por deixar claro para os usuários quais são as ações válidas em um sistema. Isto pode evitar tentativas de interação que levariam a erros e frustrações.

É papel do Planejador identificar as futuras interações possíveis sobre a resposta planejada. Na implementação do MultiADD, ele é responsável por mostrar que o texto em LN não é manipulável mas que os recursos gráficos o são.

Se apesar das sinalizações ainda houver erros de interação, [Clayton&Norman'86] mostraram que eles devem ser tratados adequadamente, não transmitindo a sensação de punição ou culpa aos usuários e mostrando para eles: que houve o erro, porque este aconteceu, e como ele pode ser contornado. Inclui-se aqui também erro na passagem pelo golfo de execução. Ou seja, um usuário pensa que determinada ação efetua suas intenções, ela é executada sem nenhum problema mas seu resultado não é o esperado. Neste caso, mesmo sendo uma ação válida, o usuário deve ter a chance de voltar ao estágio anterior à ação (recurso muitas vezes chamado de "*undo*") e recomeçar um novo golfo de execução, mapeando sua intenção para um outro conjunto de ações.

Para o Planejador, verificamos duas classes de erro a serem previstas. A primeira classe seria a de erros de interação com o Explicador no passo anterior à chamada do Planejador. Por exemplo, se o Explicador permitir que o usuário faça uma pergunta para a qual o Planejador não consegue obter resposta, uma mensagem de erro deve ser planejada de acordo com os conceitos já apresentados.

A segunda classe de erros é um efeito colateral ao requisito de fidelidade atendido. Com este requisito o Planejador muitas vezes verifica erros na base de conhecimentos. Quando o usuário efetua perguntas sobre algo que ficou inconsistente na base, o Planejador deve poder

dar uma resposta adequada e deixar claro que não houve erro de interação, mas sim de desenvolvimento da aplicação ou aquisição de conhecimento.

Finalmente [Adler&Winograd'92] propõem o desafio da usabilidade, no qual sistemas devem sustentar o potencial de entendimento, aprendizado e criatividade das pessoas que o utilizam. O design de sistemas deve poder tratar novidades, improvisações e adaptações. [Adler&Winograd'92] mostraram também que existe um diálogo implícito entre designers e usuários através dos artefatos produzidos pelos primeiros. Isto nos aponta para próxima área de estudos: a Semiótica Computacional.

2.2 Semiótica Computacional

A Semiótica é uma base teórica importante para se tratar problemas de Interação Homem-Computador e de Ciência da Computação como um todo. Todo processo de comunicação se dá através de um código [Eco'76]. Por exemplo, seres humanos se comunicam através do código da linguagem verbal ou podem se comunicar também através de um código gestual. O uso de um código por um grupo depende da criação de signos. Um signo representa alguma coisa; por exemplo, nossa cultura reconhece que 👍 é um signo cujo significado é positividade. Um signo está relacionado tanto a um objeto (a coisa que ele está substituindo) quanto a um interpretante. Um interpretante pode ser um sentimento, uma interpretação ou mesmo um outro signo, como mostrado na ilustração 3 fornecida por [de Souza&Barbosa'96]. O conceito de interpretante é uma das maiores contribuições da Semiótica e foi proposto por Peirce [Peirce'31], seguindo a tríade apresentada também na figura 3. O exemplo ilustra que o signo "casa" - uma palavra da língua portuguesa, está no lugar de um objeto concreto que ocupa um lugar e tempo relativo à vida de um indivíduo. No entanto, o signo pode ser interpretado indefinidamente, trazendo uma variedade de outros signos e significados à mente. Este processo de geração de uma corrente de significados é chamado de semiose [Eco'76].

O ganho de se usar conhecimento de Semiótica é o entendimento de que quando duas pessoas se comunicam elas negociam e regulam significados, de forma que a semiose potencialmente

ilimitada seja pragmaticamente circunscrita a um território de mútuo entendimento [Nadin'88, de Souza&Barbosa'96]. No entanto, não há garantias (e de fato muito pouca chance) de que, quando uma pessoa pergunta "Como vai você?" e a outra responde "Tudo bem.", o significado de <tudo bem> seja o mesmo para as duas. No entanto, o processo de comunicação é aparentemente satisfatório neste nível, porque de alguma forma os interpretantes das pessoas convergiram a uma configuração estável de entendimento em ambas as mentes. Quando se percebe que este grau de convergência não foi atingido, as pessoas prosseguem negociando significados, através de um diálogo, até que ele seja alcançado.

Adaptando o exemplo de [de Souza&Barbosa'96], suponha o seguinte diálogo:

- Quando é que você vai para casa?
- Ainda tenho que trabalhar e juntar mais dinheiro para conseguir comprar uma passagem...
- Não, não estava perguntando se você ia para sua terra natal. Só queria saber se você está indo para seu apartamento, pois ia oferecer uma carona....

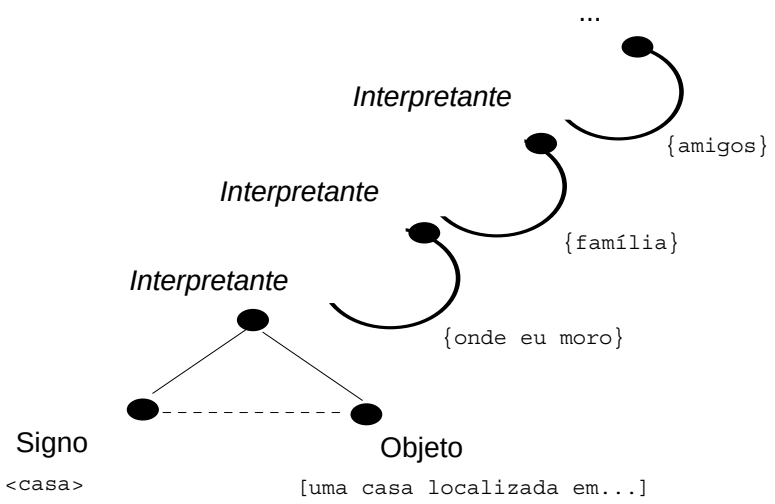


Figura 3: Interpretantes e Semiose Ilimitada

Podemos transportar o problema de comunicação mostrado no diálogo anterior para a área de Interfaces com o Usuário. Quantas vezes nos deparamos tentando interagir com um sistema e percebemos que o significado implementado não é aquele que estávamos esperando? Veja um relato ilustrativo (e verídico) do que se passa:

> Em uma antiga versão de um editor de textos para DOS havia a opção “Salvar” para gravar um arquivo em edição. Certa vez uma usuária escreveu algo que não queria em seu texto e ao se dar conta quis logo sair da aplicação e “salvar” seu trabalho anterior que estava correto. Fatalmente, ao sair, vendo a pergunta: “Deseja salvar o texto?”, respondeu SIM.

Para entender o que se passa neste caso, podemos nos apoiar na proposta de Engenharia Semiótica [de Souza’93]. Segundo esta proposta, um sistema é uma mensagem sendo transmitida de seu desenvolvedor para um usuário. O código usado para comunicar esta mensagem é a própria interface e o computador apenas o canal de transmissão da mensagem. A figura 4 mostra o Framework de Engenharia Semiótica proposto.

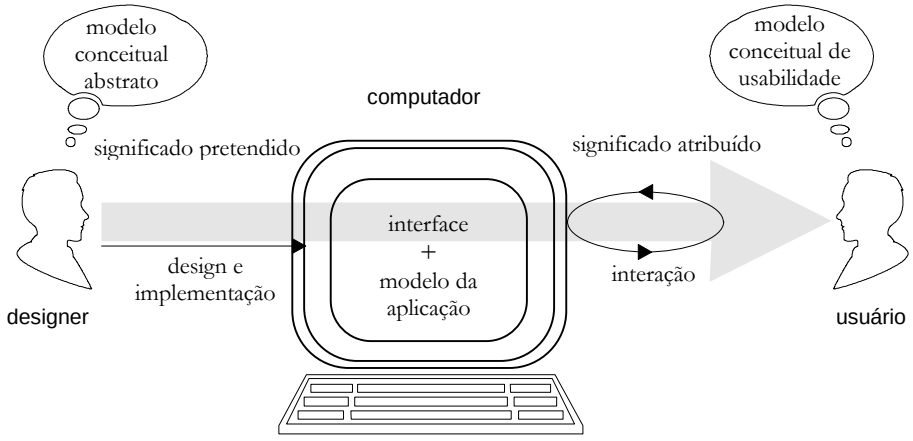


Figura 4: Framework de Engenharia Semiótica

Neste caso interfaces são mensagens em apenas uma direção, do designer para o usuário. Desta forma, não existe negociação direta para estabilizar significados e consequentemente maior dificuldade de encontrar equilíbrio entre o interpretante do designer e do usuário sobre a aplicação. Algumas regras podem colaborar para que se escolham signos de interface que encaminhem o processo de interpretação do usuário e reduzam as chances de não entendimento [Nadin’88, de Souza’93, de Souza’94]. Por exemplo devemos utilizar sempre que possível signos já estabelecidos na cultura do usuário, tais como ícones conhecidos (por

exemplo, a lixeira do *desktop*) ou as próprias estruturas de discurso propostas pelas estruturas retóricas. Ainda assim, não há garantias de que o objetivo de comunicação será atingido.

O estudo na área de Semiótica Computacional contribuiu em diferentes aspectos. Uma primeira contribuição que podemos citar é a decisão de não construir um modelo de usuário adaptativo a partir de interação com o sistema. Através de estudo dos vários artigos em [Kobsa&Wahlster'89], entendemos por modelo de usuário o componente de um sistema que é responsável por detectar características do usuário (tais como nível de conhecimento, objetivos e planos) através da interação deste com a interface. A medida que propriedades do perfil do usuário vão sendo identificadas pelo sistema, este se adapta automaticamente em busca de um diálogo mais cooperativo.

Sistemas que possuem modelo de usuário são desenhados como parceiros de diálogo onde idealmente as tarefas de construção de modelo mental do usuário e do sistema são as mesmas daquelas envolvidas na comunicação pessoa-a-pessoa [Wahlster&Kobsa'89]. Isto diverge da perspectiva da Engenharia Semiótica, onde um ou mais interpretantes do desenvolvedor são transmitidos para o usuário através de uma mensagem passada pela interface. Não há portanto simulação de comportamento humano, ou seja semiose, como é idealizado pela modelagem de usuários.

Em consequência desta divergência de abordagens, a Semiótica Computacional questiona sob vários aspectos a possibilidade de melhoria de usabilidade a partir de modelo de usuário. Primeiramente podemos dizer que a Semiótica defende a estabilidade dos códigos de interação [Nadin'88, de Souza'94], que permite ao usuário domínio total da linguagem de interface. Já modelos de usuário propõem interfaces adaptativas que quebram a estabilidade e apresentam necessidade de re-aprendizado ao usuário. Poder-se-ia argumentar que o custo de re-aprendizado é compensado por uma maior usabilidade futura. Cabe lembrar porém que o grau e a frequência de mudanças na interface a partir do modelo de usuário são decisões arbitrárias do desenvolvedor do sistema e dependentes do comportamento do usuário, dificultando assim a previsão desta compensação. Já linguagens estáveis não implicam em re-aprendizado e uma vez dominadas podem garantir usabilidade.

Outro ponto de divergência entre a Engenharia Semiótica e sistemas com modelagem de usuário corresponde ao fato de que muitos destes sistemas não esclarecem para o usuário a existência do modelo de usuário atuante, obscurecendo assim a mensagem do desenvolvedor. Caso se quisesse transparência, esta mensagem seria então algo como: *"Os possíveis perfis de usuário são: X, Y, Z. Suas características o encaixam no perfil Y."* Sendo o modelo de usuário um componente inteligente baseado em conhecimento, o usuário deveria poder perguntar ao sistema o que o levou a tomar esta decisão e quais os impactos da mesma sobre o funcionamento do sistema. Para ter controle sobre a aplicação que utiliza, ele deveria poder também discordar da decisão tomada pelo sistema. Sob a ótica do usuário, o sistema possui um sofisticado recurso de configuração de interface. Note que os processos de percepção (já que a interface se adapta autonomamente), entendimento e ajuste deste recurso de configuração competem com o objetivo final da aplicação (a resolução de um problema do usuário) pela atenção do usuário (tornando-se mais um problema a resolver), o que vai de encontro ao Design de Sistemas Centrado no Usuário.

Outra contribuição da Semiótica para o desenvolvimento do Planejador é a compreensão de que cada resposta explicativa gerada é uma interpretação nossa sobre como deve ser explicado o conhecimento e funcionamento do sistema baseado em conhecimento que serve de base e consequentemente do domínio de aplicação ali representado. As contribuições desta área influenciaram no desenvolvimento do Planejador no sentido de que tivemos que definir as mensagens que queremos transmitir através de nossas respostas. Estas mensagens estão descritas no capítulo 3 (Solução).

Durante o desenvolvimento da aplicação, os conhecimentos nesta área também nos prepararam melhor para entender requisitos de alterações nas soluções que tínhamos inicialmente proposto, seja por entender que nossa interpretação do sistema ou do domínio haviam sido errôneas, seja por compreender que a resposta gerada não ocasionava o interpretante adequado na mente do usuário.

Outra contribuição importante é o reconhecimento da maior efetividade de códigos conhecidos pelo receptor da mensagem em relação a códigos não familiares. Com isto podemos argumentar a necessidade de um Planejador de Respostas Explicativas centrado em Linguagem Natural, que é o código mais expressivo que conhecemos [Nadin'88, Brennan'90], e estruturas retóricas. Ainda assim a utilização de outros códigos é prevista pelo Planejador e quando isto acontece o Planejador procura usar a LN para ensinar ao usuário a interpretar um outro código que pode não ser convencional. O Planejador também mostra o modelo de relevância entre os diferentes códigos apresentados. É comum por exemplo que Sistemas de Explicação mostrem de alguma forma a representação da base de conhecimento (modelo gráfico da base, *trace* de regras, etc.) e a maioria das vezes este é um código não conhecido pelo usuário. Assim, é atribuição da LN (código conhecido) explicar o que se passa, tornando-se um tutorial para uma futura aquisição automática de conhecimento.

2.3 Sistemas de Explicação

Em geral uma explicação é algo que esclarece uma porção de conhecimento a uma pessoa [Cawsey'92]. O processo de explicação é intrinsecamente interativo, requerendo diálogo entre emissor e destinatário [Pollack&Hirschberg&Webber'82, Paris'91, Swartout&Moore'91, Cawsey'92, Moore'95 e Moore&Mittal'96]. Este processo se dá por completo quando o destinatário se mostra satisfeito e parece entender o que foi explicado. Sob uma perspectiva semiótica a explicação pode ser vista como uma negociação de interpretantes onde o especialista ou conselheiro tenta expressar e esclarecer seu interpretante para o iniciante ou consulente.

Com o advento de Sistemas Baseados em Conhecimento surgiu a necessidade de se desenvolverem Sistemas de Explicação que pudessem esclarecer para o usuário o conhecimento ali modelado e processado. [Moore'95] traça um histórico da evolução de Sistemas de Explicação. Os Sistemas de Explicação de primeira geração usavam principalmente duas técnicas para elaborar respostas explicativas. Uma delas é a técnica

chamada de texto “enlatado”, que foi utilizada por exemplo no sistema LISP-CRITIC [Fisher’87]. As perguntas possíveis eram antecipadas pelos desenvolvedores e explicações eram produzidas através da amostragem de textos pré-preparados que eram associados a estruturas ou processos do programa. Ou seja, havia *templates* de *strings* de texto que eram preenchidos com valores específicos de uma determinada execução do programa. A outra técnica é chamada de tradução de código e construía explicações parafraseando códigos de programas ou *traces* de execução. Esta técnica foi utilizada por exemplo nos sistemas *Digitalis Therapy Advisor* [Gorry&Silverman&Pauker’78 e Swartout’77] e MYCIN [Scott&Clancey&Davis&Shortliffe’84].

Estas técnicas se mostraram deficientes, mas proporcionaram um maior esclarecimento sobre como deveriam ser construídos Sistemas de Explicação. Os textos enlatados não garantiam fidelidade entre a explicação e a base de conhecimento visto que estes dois módulos podiam ser modificados independentemente. Para a produção desse tipo de explicação o desenvolvedor previa todas as questões e situações possíveis e então construía os respectivos *templates*. Caso houvesse poucas perguntas e um pequeno número de situações que fossem caracterizadas por poucos aspectos, textos fluentes podiam ser gerados com pouco esforço. No entanto, a medida que a complexidade destes componentes aumentava, crescia também o número de *templates*, tornando-os de difícil gerência. Esta complexidade era rapidamente alcançada quando se tentava garantir sensibilidade e/ou reatividade.

Os sistemas de tradução de código trataram o problema de fidelidade e com isso levantaram outras deficiências, tal como a representação inadequada de conhecimento. Por exemplo, no sistema MYCIN [Buchanan&Shortliffe’84] percebeu-se que regras de produção individuais misturavam em suas premissas conhecimento de mundo, de estratégia de resolução de problema e conhecimento causal respectivo ao domínio. A tradução destas regras gerava textos de difícil compreensão para o usuário. Outro problema identificado era a representação insuficiente de conhecimento visto que os sistemas só representavam a informação necessária para solucionar o problema em questão. Com isso, outros tipos de conhecimento do domínio, como por exemplo terminologia não estavam disponíveis. Em relação a técnica de geração de

explicação também foram percebidas limitações, como por exemplo a insensibilidade ao contexto de discurso.

Estas experiências incentivaram a especificação de novos requisitos para Sistemas de Explicação, como os apresentados na seção 1.1 (Requisitos do Planejador). Para atender a estas novas especificações percebeu-se que a tarefa de construção de Sistemas de Explicação precisava ser tratada como a solução de um problema por si só, independente do problema tratado pelo sistema especialista. Passamos então para o estudo de alguns projetos desenvolvidos segundo esta abordagem.

A arquitetura EES (*Explainable Expert System*) [Neches&Swartout&Moore'85, Swartout&Paris&Moore'91, Paris'91, Moore&Swartout'91, Moore'95] foi utilizada para desenvolver o sistema PEA (*Program Enhancement Advisor*) que auxilia usuários a melhorar programas desenvolvidos em linguagem LISP. A arquitetura EDGE (*Explanatory Discourse Generator*) [Cawsey'92] foi implementada e usada para gerar explicações interativas sobre funcionamento de circuitos. Finalmente o ambiente LINX foi estudado em sua versão mais atualizada onde são representados domínios classificatórios [Nunes'91, Dias'94, Quental'95, García'95, Oliveira'95], devido a ter um gerador destinado para o português do Brasil.

Uma das contribuições deste estudo foi a própria definição dos requisitos do Planejador aqui proposto. Com ela pudemos formar uma idéia concisa do trabalho que se queria desenvolver e prever os caminhos que deveriam ser percorridos para torná-lo possível. Além disso o estudo forneceu indicadores de que o Planejador trabalharia sempre com linguagens computáveis, visto que os requisitos linguísticos que ele atende são um subconjunto dos requisitos e soluções já implementados por estas experiências.

Juntamente com a definição dos requisitos, tomou-se a decisão de não se construir modelo de usuário no Planejador. Além dos argumentos em favor de uma perspectiva centrada em uma fixação de mensagem do desenvolvedor do sistema, apresentados na seção 2.2 (Semiótica Computacional), apresentamos agora como os estudos na área de Sistemas de Explicação nos levaram a manter nossa decisão.

[Moore&Swartout'91] mostraram que a grande ênfase dada a modelos de usuário em trabalhos anteriores se devia ao fato de que para cada pergunta possível havia apenas uma única resposta explicativa. Assim o desenvolvedor do sistema precisava de informações sobre o usuário para tentar garantir que este compreendesse a única resposta possível de ser gerada. A experiência na construção de modelos de usuário mostrou que as abordagens que requerem um modelo detalhado do usuário são suspeitas, pois dificilmente elas serão capazes de construir modelos de usuário completos e corretos [Sparck Jones'84, Sparck Jones'89]. Além disso, Sparck Jones questiona também se a utilização de um modelo de usuário para influenciar o raciocínio e geração de respostas é um problema tratável.

[Moore&Swartout'91] mostram que se pode melhorar a interação entre sistemas de explicação e usuários através de recursos de diálogo, fornecendo ao usuário mais do que uma chance para entender uma resposta explicativa. Esta informação se baseia no fato já mencionado de que o processo de explicação entre seres humanos é intrinsecamente interativo requerendo diálogo entre emissor e destinatário. A capacidade de diálogo em um sistema aumenta o poder de expressão do usuário possibilitando que ele:

- indique os efeitos que as respostas explicativas lhe causaram (se foram entendidas, se ele deseja uma outra resposta completa ou se deseja que alguma parte específica seja mais detalhada, etc.); e
- indique se deseja fazer perguntas subsequentes (relacionadas ao discurso anterior), entrando em um processo de diálogo contextualizado.

Os autores escrevem que, apesar de não mais requerido, o modelo de usuário ainda pode continuar sendo considerado, mas [Paris'91] mostra que o teor da informação expressa pelo usuário através de interação é sempre maior do que aquele contido no modelo de usuário. Ou seja, para o EES sempre que houver contradição entre modelo de usuário e informação fornecida pelo usuário esta última passa a ser considerada em detrimento da primeira.

Podemos dizer então que devido à dificuldade de se construírem modelos de usuários e à complexidade de sistemas capazes de participar de diálogos (o que levou à diminuição da

importância desses modelos) foi reforçada nossa decisão de não contemplar modelo de usuário. Apesar de ainda não ter capacidade de diálogo o Planejador foi construído de forma a permitir uma futura extensão neste sentido, como será visto no capítulo 5 (Discussão). Vale observar que dentre as arquiteturas estudadas, a EES e a EDGE possuem modelo de usuário.

Além da definição de requisitos e da discussão sobre modelo de usuário, os seguintes aspectos foram observados em relação às aplicações estudadas: tratamento dado às informações da base de conhecimento, escolha de perguntas, definição de contexto e informações que influenciam na determinação do conteúdo e forma das respostas explicativas e geração de linguagem natural, onde este último será descrito na seção 2.5 (Geração de Linguagem Natural). Apresentamos então um resumo dos sistemas mostrando como atacam estes pontos.

O sistema desenvolvido segundo a arquitetura EDGE é um sistema de explicação de cunho tutorial. De agora em diante, assim como em [Cawsey'92], o nome EDGE será utilizado para denominar tanto a arquitetura quanto o sistema desenvolvido a partir dela. O sistema EDGE não pretende solucionar problemas de um domínio, como é o caso do sistema PEA. A função única do EDGE é a própria explicação, ou seja, seu objetivo final é o próprio esclarecimento de uma porção de conhecimento para o usuário. Este aspecto direcionou a construção da base de conhecimento que portanto não enfrentou as dificuldades já mencionadas dos sistemas baseados em conhecimento: insuficiência ou inadequação de representação de conhecimento para o fornecimento de explicações. Além da base de conhecimento do domínio todos os sistemas estudados possuem alguma forma de representação de conhecimento linguístico, que serão vistas com mais detalhes na seção 2.5 (Geração de Linguagem Natural).

Estudando o sistema EDGE vimos que ele é caracterizado por explicações longas, tais como os monólogos que são as estruturas de discursos tutoriais. O primeiro passo no uso do EDGE é a determinação do tópico que será explicado. Durante as explicações os usuários têm a oportunidade de fazer interrupções, entrando em um processo de diálogo. As possíveis interrupções são planejadas pelo sistema que monta um menu onde os usuários podem expressar: se estão ou não entendendo a explicação em andamento; o que não está sendo compreendido; ou se eles querem que uma explicação seja repetida ou abandonada. Ou seja,

as opções de menu podem fazer referência ao conhecimento explicado ou a discurso anterior. Estas opções não foram especificadas explicitamente, no entanto pelo domínio escolhido e pelos exemplos citados ao longo do livro percebemos que o EDGE é capaz de responder a perguntas sobre o que são os objetos representados na base de conhecimento (terminologia e analogia) e como eles funcionam. Perguntas que permitem *feedback* do usuário sobre se a explicação foi ou não compreendida também estão disponíveis. Dentre estas perguntas inclui-se a pergunta "*what?*", para o caso onde o usuário não consiga identificar ou expressar o que não foi compreendido.

No sistema EDGE o contexto de discurso compreende: suposições sobre conhecimentos do usuário, objetos que estão em foco e papéis dos participantes da explicação (mestre-aluno, conselheiro-aconselhado, etc.). Os objetos em foco determinam a ordem da explicação e os papéis dos participantes influenciam aspectos da organização do diálogo (mais ou menos oportunidades de interrupções). O conhecimento do usuário determina a estratégia de explicação, se informações de pré-requisito para entendimento devem ou não ser incluídas e que tipo de diálogo posterior será possível. Apesar de não incluídos na definição de contexto, os planos de discurso anteriores e os planos parciais de discurso futuros são utilizados:

- no tratamento de interrupções colaborando na montagem dos menus que são apresentados aos usuários (permitindo referências a discursos anteriores); e
- no tratamento de incompreensões do usuário. Caso as suposições que foram feitas sobre o conhecimento de usuário não sejam a fonte do problema, os planos de discurso mostram qual foi a última explicação apresentada e com ela quais as possíveis explicações alternativas.

Em geral podemos destacar alguns aspectos que diferenciam o sistema EDGE de um ou mais dos outros projetos estudados:

- ele não está preparado para geração de explicações dentro do contexto de resolução de problemas. Para isso seria necessária uma adaptação no sentido de aumentar a definição de contexto e gerar explicações mais resumidas, visto que o objetivo principal do usuário será achar uma solução para um problema.

- a impossibilidade de mudança de tópico sem o abandono da explicação precisaria ser dirimida.
- a possibilidade de interrupção pelo usuário no meio da explicação permite que o usuário faça sua pergunta mantendo o foco e contexto de discurso próximo ao momento em que uma dúvida apareceu. Quando só é possível fazer a pergunta depois de finalizada a explicação o foco já foi perdido e toda a explicação gerada após a dúvida pode ter sido em vão.

Com isso concluímos nosso estudo sobre o sistema EDGE e passamos agora para o estudo da arquitetura EES e do sistema desenvolvido segundo ela: o PEA. A arquitetura EES permite que se desenvolva um sistema baseado em conhecimento que tem como objetivo auxiliar um usuário a realizar uma tarefa. No caso do PEA o objetivo é ajudar na melhoria de um programa LISP segundo os aspectos de legibilidade e manutenção. A arquitetura de desenvolvimento do EES facilita a especificação de informações necessárias à explicação, solucionando assim o problema de representação de conhecimento.

[Moore'95] descreve as perguntas disponíveis para os usuários do PEA (itens em itálico correspondem a variáveis):

- Por quê?
- Por que você está tentando atingir *o objetivo X*?
- Por que você está usando *o método M* para atingir *o objetivo X*?
- Por que você está executando *a ação A*?
- Como você atinge *o objetivo X*? (em geral)
- Como você atingiu *o objetivo X*? (específico)
- O que é *o conceito C*?
- Qual é a diferença entre *o conceito C1* e *o conceito C2*?
- Hã? (se o usuário quiser indicar que a resposta não foi entendida)

Uma distinção percebida em relação ao sistema EDGE é que neste as perguntas só podem ser feitas através de menus, onde cada opção de menu é uma pergunta. A arquitetura EES dá maior expressividade ao usuário permitindo que ele escreva suas perguntas em uma sub-

linguagem da língua inglesa, caso não queira utilizar os menus disponíveis. Já a arquitetura LINX permite a entrada de perguntas a partir de um subconjunto da língua portuguesa. Esta entrada pode ser feita tanto a partir de digitação livre como de seleção palavras apresentadas em uma sequência de menus que indicam as composições válidas para a gramática prevista. Concluímos então que a relação de co-expressividade entre as linguagens de entrada e saída dos sistemas LINX e EES é maior do que a do sistema EDGE.

Estudadas as perguntas possíveis no EES passamos para a verificação da definição de contexto. O contexto é definido como o histórico de diálogo, o modelo de usuário e a base de conhecimento. O histórico de diálogo contém as perguntas dos usuários com os planos de texto completos das respostas já geradas pelo sistema. Estes planos de texto representam explicitamente como as partes do texto se relacionam e para que servem. Eles mostram também, caso haja, quais suposições foram feitas em relação ao conhecimento do usuário e que estratégias alternativas poderiam ter sido selecionadas em diferentes pontos do planejamento.

O histórico de diálogo colabora para a interpretação das perguntas de várias formas:

- determinando o foco de discurso;
- evitando escolha de uma interpretação de pergunta cuja resposta já tenha sido dada pelo sistema;
- identificando o objetivo comunicativo que não foi atingido no caso de pergunta mal formulada (Hã?), que pode ser:
    - um objetivo comunicativo que introduziu novos conceitos;
    - um objetivo comunicativo que fez suposições sobre o usuário; e
    - um objetivo comunicativo de nível mais alto caso as heurísticas anteriores não gerem nenhum resultado.

  As informações do objetivo comunicativo resultante são utilizadas depois para elaborar uma re-explicação.

Para a geração de respostas o histórico de diálogo é consultado pelas chamadas heurísticas de seleção que atuam na escolha de um operador de plano dentre vários para atingir um objetivo

comunicativo. Elas checam as informações mencionadas em explicações anteriores de forma a evitar repetições ou verificar se algo foi citado, permitindo assim a construção de explicações baseadas em informações prévias do discurso. Na fase de geração de perguntas o histórico do diálogo é usado para que não se repitam perguntas que já foram feitas.

Quanto ao modelo de usuário, constatamos que ele atua:

- na interpretação de perguntas, descartando interpretações que levem a uma pergunta cujo conhecimento é dominado pelo usuário;
- na geração de respostas, permitindo que as mesmas se baseiem no conhecimento do usuário, por exemplo, no caso de analogia entre um objeto desconhecido e um conhecido; e
- na geração de perguntas, para que não se faça perguntas cujas respostas já são conhecidas pelo usuário e para impedir perguntas que peçam justificativas sobre os objetivos do usuário.

A base de conhecimento do EES mostra o conhecimento disponível e seu estado atual, inclusive um *trace* de execução. Estas informações influenciam por exemplo a escolha de atributos que serão utilizados quando objetos são comparados (perspectiva) ou, quando a comparação não é entendida pelo usuário, se existem exemplos que possam ser utilizados para facilitar a compreensão.

Em uma análise geral do EES percebemos que ele tem como pontos fortes uma arquitetura de desenvolvimento de SBCs que facilita a carga de informações de explicações e uma definição de contexto abrangente. Um aspecto que poderia ser melhorado quanto à geração de explicações seria intercalar o processo de planejamento e linearização das respostas explicativas possibilitando assim interrupções no meio de explicação.

Focalizando agora no estudo do ambiente LINX, observamos que a base de conhecimento da versão atual, assim como a do sistema EDGE, foi desenvolvida com vistas à sustentação de um diálogo com o usuário. Por isso não existiram problemas de representação de conhecimento do domínio, no que tange ao processamento do discurso. O tipo de

representação de conhecimento adotado pelo LINX difere dos outros sistemas pois ele adota, além de redes semânticas, a lógica de primeira ordem, o que causa também uma diferenciação quanto às perguntas e respostas possíveis.

Verificamos então as perguntas respondidas pelo LINX, onde os exemplos citados foram retirados de [Quental'95]. São elas:

- perguntas sobre inclusão de classes em superclasses, inclusão de indivíduos em classes ou perguntas sobre propriedades de classes ou de indivíduos. Elas podem estar nas seguintes formas:
  - perguntas SIM/NÃO, como por exemplo: Os cães são mamíferos?
  - perguntas alternativas, como por exemplo: Os cães latem ou miam?
  - perguntas com quantificador existencial, como por exemplo: Alguma galinha voa?
  - perguntas Qu_, como por exemplo: Que cão morde?
- perguntas sobre o raciocínio desenvolvido pelo sistema para chegar a uma resposta: Por quê? e Por que não...?. Esta última na verdade é articulada como a pergunta Por quê?, mas como ela pode referenciar uma sentença anterior negativa, ela traz como resposta o motivo que levou ao não atendimento da expectativa do usuário.

As perguntas podem referenciar tanto o conhecimento do domínio quanto discursos anteriores. No entanto não é possível expressar incompreensão em relação a uma resposta dada, como é o caso do PEA (EES) e do EDGE.

O LINX possui uma definição estrutural de contexto de discurso que contém as seguintes informações:

- diálogo anterior - conjunto de árvores retóricas que mostram a sequência de pares pergunta e resposta adjacentes sobre um mesmo tópico e eventualmente sobre um novo tópico;
- foco da pergunta - ponto central da dúvida do usuário. (O LINX possui um conjunto de heurísticas para determinação de foco.); e
- pressuposição e tópico da pergunta - noções que se opõem à de foco. Segundo [Nunes'91], pressuposição é:

> "...tudo aquilo que o questionador assume como verdade (ou não acredita ser falso), desde que, caso contrário, estaria violando uma convenção conversacional, a de que nenhuma resposta direta à sua pergunta seria correta - agora entendendo-se por resposta direta uma afirmação, negação ou um valor requisitado.".
>
> Esta informação é derivável a partir do foco e da forma da pergunta.

A definição de contexto influencia o planejamento da forma do discurso, como será visto a seguir. Alguns outros componentes do LINX que também exercem influência são: o léxico [Dias'94] - dicionário que apresenta também a taxonomia do domínio, a gramática [Quental'95] que é ao mesmo tempo interpretativa e gerativa da língua portuguesa pseudo-natural e a árvore de prova em Lógica de Primeira Ordem [Oliveira, Souza&Haeusler'96] gerada pelo sistema que serve de base para os textos gerados.

Na interpretação de perguntas a gramática usa os conceitos de foco e tópico, podendo inclusive resolver termos anafóricos e compreender perguntas elípticas. A interpretação dada pelo sistema é apresentada imediatamente permitindo que o usuário faça correção, caso necessário, mesmo antes de completar a sentença. Para a correção o sistema apresenta ao usuário uma hierarquia de focos prováveis dentre os quais o usuário seleciona o desejado. A partir da pergunta é gerada uma proposição lógica a ser avaliada pelo Provador de Teoremas [Oliveira'92]. O Provador retorna uma árvore de prova que determina o conteúdo da resposta a ser apresentada.

Na elaboração de respostas também são usados os conceitos de foco e tópico. O gerenciador de interações usa todas as informações da estrutura de discurso para permitir que pontos do texto ou o texto como um todo venham a ser selecionados por manipulação direta em futuras perguntas. Com isso é possível por exemplo selecionar um discurso anterior e pedir que o mesmo seja repetido só que agora usando um novo foco especificado pelo usuário. As informações de foco e tópico possibilitam também a geração de termos anafóricos e a entrada de perguntas elípticas. O léxico evita perguntas sem sentido (tal como o exemplo de [Quental'95]: "O dono de Rex late?") ou ambíguas.

Avaliando o ambiente LINX percebemos que alguns aspectos poderiam ser melhorados. Ele não permite que o usuário expresse incompreensão e consequentemente não possui respostas alternativas para as perguntas, embora ele disponha de um nível de cooperatividade em casos de incompreensão. A geração de textos que explicam trechos de prova em maior profundidade representa, a rigor, uma alternativa de explicação através de maior elaboração. Assim como no sistema EDGE, outra melhoria seria prepará-lo para ser usado em resolução de problemas de natureza mais procedimental. Seu ponto forte e distintivo se encontra em prover uma linguagem natural para a realização de perguntas, garantindo co-expressividade entre a linguagem de entrada e de saída.

O estudo de todas estas experiências enriqueceu nosso conhecimento sobre Sistemas de Explicação esclarecendo as diversas formas de se atacar um mesmo problema e mostrando suas vantagens e desvantagens. Este estudo nos levou à solução que será proposta no capítulo 3 (Solução). Com esta solução pretendemos oferecer um Planejador que ajude a construção de Sistemas de Explicação satisfatórios, acarretando assim maior grau de usabilidade e confiança [Paris'91].

No entanto, antes de apresentar a solução ainda é preciso esclarecer como as áreas de *Rhetorical Structure Theory* e Geração de Linguagem Natural podem contribuir para o desenvolvimento de Sistemas de Explicação e do Planejador aqui proposto.

2.4 *Rhetorical Structure Theory*

A *Rhetorical Structure Theory* (RST) é um método linguístico útil para descrever textos em linguagem natural [Mann&Thompson'87]. Ela caracteriza a estrutura destes textos através das relações funcionais existentes entre as partes que os constituem. A estrutura identificada nos textos é hierárquica e seu papel comunicativo também é discutido pela RST.

Como mostram [Mann&Thompson’87] a RST possui quatro tipos de objetos definidos:

- Relações;
- Esquemas;
- Aplicações de Esquemas; e
- Estruturas.

Basicamente as relações acontecem entre duas partes de texto, chamadas núcleo e satélite, que podem se apresentar em qualquer ordem. O núcleo é a parte do texto que é essencial ao objetivo do escritor, enquanto que os satélites reforçam o efeito do núcleo. A definição de uma relação consiste em quatro campos, além do núcleo e do satélite:

- efeito → reação que o escritor pretende provocar no leitor através do uso da relação
- restrições no núcleo → condições quanto ao núcleo que devem ser atendidas para que a relação seja válida
- restrições no satélite → condições quanto ao satélite que devem ser atendidas para que a relação seja válida
- restrições na combinação do núcleo com o satélite → condições quanto à combinação do núcleo com o satélite que devem ser atendidas para que a relação seja válida

Os esquemas RST são formados de diferentes partes de texto e, ao serem aplicados a um texto, determinam as possíveis estruturas retóricas do mesmo. Os cinco tipos básicos de esquemas RST, exemplificados na figura 5, resultam da definição destes em termos de relações retóricas.

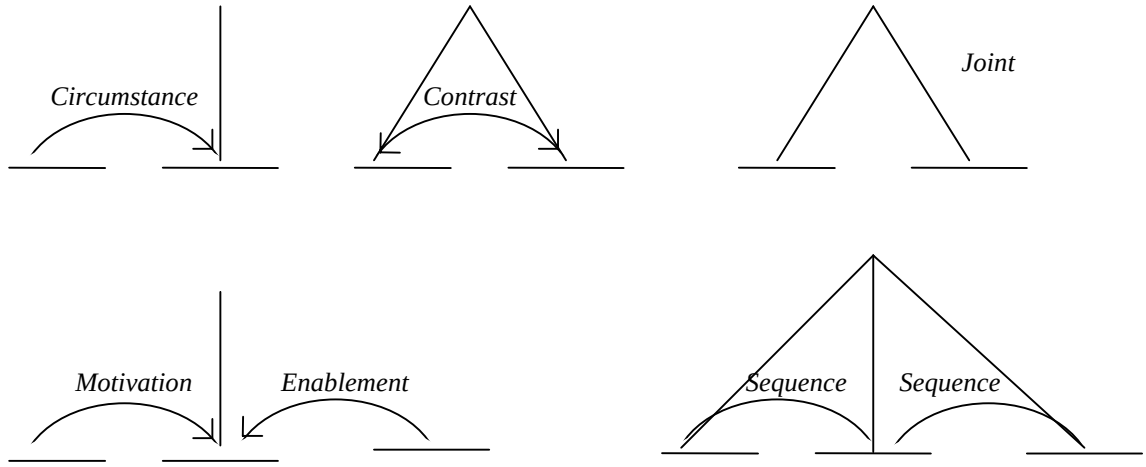


Figura 5: Exemplos dos Cinco Tipos de Esquemas RST [Mann&Thompson'87, pp. 7]

Os esquemas não mencionados na figura seguem o padrão representado pela relação *Circumstance*: uma única relação com núcleo e satélite [Mann&Thompson'87].

Os fatores diferenciais dos tipos de esquemas são:

- quantidade de núcleos. Os núcleos estão representados na figura pelas linhas verticais e diagonais. Como se vê, os esquemas (assim como as relações retóricas) podem ter um núcleo, dois núcleos ou vários núcleos.
- quantidade de relações. Um esquema pode conter uma ou mais relações retóricas, como é o caso do quarto esquema da figura 5 que apresenta duas relações (representadas pelos arcos) que compartilham um mesmo núcleo.

A análise estrutural de um texto mostra um conjunto de aplicações de esquemas. Veja um exemplo de uma análise RST realizada sobre um texto de resposta explicativa da aplicação que será descrita no capítulo 4 (Aplicação).

Texto:

“O valor do parâmetro *P* é *val(p)* pois assim foi determinado pelo projetista. Isto retifica o valor obtido pelo sistema - *val (s)* e causa uma inconsistência com as heurísticas que podem ser vistas na Tabela de Heurísticas.
Veja no Histórico o contexto onde esta decisão foi tomada. Para maiores detalhes clique sobre os passos do Histórico.”

Análise RST:

Partes de texto:

1= “O valor do parâmetro *P* é *val(p)*”

2= “pois assim foi determinado pelo projetista”

3= “Isto retifica o valor obtido pelo sistema - *val(s)*”

4= “e causa uma inconsistência com as heurísticas”

5= “que podem ser vistas na Tabela de Heurísticas.”

6= “Veja no Histórico o contexto onde esta decisão foi tomada”

7= “Para maiores detalhes”

8= “clique sobre os passos do Histórico.”

Diagrama RST, indicando as relações retóricas entre as partes de texto:

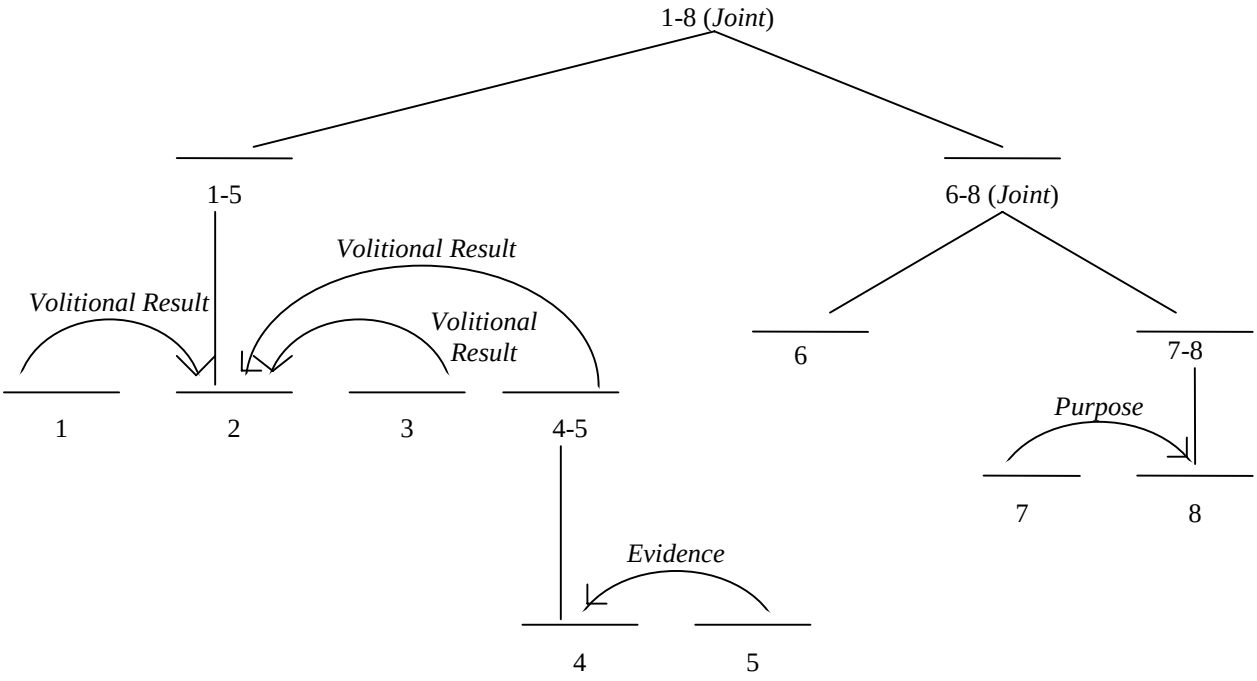


Figura 6: Diagrama RST

Ao se analisar um texto para se construir sua estrutura RST uma pessoa avalia se as partes do texto lido são plausíveis de se encaixarem em determinadas relações retóricas. Como é da natureza da análise de textos, não há certeza de que aquelas foram de fato as relações objetivadas pelo escritor e muitas vezes podem ocorrer ambiguidades na determinação das relações [Mann&Thompson’87].

Sob o ponto de vista semiótico isto se deve ao fato de que não é possível determinar os interpretantes que serão gerados nas mentes dos leitores. O processo de semiose (limitado pelo leitor) pode: tomar um rumo diferente daquele esperado pelo autor, onde o leitor dificilmente chegará ao interpretante desejado pelo autor; tomar um rumo onde é possível alcançar o interpretante do autor, mas o leitor pára o processo de semiose em um ponto onde o interpretante não é o mesmo; felizmente, cessar onde o interpretante do leitor é aquele desejado pelo autor.

Em uma das análises do texto anterior conseguimos chegar a uma estrutura retórica diferente daquela apresentada na figura 6. As partes 3, 4 e 5 poderiam ser entendidas através de uma relação de *Non-Volitional Result* com a parte 2. Quando construímos o texto supusemos (na posição de autor) que estava claro para o usuário da Explicação que a retificação de valores do sistema é uma atitude consciente do usuário de Design (visto que o sistema apresenta uma mensagem alertando o usuário de Design para isso). No entanto percebemos outro interpretante sendo gerado pelo usuário da Explicação, onde ele inferiu que esta consequência era desconhecida por aquele que tomou a decisão. Isto nos aponta, inclusive, para uma linearização de texto em que itens lexicalmente marcados desfaçam as ambiguidades. Veja-se, por exemplo, como a palavra "deliberadamente" muda o quadro interpretativo em 2: "pois assim foi deliberadamente resolvido pelo projetista". Outra alternativa na mesma linha seria:

> "O valor do parâmetro *P* é *val(p)* pois assim foi determinado pelo projetista. Ele retificou o valor obtido pelo sistema - *val (s)* e causou uma inconsistência com as heurísticas que podem ser vistas na Tabela de Heurísticas.
> Veja no Histórico o contexto onde esta decisão foi tomada. Para maiores detalhes clique sobre os passos do Histórico."

Nos dois casos, a ambiguidade seria menos provável. [Scott&de Souza'90a] apresentam heurísticas que podem ser aplicadas de forma a diminuir ambiguidades em esquemas retóricos.

Embora a RST tenha sido criada para análise e construção de textos, até o trabalho de Hovy em 1988 ela tinha sido usada principalmente para análise. O uso da RST para geração de textos possui duas vantagens principais. A primeira é que o esquema RST favorece a geração de textos multi-sentenciais estruturados e coerentes para os padrões humanos. A segunda virtude é o conhecimento do possível efeito que o uso de uma relação retórica provoca no modelo mental do leitor. Com isto pode-se associar esquemas a estratégias de discurso para atender um objetivo de discurso particular [Paris'91]. A estrutura hierárquica permite que qualquer esquema RST seja expandido em qualquer outro em qualquer ponto, não impondo restrições de ordem ou sequência entre as partes do texto.

Semioticamente esta segunda vantagem mostra que podemos usar as especificações das relações retóricas (campos efeito e restrições) e inferências sobre uma combinação delas como indicação ou guia para os interpretantes pretendidos no leitor.

Computacionalmente a geração de texto deve ser determinística e por isso deve-se ditar qual relação incluir e quando [Paris'91]. Vemos que as restrições descritas nas relações de [Mann&Thompson'87] são muito vagas para permitir que um sistema de geração as use para construir textos, buscando as informações apropriadas em uma base de conhecimento.

No Planejador proposto os esquemas são especificados pelo desenvolvedor da aplicação (um ser humano) que é capaz de tratar as indeterminações da RST. O problema de planejamento de textos se transforma, a rigor, em uma espécie de problema de classificação, onde o sistema busca atender um objetivo comunicativo através dos esquemas pré-especificados pelo desenvolvedor. Este fator voltará a ser discutido no capítulo 5 (Discussão).

O passo após o planejamento compreende a realização do texto. A próxima seção estuda a área de Geração de Linguagem Natural como um todo, observando o relacionamento entre o passo de planejamento e o de realização.

2.5 Geração de Linguagem Natural

"Um sistema capaz de selecionar informações relevantes para se atingir um determinado objetivo de discurso e organizar estas informações em uma apresentação coerente em linguagem natural que [potencialmente][2] atinja este objetivo é um sistema de planejamento de texto" [adaptado de Moore'95, pág. 81].

Ao se construírem sistemas de planejamento de textos devem ser tomadas decisões importantes cujo impacto poderá ser percebido no conteúdo, estrutura e estilo do texto gerado

[2] Palavra inserida pela autora

e tipos de interação possíveis. Uma destas decisões é a escolha da arquitetura de geração de textos. Ela pode ser sequencial, integrada, intercalada ou paralela. A arquitetura sequencial foi proposta por [McKeown'85] e aplicada por exemplo nos ambientes EES [Neches&Swartout&Moore'85, Swartout&Paris&Moore'91, Paris'91, Moore&Swartout'91, Moore'95] e LINX [García'95, Quental'95, Oliveira'95, Nunes'91]. Nesta arquitetura todo o texto é completamente planejado antes da realização.

A arquitetura hipotetiza que o estado das bases de informações usadas para geração de texto não se alteram desde o início do planejamento até o fim da linearização do texto. Ela é muito usada pois permite que se mantenha o foco de pesquisa em questões ou de determinação de conteúdo e organização do texto (*deep generation*) ou em questões de realização (*surface generation*), tais como estruturas lexicais e sintáticas. Como o Planejador aqui proposto se concentra somente nas questões de *deep generation* a arquitetura sequencial foi escolhida. A figura 7 mostra a arquitetura do explicador onde o Planejador pode ser aplicado:

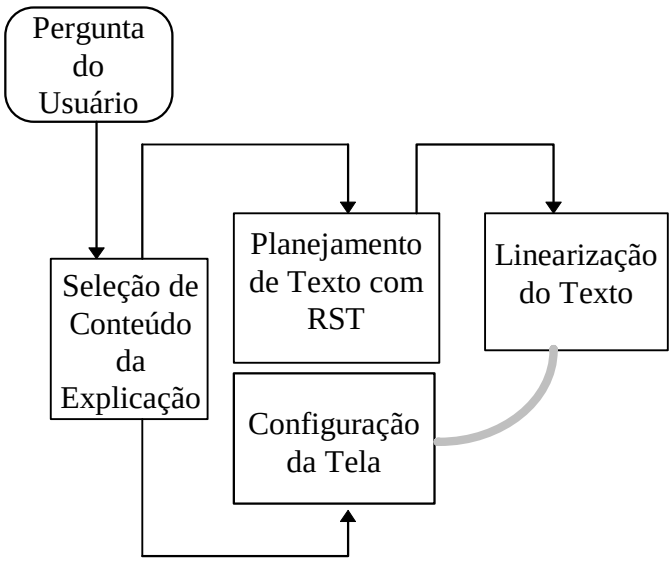


Figura 7: Arquitetura para o Planejador de Respostas Explicativas

Como o planejamento de todo o texto é feito antes de qualquer realização, a arquitetura tem a vantagem de não restringir a ordem do planejamento. Por exemplo, o final do texto pode ser

planejado antes do começo. Por outro lado isso impede linearizações incrementais de segmentos do texto acarretando dois problemas inter-relacionados. O primeiro é que o texto só poderá ser lido pelo usuário em sua integridade. Se uma parte do texto explicativo não for compreendida pelo usuário, a explicação pode se tornar inválida deste ponto em diante, visto que se baseará em algo que não foi entendido. O outro problema é que, sem linearizações incrementais, não existem pontos de parada onde o usuário possa interagir com o sistema e expressar que algo não ficou claro, o que solucionaria o primeiro problema.

Pela sua concepção a arquitetura sequencial, diferentemente das outras, se mostra incapaz de tratar as questões de *surface generation* ao longo do planejamento. Caso o objetivo seja simular um processo mental em computador cabe lembrar que, segundo [Hovy'88], pesquisas em psicolinguística sugerem que o modelo estritamente sequencial que separa planejamento e realização está incorreto.

Depois da arquitetura sequencial foi estudada a arquitetura integrada [Appelt'82, Appelt'83] onde as decisões sobre conteúdo e estrutura do texto e sobre forma linguística são tomadas em um único processo de planejamento. Nesta arquitetura todas as restrições têm a mesma forma de representação e são tratadas igualmente, implicando na busca por uma única representação capaz de sustentar informações de naturezas distintas, como por exemplo restrições de conhecimento de usuário, de foco e sintáticas. A distinção que se pode fazer entre as restrições se limita ao momento quando elas serão processadas. Normalmente, com o processo de planejamento hierárquico, faz-se com que as restrições sintáticas apareçam no final do processamento. O planejamento segue até o nível de determinação de palavras, o que por sua vez diminui o papel do realizador.

Estudando a análise desta arquitetura feita por [Hovy'88] verificamos que ela tem sobre a arquitetura sequencial a vantagem de permitir re-planejamento do texto, segundo [Appelt'83] "*unlimited backtracking*". No entanto, este recurso traz uma mesma limitação da arquitetura sequencial que só torna possível a linearização do texto após o seu completo planejamento. A capacidade do re-planejamento ilimitado afeta também o desempenho do sistema tornando-o

lento devido ao alto custo de processamento e, segundo [McKewon'85], levando a arquitetura a ter sido usada apenas para gerar textos pequenos.

Passamos então para o estudo da arquitetura intercalada onde o planejamento e a realização de texto são dois processos distintos, assim como na arquitetura sequencial. A diferença consiste no poder de comunicação entre os dois processos que permite transmissão de informação do planejador para o realizador e vice-versa, decorrendo disso a possibilidade de iteração e *backtracking*. A arquitetura intercalada abrange então os modelos sequenciais e integrados. Uma de suas distinções é apresentar a possibilidade de planejamento e realização incremental do texto a ser produzido. À medida que a construção de segmentos de texto se completam eles já podem ser apresentados aos usuários. No entanto, para isso ser possível, é imposta a restrição de que o planejamento seja feito na mesma ordem da linearização do texto, ou seja, o início do texto deve ser planejado antes do meio e assim por diante.

A arquitetura intercalada é particularmente útil para sistemas que constróem textos a partir de bases de informação cujo estado pode se alterar ao longo do planejamento. Por exemplo, em sistemas que possuem modelo de usuário, um novo conhecimento apresentado no meio de uma explicação pode causar uma mudança neste modelo que então afetará o planejamento posterior.

A arquitetura permite também que o usuário expresse uma dúvida no meio de uma explicação, entre a linearização de um segmento de texto e outro. A retirada da dúvida pode influenciar os planos futuros de explicação. Esta permissão de interrupção do usuário tem a vantagem de evitar que uma explicação prossiga fazendo construções baseadas em algo que não foi entendido. Este é o caso por exemplo do sistema de explicação EDGE [Cawsey'92]. Mas para tratar interrupções do usuário ela necessita também de um conjunto de heurísticas sobre como retornar à explicação anterior à dúvida do usuário. Em um caso extremo o usuário pode entrar em uma cadeia de dúvidas como por exemplo:

Imagine que um usuário que nunca teve conta em banco acabou de ganhar um cartão com código de barra. Ele pergunta então ao sistema de explicação como usar o cartão. Veja o diálogo decorrente:

Resposta 1 do sistema: “O cartão com código de barra funciona da mesma forma que o cartão magnético...”

Interrupção do usuário (dúvida) 1.1: “O que é um cartão magnético?”

Resposta 1.1.: “O cartão magnético é um identificador.”

Interrupção 1.1.1: “O que quer dizer ser um identificador?”

Resposta 1.1.1:.... “O identificador é o que distingue um cliente dos outros.”

... continuação da Resposta 1.1: “Como dizia antes o cartão magnético informa ao computador o número da sua conta de forma que você possa retirar dinheiro.”

... continuação da Resposta 1: Voltando à questão inicial, o cartão em código de barra funciona da mesma forma que o cartão magnético, passando-o pela leitora da esquerda para a direita.”

Este tipo de diálogo requer do sistema técnicas e heurísticas sofisticadas de forma a retornar ao que estava sendo explicado antes de forma natural e coerente.

A arquitetura intercalada convive com uma falta de conhecimento do texto que será gerado no futuro, o que acarreta alguns problemas. Um deles pode ser descrito através de exemplo retirado do sistema EDGE. O sistema afirmava: “Agora vou explicar *X*.”. Ao tentar planejar a explicação sobre *X* o sistema verificava que o usuário já conhecia *X* e portanto a explicação não era necessária e não era gerada. Isto acontecia pois o sistema não tinha como prever o tamanho ou a complexidade da explicação que viria em seguida: o que daria oportunidade para resolução de alguma incorreção, por exemplo através de re-planejamento ou *backtracking*.

Outro percalço relativo à falta de conhecimento do que será planejado no futuro é que muitas vezes uma pergunta feita pelo usuário no meio da explicação será explicada no futuro. Porém, como não se tem o plano completo da explicação futura o sistema pode acabar atendendo duas

vezes o mesmo objetivo comunicativo, no pior caso chegando a gerar o mesmo texto explicativo. Cawsey propõe como solução um planejamento mais extenso antes da linearização, embora seja difícil determinar qual seria seu tamanho ótimo, se ele deve ou não variar e em que circunstâncias.

[Hovy'88] lembra que a arquitetura intercalada não se restringe apenas à intercalação entre planejamento e realização, onde todo o planejamento de um segmento de texto é feito e depois realizado. Antes da linearização pode haver interação entre estes dois processos. Isto permite que o planejamento do conteúdo e organização do texto leve em consideração por exemplo informações comunicadas pelo realizador, tais como estruturas lexicais e sintáticas ou pedido de re-planejamento. Hovy propõe então a comunicação direta entre estes dois processos. Já [Scott&de Souza'90b] propõem o intercâmbio de informações entre os processos através de uma base de informação compartilhada, permitindo que o processo de planejamento tenha acesso a algumas informações do realizador e que o realizador atue sobre decisões que foram tomadas anteriormente pelo planejador. As autoras argumentam também que dessa forma pode-se evitar o replanejamento: "...visto que o planejador controla o tipo de alterações que podem ocorrer, não há necessidade de criticar o novo plano aperfeiçoado" pelo realizador [Scott&de Souza'90b, pág. 40]. Elas chamam esta forma de planejamento intercalado de Planejamento Conciliatório.

Existe ainda a arquitetura paralela. Nela as questões de *deep generation* e *surface generation* podem ser tratadas em processos executados paralelamente ou cada um deles pode ser dividido em sub-processos também executados simultaneamente. Esta arquitetura é análoga à intercalada com o adicional dos aspectos inerentes ao paralelismo, onde precisam ser definidos:

- quais processos ou sub-processos podem ser executados em paralelo;
- qual a prioridade e intervalo de tempo a ser alocado para cada processo ou sub-processo; e
- pontos de sincronismo e inter-dependência entre os processos.

Esta arquitetura foi simulada por exemplo por [De Smedt'90] onde o processo de determinação de conteúdo e organização de textos é executado paralelamente ao processo

realizador. Este último por sua vez é dividido em sub-processos que também são executados simultaneamente. Em particular o processo de formação de árvores sintáticas de segmentos de textos pode ser não-linear, ou seja, ramos da árvore são construídos paralelamente e depois unidos. O primeiro processo (*deep generation*) é iniciado e incrementalmente passa informações ao segundo. O realizador pode trabalhar paralelamente na construção de diferentes segmentos de textos.

Para a arquitetura poder fazer linearizações incrementais é preciso determinar em que pontos a linearização pode ser disparada, por exemplo no momento em que se identificar que a construção de uma sentença está completa. Como as árvores sintáticas não são montadas hierarquicamente é difícil prever por exemplo se uma sub-árvore será inserida no meio de outras duas que já estão prontas para linearização. Portanto são necessárias heurísticas que definam o fim de um sentença ou segmento de texto - tal como a mudança de tópico.

Cabe lembrar que a experiência de [De Smedt'90] visa apresentar um modelo computacional que simule o comportamento do falante humano, baseado na afirmação de que ocorre paralelismo no processo mental de construção da fala.

Com isso fechamos o estudo das arquiteturas, lembrando que o Planejador proposto estará inserido em uma arquitetura sequencial. Agora é preciso tomar outra decisão relativa à construção de sistemas de planejamento de texto. Deve-se definir como será a interação entre a determinação do conteúdo e a determinação da organização ou estrutura do texto.

O planejamento de texto direcionado pelas necessidades cognitivas e intencionais do usuário (representadas pelo modelo de crenças do sistema) e por questões psicolinguísticas que visam atender estas necessidades normalmente fazem um tratamento integrado de conteúdo e organização de texto. A partir de uma necessidade de geração de texto e de um modelo de crenças o sistema define um objetivo de discurso. A partir deste objetivo ele constrói o plano de texto intercalando raciocínio sobre conhecimento linguístico e busca na base de conhecimento do domínio. Neste caso os conhecimentos linguísticos predominam e ditam a

necessidade de acesso à base de conhecimento de acordo com o objetivo de discurso que visam atender.

É importante lembrar que a relação entre objetivos de discurso e os recursos linguísticos utilizados para alcançá-los estão sub-especificados na literatura de linguística estudada no que tange a aplicação em meio computacional. Um exemplo a ser citado é a RST que só permite a construção de textos via computador a partir da utilização de heurísticas pragmáticas que visam preencher o espaço vazio entre o conhecimento linguístico e as necessidades computacionais.

Este tipo de abordagem, por ser dominado pelas questões psicolinguísticas, garante maior independência em relação ao sistema (à base de conhecimentos, ao racionador utilizado, etc). O texto resultante porém reflete o processo do raciocinador linguístico que não necessariamente será adequado para esclarecer ao usuário como o raciocinador do sistema baseado em conhecimento funciona. Sob o ponto de vista semiótico, isto acarreta uma dificuldade na transmissão da mensagem do designer do sistema para o usuário.

Já o tratamento sequencial, como é o caso do ambiente LINX e do Planejador aqui proposto, procura formas linguísticas adequadas para apresentar de forma cognitivamente motivada o conhecimento e raciocínio utilizados pelo sistema. Ou seja, primeiro seleciona-se o conteúdo da mensagem através dos recursos disponíveis no sistema e partir dele escolhe-se a melhor forma de apresentação ou expressão da mensagem. A maior vantagem reside na potencial transparência do conteúdo da mensagem designer-usuário em detrimento da independência do tipo do sistema e futura possibilidade de reutilização em sistemas cuja representação de conhecimento e raciocinadores sejam distintos.

Para o tratamento sequencial o ideal é que a construção do planejador de respostas explicativas e da interface caminhe paralelamente à construção do raciocinador e da base de conhecimento [García'95]. Com isso pode-se avaliar os impactos que a tomada de decisão em um componente tem sobre os outros. Uma das questões que serão levantadas pela construção

da interface e do planejador será a utilização ou não de outros meios de expressão que não apenas a linguagem natural.

Esta questão surge a partir da observação do comportamento humano em que percebemos que usamos diferentes formas de expressão, além da linguagem natural, para transmitir uma mensagem. Isto traz para a área de planejamento de textos a necessidade de se avaliar se o planejamento das possíveis amostragens e interações com outros recursos (gráficos, sonoros, etc.) deve ser inserido no planejamento dos textos ou não.

O planejamento de texto independente dos outros recursos parece mais adequado para o caso onde o conteúdo apresentado pelos recursos não-LN é diferente daquele da resposta explicativa textual. Isto porque provavelmente a fonte de conhecimento também será diferente (complementar ou com alguma interseção) daquela utilizada para a geração do texto.

Uma vantagem do planejamento em separado é a praticidade de se suprimir o tratamento dos recursos não-LN quando desejado, bastando para isso não fazer chamada aos módulos do sistema responsáveis. Mais importante ainda é o fato desta supressão não afetar o processo de geração de texto.

Este mesmo modelo pode ser usado para a geração de recursos suplementares. Neste caso, como a fonte de conhecimento é a mesma para todos geradores (de texto, gráfico, etc.) o ideal é que se crie uma área de memória compartilhada que possa ser consultada por eles.

O Planejador proposto faz um planejamento separado para cada um dos recursos que se deseja utilizar. Atualmente são planejadas e geradas respostas explicativas nas formas textuais e gráficas. Esta abordagem apresenta também a vantagem se poder adicionar outras formas de expressão de forma modular e independente das já previstas.

Já no planejamento integrado o texto e os outros recursos são planejados intercaladamente em um mesmo processo. Esta abordagem é sugerida pelos estudos psicolinguísticos e foi utilizada por diversos planejadores [Appelt’82, Cawsey’92].

No caso de aplicações multimídia existe a vantagem de ser mais propícia a sincronização entre os diferentes recursos caso necessário. Na situação anterior seria necessária a comunicação entre os processos de planejamento. No entanto a falta de modularização dificulta a opção de se desativar um dos mecanismos de expressão, como por exemplo a expressão sonora, no caso do usuário não contar com os instrumentos computacionais que a viabilizem.

A saída gerada pelo planejamento integrado apresenta um dos seguintes requisitos:

- que o gerador consiga reconhecer a que tipo de expressão os componentes do plano resultante se referem (de forma a fazer a apresentação adequada para o usuário).
- que o planejador separe para geradores distintos cada um dos planos dos diferentes recursos de expressão. Com isso no entanto deve-se atentar para a garantia de coordenação e sincronismo entre os recursos caso sejam necessários.

O formato da saída do planejador é outra decisão a ser tomada durante a sua construção. A estrutura de informação gerada precisa ser compreendida pelo gerador de textos ou gerenciador de recursos gráficos.

No caso do Planejador proposto cada forma de expressão tem uma saída diferente, direcionada para o respectivo realizador. O plano de texto corresponde a uma sequência de atos de fala que deverão ser lidos pelo realizador e linearizados. O realizador atualmente faz um mapeamento dos atos de fala para *templates*, como será exemplificado no capítulo 3 (Solução). Atualmente o Planejador gerencia também recursos gráficos. Dependendo do conteúdo de explicação é gerado um plano para esta forma de expressão. O plano corresponde a um conjunto de funções que deverão ser executadas, sendo responsáveis pela atualização e configuração de interface.

A saída do Planejador em formato de ato de fala permite que a geração de texto possa ser melhorada, por exemplo se for utilizado um Planejador de Sentenças como o SPL usado por [Moore’95], mas no nosso caso para o português.

Com isso terminamos o estudo de embasamento teórico podendo passar então para a descrição da proposta do Planejador de Respostas Explicativas Baseado em uma Biblioteca de Esquemas RST.

3. A Solução

A partir dos requisitos citados na seção 1.1 (Requisitos do Planejador) e das necessidades que as diversas áreas estudadas nos apontam propomos o Planejador de Respostas Explicativas Baseado em uma Biblioteca de Esquemas RST.

Seguindo o Framework de Engenharia Semiótica, precisamos gerar as mensagens que queremos transmitir para os usuários do Sistema de Explicação. Estas mensagens servem como pano de fundo para as informações que serão passadas para os usuários finais através da interface. Elas descrevem, em alto nível, nosso interpretante sobre o que é o Planejador, o que ele pode ou não fazer e qual a sua utilidade. Ou seja, elas mostram o que pensamos sobre um módulo de Explicação que faz uso do Planejador de Respostas Explicativas aqui proposto. Elas regulam nossa solução e expectativas em relação a interpretantes dos usuários. Servem também como medidores do produto gerado sempre que os usuários nos informam sobre como eles estão entendendo as mensagens, ou seja, a própria interface. Veja exemplos das mensagens elaboradas:

"As respostas explicativas expõem o modelo do domínio da aplicação e seu funcionamento. Você pode ver os aspectos que foram entendidos como relevantes e como cada elemento do domínio foi automatizado de forma a auxiliar o seu trabalho."

"As respostas explicativas mostram como foi realizado o processo de resolução do problema previsto pelo domínio e como cada decisão foi tomada."

"A Explicação mostra como o domínio foi modelado e permite que sejam questionados aspectos dos componentes deste modelo."

"Os objetivos ulteriores das respostas explicativas são permitir que você valide o modelo desenvolvido e, sempre que este estiver correto, aumentar a taxa de aproveitamento de uso do sistema e consequentemente sua produtividade."

Estas mensagens têm reflexo no texto gerado. A primeira mensagem por exemplo explicita que houve automatização do domínio, ocasionando textos de resposta do tipo: "Este valor foi calculado **pelo sistema** conforme o método...." . Se a mensagem tivesse como objetivo

esconder a automatização a resposta provavelmente seria: “Este valor se deve ao método...”, acarretando provavelmente a interpretação de erro no domínio e não no modelo do sistema. Em termos semióticos isto mostra como o design de uma linguagem de interface depende da mensagem que se quer transmitir e da escolha de signos que invoquem um caminho de interpretação desejado pelo designer da linguagem. As mensagens servirão portanto para a definição de restrições de discurso (pragmáticas) e de sentido (semânticas) na construção dos esquemas RST. Ao se desenvolver uma aplicação é preciso refinar estas mensagens adaptando-as ao domínio específico que será atendido.

Podemos agora fazer uma especificação mais detalhada do Planejador. A figura 8 mostra um ambiente típico de aplicação do Planejador e seus componentes, seguindo a arquitetura de [McKeown’85] para geração de texto:

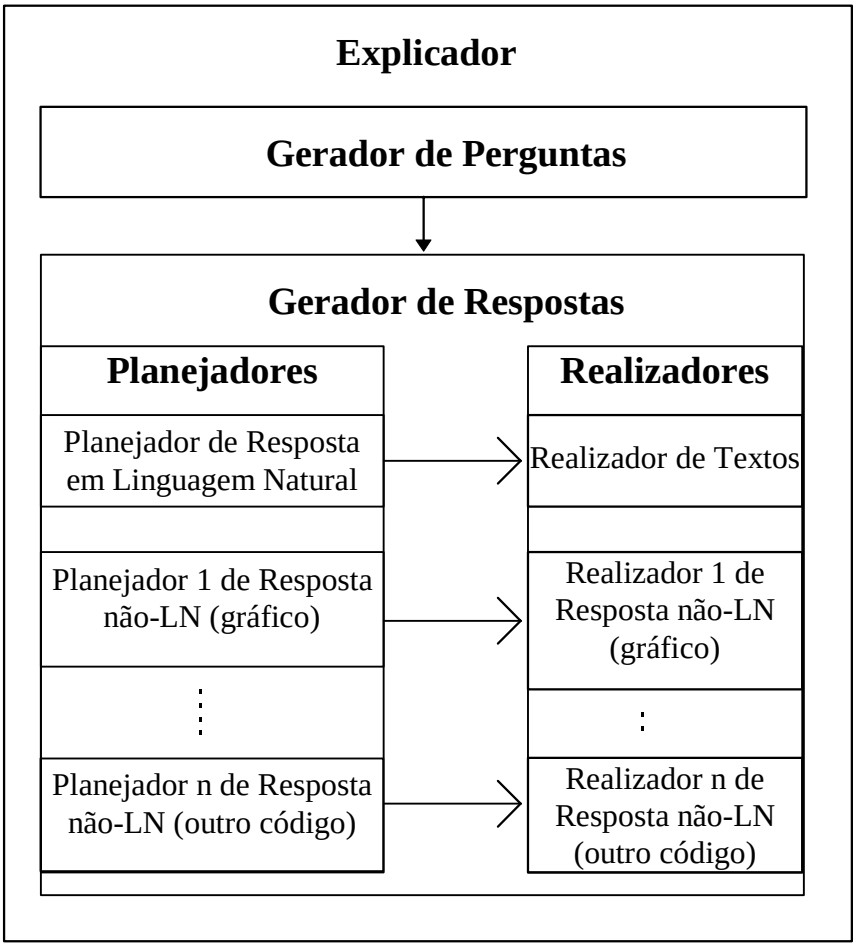


Figura 8: Ambiente de Inserção do Planejador e seus Componentes

A figura 9 apresenta as etapas do Planejador de Resposta em LN.

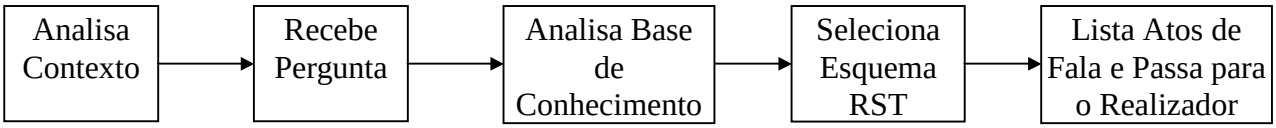


Figura 9: Etapas do Planejador de Respostas Explicativas em LN

A primeira tarefa do Planejador é analisar seu contexto de ativação. Este contexto está representado através de um identificador que reflete o ponto em que o usuário se encontra no modelo de tarefas do domínio. Compreendemos que esta noção de contexto pode ser expandida, como será visto no capítulo 5 (Discussão).

Na atual aplicação o Planejador pode ser acionado a partir de diferentes pontos. Embora ainda não implementado, ele pode ser adaptado, a baixo custo, para reagir de forma diferente dependendo do ponto onde foi chamado, seja gerando uma expectativa de pergunta, seja pela resposta explicativa gerada. Entendemos que diferentes pontos de acesso podem indicar diferentes intenções dos usuários, pois as chamadas provavelmente ocorrerão em diferentes momentos na resolução de um problema do domínio previsto pelo Sistema Baseado em Conhecimento. Diferentes pontos podem também fornecer mais ou menos informações para o Planejador - por exemplo um ponto de acesso pode já possuir algumas informações sobre a pergunta que o usuário irá fazer.

Reconhecido o contexto, o Planejador se prepara para receber a pergunta. Questões típicas de módulos de Explicação em Sistemas Baseados em Conhecimento são: “Por que?”, “Como?”, “Qual o valor?”, “Por que não?”, “E se?” entre outras. O Planejador atualmente responde as questões “Qual o valor?”, “Por que?” e “Por que não?”. Escolhemos a estratégia de desenvolvimento incremental onde foram estabelecidas prioridades entre as diferentes perguntas que se queria responder. O aspecto principal para decisão sobre as prioridades foi a disponibilidade de informações na base de conhecimento. Como veremos no capítulo 5

(Discussão), outras perguntas estão planejadas para serem respondidas futuramente. As perguntas atuais pretendem atender as necessidades cognitivas dos usuários quanto a:

- justificativa de porque o sistema chegou a determinados resultados através da pergunta “Por que?”.
- falha no atendimento de expectativas do usuário através da pergunta “Por que não?”.
- informação sobre o valor atual de entidades da base de conhecimento através da pergunta “Qual o valor?”.

O Planejador também recebe como informação da pergunta a entidade da base de conhecimento sobre a qual a questão deve atuar (por exemplo uma variável, um *frame* ou um objeto, atributos, métodos, regras ou mecanismos de inferência).

A definição de quais respostas devem ser dadas é dependente do domínio. Por isso, para saber as respostas de cada uma das perguntas é preciso fazer uma aquisição de conhecimento com os desenvolvedores e com os usuários da aplicação. Isto permite que se levante que conhecimentos extra devem ser codificados na base de conhecimento, assim como pode se formalizar uma estratégia de discurso que permitirá ao Planejador gerar as respostas adequadas [Paris’91]. Após a aquisição de conhecimento pode-se então finalizar a modelagem e implementação dos objetivos comunicativos e carregar a Biblioteca de Esquemas RST.

A partir das informações da pergunta o Planejador faz uma análise da base de conhecimento em busca das informações disponíveis para gerar uma resposta. Ele obtém então um objetivo comunicativo. A árvore mostrada na figura 10 esquematiza como seria a definição de um objetivo comunicativo:

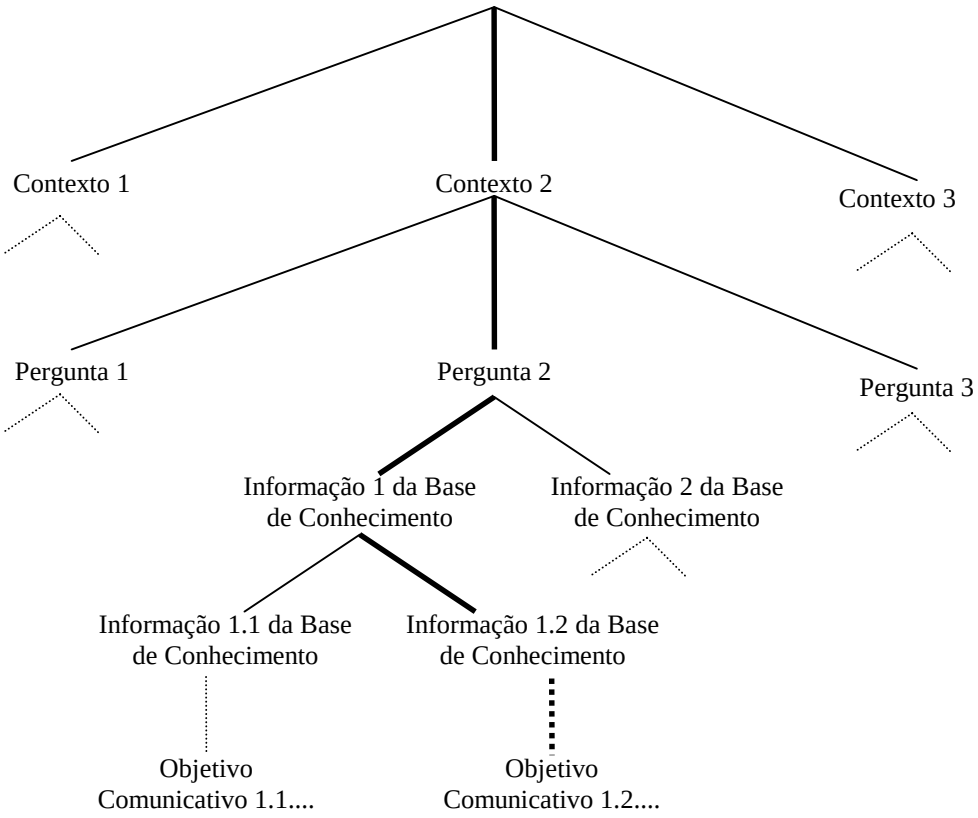


Figura 10: Árvore Genérica de Definição de Objetivos Comunicativos

A partir desta árvore conseguimos encontrar um objetivo comunicativo, ou seja, que efeito se deseja provocar no modelo mental do usuário. Este objetivo é uma descrição, tal como:

- Justificar a situação *X*; ou
- Comparar a situação *X* com a situação *Y*.

onde algumas partes da descrição são fixas e outras são preenchidas dinamicamente. Por exemplo a primeira descrição depois de preenchida poderia ficar assim: “Justificar a situação [contexto 1; informação 1...]”. Uma situação reflete o caminho percorrido na árvore de definição de objetivo comunicativo e indica as informações relevantes que futuramente serão usadas no planejamento e consultadas pelo realizador.

Definidos os objetivos comunicativos, deve-se planejar os esquemas RST que os atenderão. Isto é feito pelos desenvolvedores da aplicação juntamente com os usuários. Os esquemas RST garantirão a geração automática de textos coerentes para os padrões humanos.

Os esquemas RST são representados na biblioteca através de árvores que serão percorridas recursivamente pelo Planejador. Cada nó da árvore possui as seguintes informações:

- Efeito: descrição do objetivo comunicativo atendido pelo esquema ou sub-árvore do esquema.
- Núcleo(s): contém um ou mais atos de fala ou um ou mais objetivos comunicativos. No caso de objetivos comunicativos estes agora representam raízes de sub-árvores internas do esquema RST.
- Satélite(s): um nó pode possuir zero ou mais satélites. Um satélite também pode conter um ato de fala ou um objetivo comunicativo. Para o Planejador atual sua realização sempre acontece. No entanto ela poderia ser opcional, bastando para isso definir e considerar critérios.

Em termos computacionais, depois de escolhido o objetivo comunicativo, o próximo passo constitui uma pesquisa na Biblioteca de Esquemas RST em busca do esquema que atende a este objetivo comunicativo. Para isto é feita uma comparação entre a descrição do objetivo comunicativo e o conteúdo do campo efeito do esquema RST. Quando houver equivalência entre estas informações começa-se a percorrer recursivamente a árvore deste esquema RST. À medida que são encontrados atos de fala estes são colocados em uma lista que servirá de entrada para o realizador.

Veja como seria o planejamento do texto do exemplo utilizado na seção 2.4 (*Rhetorical Structure Theory*) cujo esquema RST é mostrado na figura 6 e cuja descrição é: “O valor do parâmetro *P* é *val(p)* pois assim foi determinado pelo projetista. Isto retifica o valor obtido

pelo sistema *val- (s)* e causa uma inconsistência com as heurísticas que podem ser vistas na Tabela de Heurísticas.
Veja no Histórico o contexto onde esta decisão foi tomada. Para maiores detalhes clique sobre os passos do Histórico."

Suponha que o usuário tenha feito a pergunta: "Por que o parâmetro *P* possui o valor *val(p)*?". Após o reconhecimento de contexto (que aponta para Questionamento de *Rationale*) e consulta de informações da base de conhecimento, o Planejador chega ao objetivo comunicativo: "Justificar a situação [contexto=Questionamento de *Rationale*; parâmetro=derivado; agente=projetista; método de cálculo=heurística; valor atual≠expectativa do sistema]. Direcionar atenção para a área de interface [Histórico]"

Uma busca na biblioteca de esquemas RST retorna o seguinte esquema, que corresponde a relação *Joint*:

Efeito: Justificar a situação [contexto=Questionamento de *Rationale*; parâmetro=derivado; agente=projetista; método de cálculo=heurística; valor≠expectativa do sistema].
Direcionar foco para a área de interface [Histórico].

Núcleos: Justificar a situação [contexto=Questionamento de *Rationale*; parâmetro=derivado; agente=projetista; método de cálculo=heurística; valor≠expectativa do sistema]. (objetivo comunicativo)
Direcionar foco para a área de interface [Histórico]. (objetivo comunicativo)

A árvore retórica resultante deste esquema é percorrida segundo o procedimento descrito abaixo:

****

Os ítens que possuem a marca "(objetivo comunicativo)" representam nós internos da árvore retórica e são expandidos pelo Planejador. O primeiro núcleo do esquema é expandido através de nova pesquisa na biblioteca e retorna a seguinte sub-árvore do esquema, representativa da relação retórica *Volitional Result*:

Efeito: Justificar a situação [contexto=Questionamento de *Rationale*; parâmetro=derivado; agente=projetista; método de cálculo=heurística; valor≠expectativa do sistema].

Núcleo: Informar que projetista causou a situação [agente=projetista] (ato de fala)

Satélites: Informar valor do parâmetro (ato de fala)

Informar consequência do atributo valor da situação [valor≠expectativa do sistema] (ato de fala)

Mostrar consequência do atributo método da situação [método de cálculo= heurística] (objetivo comunicativo)

Os atos de fala não precisam ser expandidos e são passados para uma lista que servirá de entrada para o realizador. O objetivo comunicativo “Mostrar consequência do atributo método da situação [método de cálculo=heurística]” ainda precisa ser expandido. O procedimento pesquisa a biblioteca de esquemas RST e obtém a sub-árvore que representa a relação *Evidence*:

Efeito: Mostrar consequência do atributo método da situação [método de cálculo= heurística]

Núcleo: Informar consequência do atributo método da situação [método de cálculo= heurística] (ato de fala)

Satélite: Indicar evidência do método [método de cálculo=heurística] (ato de fala)

Foi atingido um ponto onde não há mais objetivos comunicativos a serem expandidos para este ramo da árvore iniciado pela relação *Joint* (veja figura 6 na seção 2.4: *Rhetorical Structure Theory*). Parte-se então para a expansão do ramo direito, onde se encontra o esquema a seguir, também refletindo a relação *Joint*:

Efeito: Direcionar foco para a área de interface [Histórico]

Núcleos: Apontar foco para a área de interface [Histórico], que possui contexto de tomada de decisão (ato de fala)

Mostrar como buscar mais informação na área de interface [Histórico] (objetivo comunicativo)

Expandindo o segundo núcleo recupera-se a seguinte sub-árvore, respectiva à relação *Purpose*:

Efeito: Mostrar como buscar mais informação na área de interface [Histórico]

Núcleo: Informar conteúdo da interação com a área de interface [Histórico] (ato de fala)

Satélite: Informar como interagir com a área de interface [Histórico] (ato de fala)

Tendo chegado ao fim obtém-se como resultado uma lista de atos de fala que será passada para o realizador. Este por sua vez fará associação entre os atos de fala e *templates*. Os *templates* são preenchidos através de busca de informações na base de conhecimento. Os *templates* são então linearizados.

****

4. Aplicação

4.1 Domínio da Aplicação

4.1.1 Design de Engenharia em Grupo

Um artefato de engenharia representa uma resposta encontrada por um ou mais designers a um conjunto de especificações e necessidades atendendo a restrições identificadas. O passo anterior à construção de um artefato é o seu design. Documentos de design são gerados para informar uma descrição do artefato, instruções para a construção do mesmo, assim como o raciocínio utilizado durante o design, ou seja, o processo de decisão que levou a um design específico. Para a maioria dos projetos de design a quantidade de informação que deve ser documentada é grande.

Além das dificuldades de geração de todas as informações necessárias, estas precisarão ser recuperadas posteriormente por pessoas que podem não ter participado do processo de design. Cada uma delas possui um ponto de vista diferente, por exemplo, um proprietário possui uma perspectiva diferente das pessoas que irão construir o artefato.

Em projetos de design em grupo normalmente cada participante trabalha em uma parte diferente. Havendo interdependências entre as partes, algumas decisões precisam ser tomadas em conjunto e eles podem querer consultar informações uns dos outros durante todo o processo de design.

No ambiente em que estaremos trabalhando todo o grupo de designers trabalha em busca de um objetivo comum [Garcia'95]. Existem interdependências entre os diferentes componentes do grupo, ou seja, uma decisão de um integrante pode afetar o design de outro. Estas interdependências tendem a gerar conflitos entre eles pois nem sempre o ótimo local de um designer é o ótimo local do outro. No processo de negociação entre designers cada um procura defender seu ponto de vista, mostrando seu documento de design como base de argumentação

para se chegar a um acordo. Como o objetivo global é comum ao grupo como um todo, um designer deverá estar sempre disposto a sacrificar seu ótimo local em benefício da solução global.

4.1.2 Design de Planta de Processo para Plataformas Off-Shore

Nosso contexto de trabalho é o Design de Planta de Processo para Plataformas Off-Shore. A disciplina de Processo, no caso de uma plataforma localizada em alto mar (off-shore), é responsável pelo tratamento dado ao petróleo antes que o mesmo seja transferido para terra firme. O óleo bruto (uma mistura de óleo, gás e água) vem de um reservatório submarino e passa por um processamento na plataforma que finalmente exporta o óleo e o gás resultantes.

Esta disciplina é composta de dezenove subsistemas interligados, como mostrado na figura 11. Cada subsistema é responsável por uma etapa do processamento do petróleo. Por exemplo, temos o subsistema de Recebimento que é responsável por receber na plataforma o óleo bruto que vem de um reservatório. O subsistema de Aquecimento aumenta a temperatura do óleo bruto, de forma a facilitar o trabalho do subsistema de Separação, que separa o óleo da água e do gás por sedimentação. Após separado, o óleo é tratado (subsistema de Tratamento de Óleo) e o gás passa para o subsistema de Compressão de Gás que primeiro retira as impurezas que podem danificar o equipamento de compressão (subsistema de Tratamento de Gás) e então o comprime. A compressão do gás provoca um aumento de temperatura e se torna necessário um mecanismo de resfriamento (subsistema de Resfriamento). A água resultante precisa ser tratada, fazendo-se a retirada de óleo, antes de ser devolvida ao mar (subsistema de Águas Oleosas). Assim, o petróleo vai sendo processado até o subsistema de Bombeamento que o exporta para terra firme.

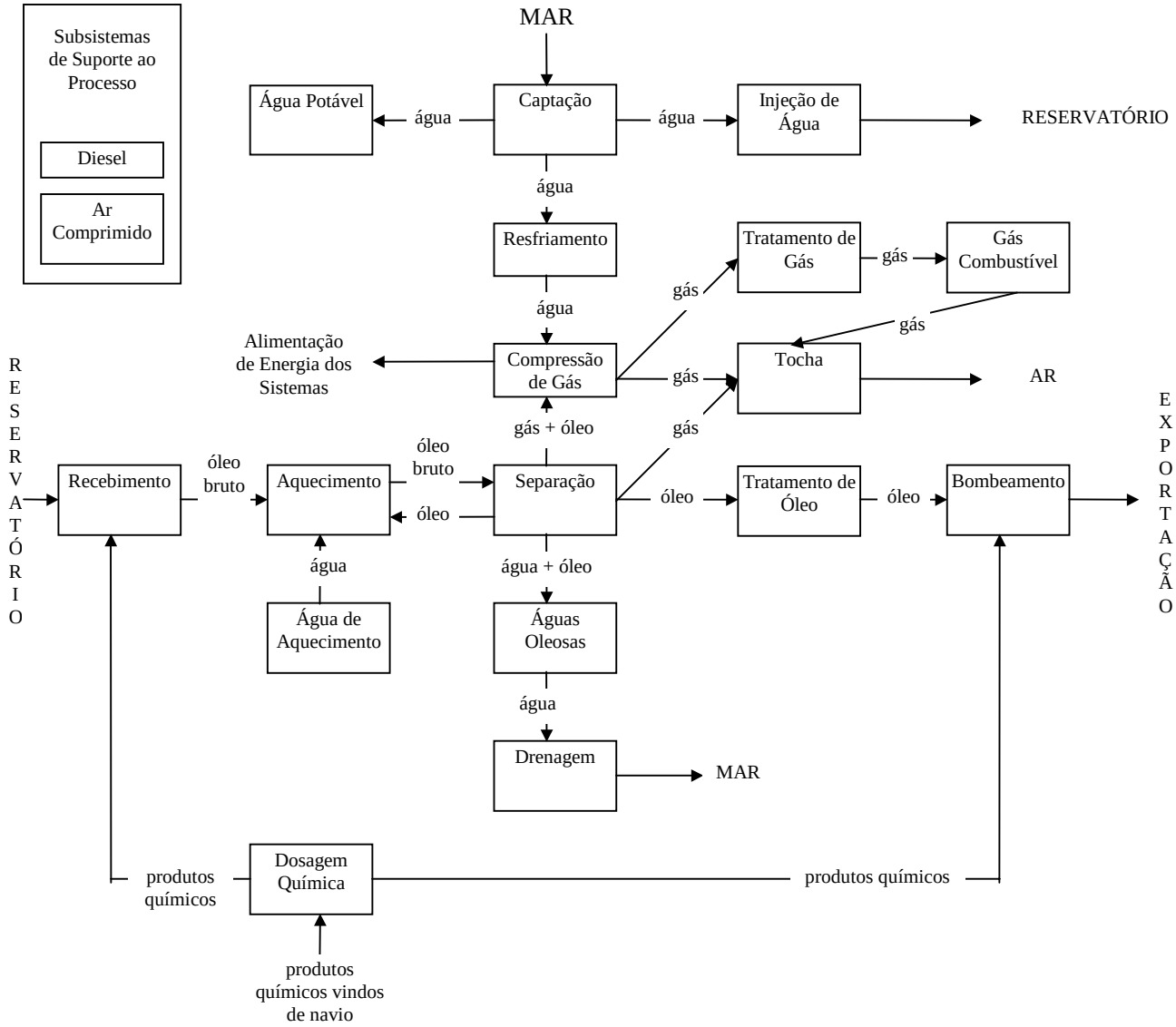


Figura 11: Subsistemas Componentes da Disciplina de Processo de Plataformas Off-Shore [Garcia'95]

Cada um dos subsistemas possui um designer responsável que procura encontrar a melhor solução local seguindo as restrições identificadas. No entanto, devido às interseções entre os subsistemas, a melhor solução local pode impactar um outro subsistema, gerando então um conflito entre eles. Estes conflitos devem ser resolvidos para o fechamento do projeto e então verificamos um processo de argumentação entre designers, onde cada um apresenta os

motivos que o levam à sua decisão. Se um conflito persistir por muito tempo, o coordenador do projeto pode intervir e determinar uma solução.

Pronto o design da disciplina de Processo, este é passado para a equipe de detalhamento que precisará entender as informações ali colocadas, assim como as decisões tomadas individualmente e em grupo, de forma a poder detalhar o projeto e prepará-lo para a construção.

4.2 Tecnologias Envolvidas: ADD e MultiADD

ADD [Garcia'92] é um modelo para desenvolvimento de sistemas inteligentes que auxiliam um processo de design gerando simultaneamente sua documentação. A documentação gerada consiste não só dos dados do projeto mas também do *rationale* do designer.

Um documento ativo de design contém um modelo inicial do artefato a ser desenvolvido e é capaz de gerar expectativas sobre as decisões do usuário. Ele acompanha os passos do designer e sempre que estes atendem suas expectativas o ADD consegue gerar automaticamente explicações para os mesmos. Assim o ADD consegue documentar as decisões de projeto sem nenhuma sobrecarga para o designer. No entanto, nem sempre as expectativas do sistema são atendidas. Neste caso, o ADD solicita ao designer que explique, ele mesmo, sua decisão. Isto pode ser feito de duas formas:

1. atualizando o conhecimento da base do ADD, o que permite que a decisão seja posteriormente explicada pelo próprio sistema
2. fazendo uma anotação em texto livre. Desta forma, como a decisão não possuirá uma explicação automática, o sistema se atém a mostrar a nota escrita pelo designer, no lugar de uma explicação completa

Como mostrado na figura 12, a arquitetura do ADD consiste de 3 elementos principais:

- interfaces - através das quais o designer interage com o sistema. São quatro os tipos de interface. A interface de Design permite que o usuário explore ou forneça uma solução para um problema. A interface de Explicação busca mostrar para o usuário o *rationale* aplicado

em determinado design. A interface de Anotações permite que o usuário coloque informações em texto livre que complementam o modelo do ADD. E a interface de Aquisição de Conhecimento permite que o usuário altere o modelo do ADD.

- base de conhecimento - contém o conhecimento referente ao domínio, ao processo de tomada de decisão e uma base de casos anteriores. O conhecimento é expresso através de uma rede paramétrica. Este elemento é responsável pela geração de expectativas do ADD.
- componentes de raciocínio - são responsáveis por manipular o conhecimento existente. Geram decisões de design e as comparam com as do usuário, pedem conhecimento adicional quando necessário e controlam o processo de documentação. São três os componentes de raciocínio. O Antecipador gera uma expectativa quanto ao valor de um parâmetro, quando o designer solicita ao sistema. O Propagador propaga alterações feitas pelo usuário, reavaliando as expectativas do sistema. E o Reconciliador compara a expectativa do ADD com o valor dado pelo designer e, caso sejam diferentes, chama o módulo de Aquisição de Conhecimento. O Reconciliador trabalha junto com o módulo de Explicação que interpreta a informação dada pelo usuário e apresenta o raciocínio do sistema. Baseado nisto, o usuário pode entender o modelo do sistema e, se necessário, alterá-lo adequadamente.

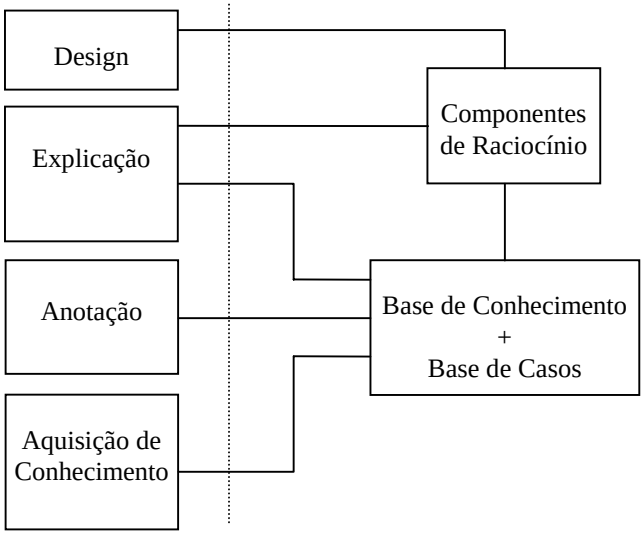


Figura 12: Arquitetura ADD (baseado em [Garcia’92])

O ADD funciona ao mesmo tempo como um assistente e um aprendiz do usuário. Um assistente no sentido de que ele pode sugerir soluções quando o usuário pedir, gerar documentação automaticamente e ainda explicar o *rationale* para estas soluções, quando as mesmas estiverem de acordo com as expectativas do sistema. O ADD funciona como um aprendiz quando o designer propõe uma solução diferente da esperada. O ADD pede então que o usuário atualize o modelo do ADD através da interface de Aquisição de Conhecimento, ou então justifique e explique sua decisão através da interface de Anotação.

Em um modelo ADD designers geram documentos para comunicar sua concepção do artefato. O documento de design final serve a diferentes pessoas com diferentes interesses em relação ao mesmo. Por exemplo, um engenheiro pode estar interessado em planejar a construção da planta desenhada e identificar equipamentos necessários para a construção, enquanto o gerente da construção pode estar atento a questões de custo.

As decisões tomadas no momento de design podem ser questionadas por aqueles envolvidos na etapa seguinte, por exemplo na etapa de construção. A função principal do módulo de Explicação é mostrar a estas pessoas os motivos que levaram a essas decisões. Atingido este objetivo, temos que a comunicação entre o designer e os usuários do documento ativo tende a diminuir, aumentando a produtividade de ambos. Quando a comunicação ainda é necessária, ela é qualitativamente melhor e mais enriquecedora, visto que todos têm conhecimento da decisão, do *rationale* e contexto que levaram à solução atual. Outro recurso importante da facilidade de Explicação é a capacidade de simulações. Antes de uma pessoa pedir uma alteração em um documento ativo, ela pode fazer uma simulação desta mudança, sem a presença do designer, e avaliar que impactos que seriam causados. Veja na figura 13 um esquema de interface simulando uma situação típica de uso da facilidade de Explicação de um ADD [Garcia’94].

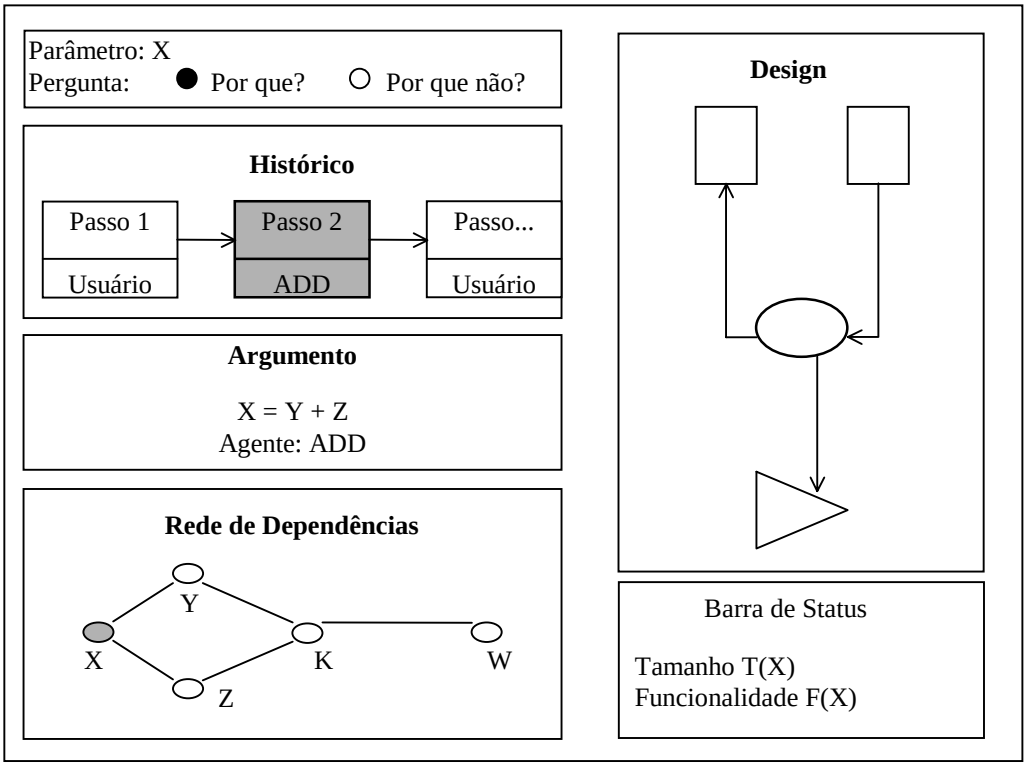


Figura 13: Esquema de Interface da Facilidade de Explicação em um sistema ADD

As versões iniciais do ADD [Garcia'92, Garcia'94] foram projetadas para auxiliar um designer em um domínio. Para lidar com design em grupo, que é o caso da aplicação ADDProc (Design de Planta de Processo de Plataformas Off-Shore) para a qual o Planejador foi desenvolvido, foi utilizada a proposta MultiADD de [Vivacqua&Garcia'96a e Vivacqua&Garcia'96b].

Nesta proposta o ADD é adaptado, se tornando um sistema multiagente e multiusuário. Cada subsistema da disciplina de processo, assim como o coordenador, é um agente representado por um designer e seu respectivo ADD. Existe um agente especial, o Controlador, que é responsável pela comunicação de conflitos entre os agentes envolvidos.

Este novo ambiente de ADD gera requisitos de adaptação ao módulo de Explicação. Suas atribuições anteriores continuam válidas, mas agora ele deve também atender a argumentação

entre os diferentes agentes que buscam a resolução de um conflito, durante o processo de design. Visto que este módulo junto com o sistema de Anotações mostram o *rationale* de cada decisão tomada, eles são mais do que adequados para realizarem esta função em conjunto [Vieira da Cunha&de Souza'97].

4.3 Sistema de Explicação do MultiADD

Antes de iniciar a construção do módulo de Explicação do MultiADD foi feito um levantamento de quais seriam suas funcionalidades, integrando as que já eram previstas por explicações do ADD com as do Planejador. Chegamos então ao seguinte conjunto de requisitos para o módulo de explicação do MultiADD:

- naturalidade, flexibilidade e sensibilidade: requisitos provenientes do Planejador. O requisito de naturalidade vai ao encontro da necessidade de se inserir na Explicação do ADD um texto em linguagem natural observada em [de Souza&Garcia'94, Garcia&de Souza'95 e Garcia&de Souza'97].
- fidelidade, suficiência e extensibilidade: requisitos que já eram atendidos pelo ADD e que também estão previstos pelo Planejador.
- expressão de respostas explicativas usando códigos gráficos e textuais: requisito proveniente de sistemas ADD que precisa ser previsto pelo Planejador.

Em relação aos requisitos de respostas explicativas verificamos que o ADD já atendia quanto à coerência e relevância. Com o uso do Planejador pretendemos melhorar os aspectos compreensão e correção quanto ao registro e tom conversacional.

Como visto em [de Souza&Garcia'94, Garcia&de Souza'95 e Garcia&de Souza'97], somente a amostragem das bases de conhecimento do ADD, ou seja da Rede de Dependências, do Histórico de Design e dos valores do parâmetros, não são suficientes para garantir o uso e entendimento adequados do documento apresentado.

Sob o ponto de vista semiótico, estes recursos não esclarecem completamente a mensagem do desenvolvedor para o usuário, pois a forma de expressão utilizada reflete a tecnologia aplicada que não necessariamente é dominada pelo usuário. Para usuários que ainda não aprenderam a interpretar esta forma de expressão a presença de uma outra pessoa que os ensine (que não o desenvolvedor: por exemplo um outro usuário um pouco mais experiente) traz a desvantagem de que esta pessoa pode não possuir um interpretante convergente com o do desenvolvedor e consequentemente poderá direcionar o usuário para um interpretante inadequado. Cabe lembrar que estamos aqui nos baseando no fato de que usuários tendem a buscar esclarecimentos através de contato com seus colegas de trabalho em não em sistemas de ajuda convencionais [Lang&Auld&Lang'82, Lang&Lang &Auld'81, Scharer'83].

A solução então é que o desenvolvedor inclua na sua mensagem um conteúdo com função tutorial para os outros códigos. Isto implica na necessidade de que este conteúdo seja expresso em um código dominado tanto pelo usuário quanto pelo desenvolvedor - o texto em linguagem natural. Vários autores em [Maybury'93] enfatizam o papel da linguagem natural como auxílio no sentido de se atingir objetivos de comunicação. O uso da LN colabora para que sejam atendidas as necessidades cognitivas de um usuário de Explicação do ADD [de Souza&Garcia'94, Garcia&de Souza'95 e Garcia&de Souza'97] e permite que se transmita fielmente a mensagem do desenvolvedor.

É função deste texto responder a pergunta realizada pelo usuário e adicionar uma estruturação e organização retórica às informações mostradas. Ele sinaliza a relevância e relacionamento entre os diferentes recursos que expressam o conhecimento contido no ADD. Com isso possibilitaremos que usuários explorem toda a riqueza dos recursos de conhecimento disponíveis em um documento ativo, assim como diagnostiquem erros nas suas concepções sobre o modelo e histórico de design.

Cabe lembrar que a Explicação é um módulo na aplicação MultiADD e que é importante que a mensagem por ela transmitida seja coesa e consistente com as dos diferentes módulos. Ou seja, deve haver um *continuum* semiótico entre os diferentes componentes do sistema.

Devido ao requisito do MultiADD de que explicações sirvam também como auxílio no processo de negociação entre usuários com soluções conflitantes, percebemos que o Framework de Engenharia Semiótica [de Souza'93] precisa ser adaptado. Neste caso, as explicações são, além de mensagens de desenvolvedores para usuários, mensagens de usuário para usuário [Vieira da Cunha&de Souza'97]. Ou seja, um usuário usa sua explicação para transmitir informações necessárias a uma argumentação que sustente sua decisão. O Framework adaptado seria então representado como mostrado no Cenário do 2 da figura 14, que é uma extensão do Cenário 1.

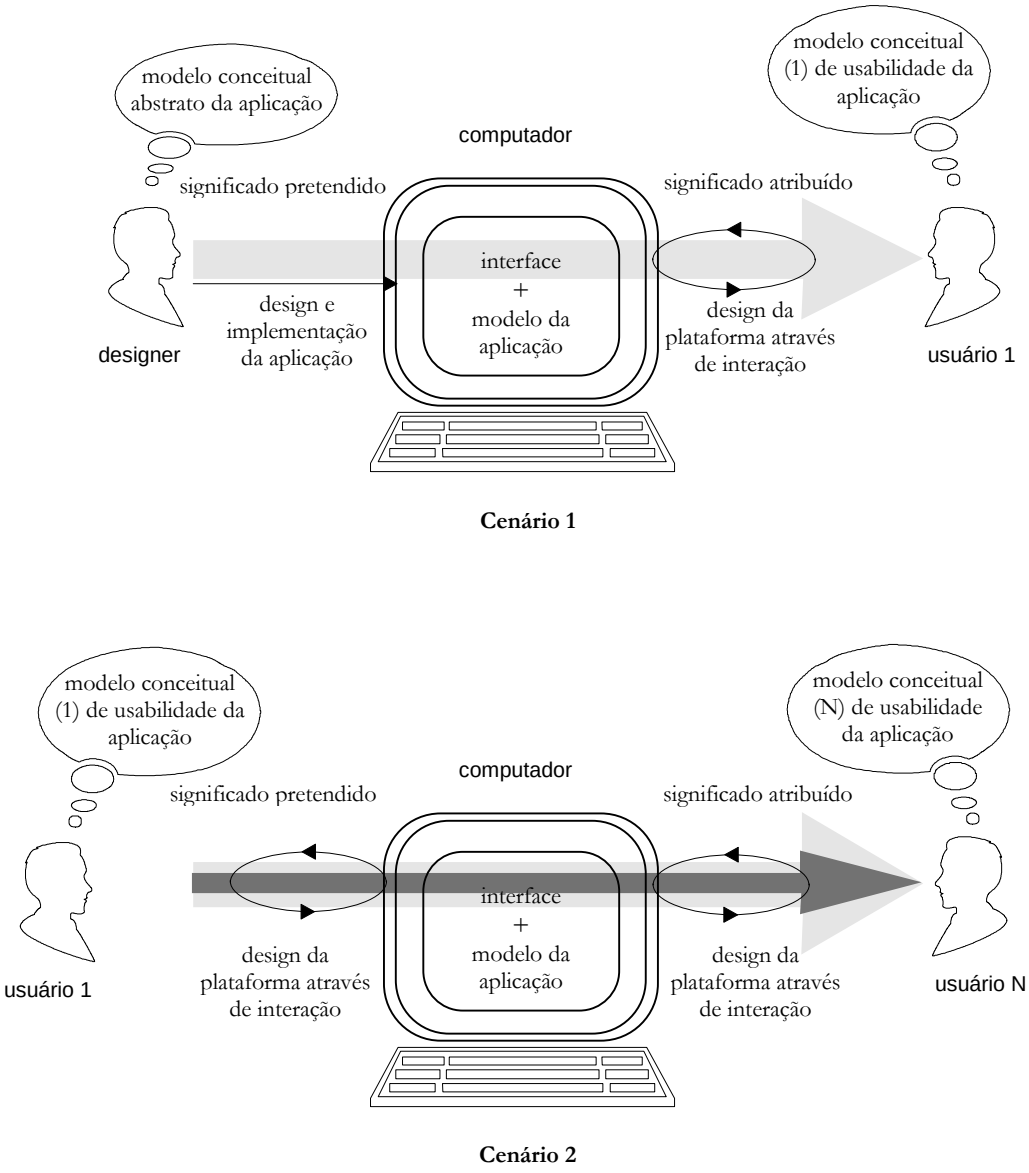


Figura 14: Framework de Engenharia Semiótica adaptado para o MultiADD

A função principal da Explicação como recurso de negociação entre os usuários do MultiADD é esclarecer para os designers envolvidos em um conflito os motivos que levaram cada um a tomar a decisão escolhida, mantendo-se o foco nas informações relevantes. Acreditamos que estes fatores incentivem um comportamento colaborativo, principalmente porque o código

utilizado da Explicação do MultiADD foi desenhado pelo desenvolvedor com este objetivo. Ele, por si só, pode evitar algumas situações de competitividade, como por exemplo o ocultamento de informações.

No entanto este código pode não expressar plenamente a mensagem de um usuário para os outros e por isso está disponível também o recurso de Anotação. Cabe lembrar que a utilização deste recurso pode afetar o processo colaborativo de negociação, por exemplo através de desvio de foco.

A frequente utilização de anotações pode ser um indício de que o código da Explicação não é expressivo o suficiente para transmitir as mensagens dos usuários. Todas estas informações sobre o papel da Explicação no processo de negociação, embora intuitivamente plausíveis, ainda precisam ser validadas a partir de um uso mais extensivo deste recurso e estudos aprofundados na área de trabalho cooperativo em meio computacional.

Voltando ao desenvolvimento do MultiADD, no processo de aquisição de conhecimento buscamos uma especificação inicial de como seria o seu módulo de Explicação e a utilização do Planejador de Respostas Explicativas. Utilizando as mensagens do desenvolvedor do Planejador descritas no capítulo 3 (Solução) e criando algumas específicas para o MultiADD chegamos a um conjunto de conteúdos da mensagem que queremos passar para os usuários finais.

Veja alguns exemplos:
“O módulo de Explicação expõe o modelo do domínio da aplicação e seu funcionamento. Você pode ver os aspectos foram entendidos como relevantes e como cada elemento do domínio foi automatizado de forma a auxiliar o seu trabalho.”
“A Explicação mostra como o domínio foi modelado e permite que sejam questionados aspectos dos componentes deste modelo.”
“Você pode consultar a Explicação dos outros usuários do sistema e eles podem consultar a sua interface de Explicação. Um dos objetivos deste recurso é melhorar o processo de negociação durante a resolução de conflitos.”

"No MultiADD o conhecimento está representado através de parâmetros. Cada parâmetro possui seu próprio valor e a Explicação mostra como cada valor foi obtido."
"Os vários parâmetros da base conhecimento estão conectados uns com os outros. Estas conexões são direcionadas e dado um parâmetro você pode ver de quem ele depende e quem ele impacta. É o que chamamos de Rede Paramétrica."
"A Explicação mostra como foi realizado o processo de resolução do problema previsto pelo domínio e como cada decisão foi tomada."
"O Histórico mostra os passos de design que foram realizados até o momento. Consultando suas informações você pode conhecer o contexto no qual as decisões de design foram tomadas."
"O modelo de relevância para informação sobre os recursos de conhecimento disponíveis é o seguinte:

- *quando o valor de um parâmetro tiver sido dado pelo usuário é importante visualizar o Histórico de design que mostra o contexto que influenciou esta decisão.*
- *quando o valor tiver sido informado pelo sistema, ocorre o seguinte: para parâmetros primitivos, deve-se mostrar o valor default e o Histórico que indicará que outro parâmetro levou à atribuição do valor default pelo sistema; para parâmetros derivados é importante mostrar a fórmula ou heurística aplicada, assim como a Rede de Dependências que confirma o modelo de conhecimento; para parâmetros decididos deve-se mostrar as avaliações das diferentes alternativas de acordo com as restrições e critérios previstos. Aqui também a Rede de Dependências confirma o modelo de conhecimento."*

"Os objetivos ulteriores do módulo de Explicação são permitir que você valide o modelo desenvolvido e, sempre que este estiver correto, aumentar a taxa de aproveitamento de uso do MultiADD e consequentemente a produtividade no processo de design e documentação."

Aprofundando o processo de aquisição de conhecimento, partindo de entrevistas abertas para entrevistas focadas, foi feito um levantamento de quais poderiam ser os contextos de chamada do módulo de Explicação. Verificaram-se três contextos distintos. O contexto de design, chamado de Questionamento de *Rationale*, corresponde à chamada do módulo por um usuário que procura explicações sobre o seu próprio design ou de outro usuário. Neste contexto a Explicação permite que se explore as bases de conhecimento, mostrando decisões que já foram tomadas. Outro contexto é o de Reconciliação. O Reconciliador é ativado quando um valor informado pelo usuário é incompatível com a expectativa do MultiADD e através dele

permite-se a ativação da Explicação. Neste caso, o valor que o usuário deseja informar ainda não foi de fato atribuído e a Explicação deve tentar mostrar as diferenças entre o valor que o usuário deseja e a expectativa que o sistema tem. Além disso, a Explicação deve incentivar o uso do recurso de Aquisição de Conhecimento. O terceiro contexto é o de Resolução de Conflitos. Quando o usuário é avisado da existência de um conflito ele tem a opção de chamar a Explicação respectiva a este conflito. Visto que estamos em um ambiente cooperativo, a Explicação deve incentivar o acordo entre os agentes em conflito.

Em termos implementacionais o Planejador reconhece atualmente o contexto de Questionamento de *Rationale*. Embora a Explicação já possa ser acionada a partir do ambiente de Reconciliação, o Planejador implementado ainda não está sensível a este contexto e gera os mesmos conteúdos de resposta daqueles apresentados no contexto de Questionamento de *Rationale*. A chamada da Explicação em contexto de Negociação ainda não é possível. No entanto, o Planejador foi construído de forma que a expansão de sensibilidade a contextos seja realizada a um baixo custo, como será mostrado no Apêndice A (Aspectos de Implementação).

Outros determinantes do conteúdo da resposta são a pergunta que o usuário escolheu e sobre que entidade da base de conhecimento ela se refere. Quanto a entidades da base de conhecimento, o Planejador atualmente permite que se façam perguntas sobre seus parâmetros. Como será visto no capítulo 5 (Discussão) poder-se-ia permitir perguntas sobre outras entidades, tais como os mecanismos de inferência. Maiores detalhes sobre os parâmetros serão informados a seguir.

Para a aplicação MultiADD estão previstas as perguntas “Qual o valor?”, “Por que?” e “Por que não?”. Outras perguntas requisitadas pelo MultiADD, tal como “E se?” para simulações, estão previstas para trabalhos futuros.

A pergunta “Qual o valor do parâmetro *X*?” expõe o valor atual dos parâmetros da base de conhecimento. Cabe lembrar que no MultiADD outra forma de se obter esta informação é através de manipulação direta sobre a área de Rede de Dependências (veja figura 18). No

entanto, a Rede de Dependências pode em determinado momento não mostrar o parâmetro sobre o qual o usuário quer fazer a pergunta. Desta forma, a pergunta fora do ambiente da Rede de Dependências apresenta menor distância articulatória.

A pergunta “Por que o parâmetro *X* possui o valor *(v)*?” justifica porque o sistema configura um determinado resultado. E a pergunta “Por que o parâmetro *X* não possui o valor *(v')*?” explica porque não seria possível atender a expectativa do usuário da Explicação.

Com a definição de quais perguntas deveriam ser respondidas passou-se para a documentação de que informações estão disponíveis na base de conhecimento. Seguindo o requisito de fidelidade proposto por [Moore'95], verificamos que o Planejador deve ser capaz de reconhecer os parâmetros da base e todos os estados em que eles podem se encontrar.

Assim sendo, podemos gerar textos mais ricos e mais precisos e consequentemente ganhar a confiabilidade do usuário, dado que a resposta por ele obtida reflete fielmente o que acontece no sistema.

Buscando atender a este requisito, levantamos todos os estados possíveis em que um parâmetro pode se encontrar. A Rede Paramétrica possui três tipos de parâmetro:

- primitivos - parâmetros informados pelo usuário. Se o sistema precisar de um valor para este parâmetro e nada tiver sido informado pelo usuário, ele pode usar um valor *default*
- derivados - parâmetros calculados pelo sistema através de fórmulas ou heurística. Se o usuário quiser, ele pode informar seu próprio valor. O Reconciliador será ativado, checando este valor contra a expectativa do sistema
- decididos - parâmetros cujo valor é selecionado a partir de um conjunto de alternativas. As alternativas são avaliadas segundo restrições e critérios.

Veja na figura 15 um exemplo de um segmento de uma Rede Paramétrica.

Águas Oleosas

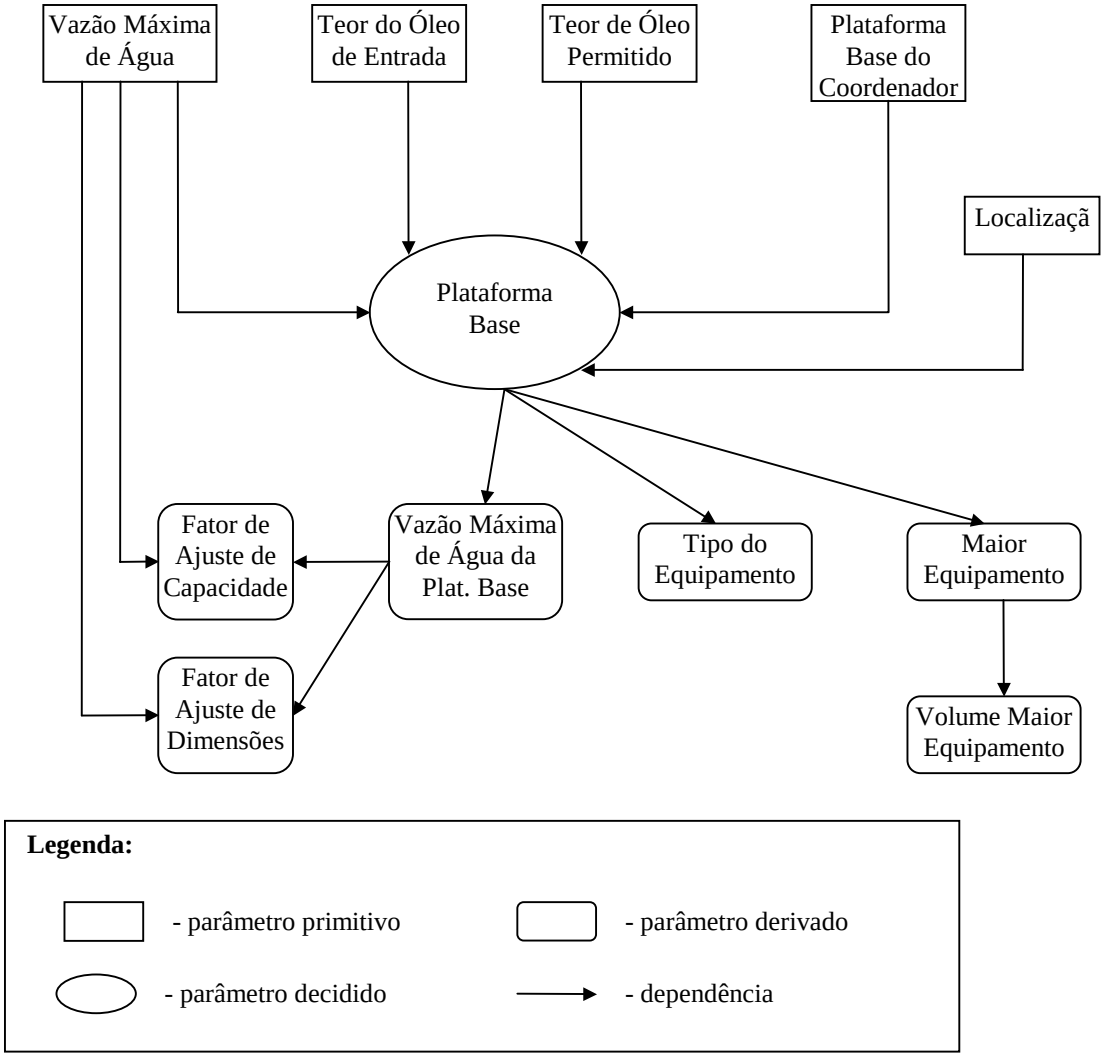


Figura 15: Exemplo de Rede Paramétrica

Para os diferentes tipos de parâmetros foram levantados todos os estados possíveis. Por exemplo, suponha um parâmetro do tipo Decidido. Cada tupla <valor, responsável, avaliação> da tabela da figura 16 exemplifica a representação de alguns estados em que este tipo de parâmetro poderia se encontrar:

Estado	Valor do Parâmetro Decidido	Responsável pelo Valor	Avaliação de Alternativas
Inicial	Nenhum	Nenhum	Não Realizada
2	Nenhum	ADD	Nenhuma foi selecionada
. . .	. . .	. . .	. . .
7	Sim	ADD	Uma alternativa foi selecionada
8	Sim	ADD	Um subconjunto de alternativas foi selecionado
9	Sim	ADD	Todas as alternativas são igualmente válidas
. . .	. . .	. . .	. . .
19	Sim	Usuário para Resolução de Conflito	Nenhuma foi selecionada
20	Sim	Usuário para Resolução de Conflito	Uma alternativa foi selecionada e é a mesma que o usuário escolheu
21	Sim	Usuário para Resolução de Conflito	Uma alternativa foi selecionada e é diferente da que o usuário escolheu. A alternativa do usuário passou pelas restrições
. . .	. . .	. . .	. . .
Estado	Valor do Parâmetro Decidido	Responsável pelo Valor	Avaliação de Alternativas
. . .	. . .	. . .	. . .
27	Sim	Usuário para Resolução de Conflito	Todas as alternativas são igualmente válidas, mas o usuário escolheu um valor que não pertence a este conjunto

Figura 16: Tabela de Exemplos de Estados de Parâmetros do tipo Decidido

Neste momento o processo de aquisição de conhecimento apresentou como resultado um mapa das diversas situações onde diferentes respostas explicativas deveriam ser geradas. O próximo passo foi definir os objetivos comunicativos de cada uma delas, planejar, construir e armazenar em biblioteca os respectivos esquemas RST.

A árvore de decisão na figura 17 mostra um caminho possível de determinação de objetivo comunicativo no MultiADD.

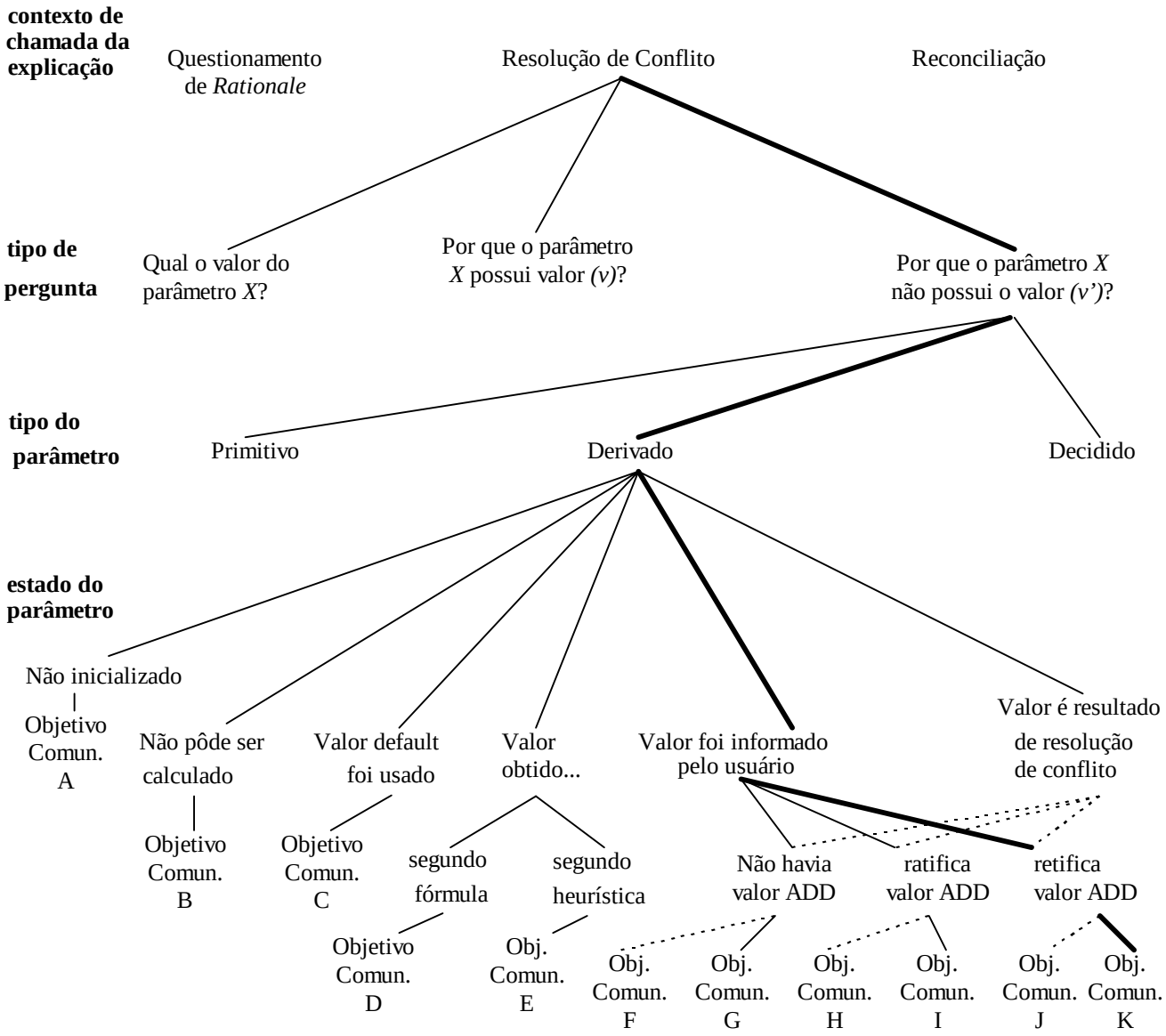


Figura 17: Exemplo de Árvore para definição de Objetivos Comunicativos do MultiADD

Continuando o processo de aquisição de conhecimento levantamos que respostas deveriam ser dadas em cada uma das situações previstas. Para cada situação foi levantada a meta comunicativa do desenvolvedor. Durante as transformações destas metas em esquemas RST foi necessário observar que os esquemas sendo construídos eram coerentes com os conteúdos das mensagens de alto nível que já haviam sido especificadas.

As mensagens de alto nível portanto eram transformadas em restrições de discurso (pragmáticas) e de sentido (semânticas) na construção dos esquemas RST. Por exemplo, a mensagem:

"O módulo de Explicação expõe o modelo do domínio da aplicação e seu funcionamento. Você pode ver os aspectos foram entendidos como relevantes e como cada elemento do domínio foi automatizado de forma a auxiliar o seu trabalho."

apresentou como resultado algumas heurísticas, tais como:

Para perguntas "Por que?" e "Por que não?" sempre devem ser mostradas as formas como os valores atuais dos parâmetros foram obtidos e se relacionam com o modelo previsto. No caso de parâmetros primitivos, a seguinte restrição deve ser atendida:

- *caso o valor atual seja o valor default, deve-se esclarecer este fato e expor que este foi aplicado devido à falta de informação do usuário.*

  Esta restrição, quando satisfeita, resulta no seguinte texto para a pergunta "Por que?":

  "O valor do parâmetro *P* é *(v)* porque este é seu valor default, que foi utilizado pois nenhuma informação foi fornecida pelo projetista."

  Ela impede a construção do seguinte texto:

  "O valor do parâmetro *P* é *(v)* devido à atribuição feita pelo ADD."

  Esta realização poderia causar no usuário a interpretação de que o valor do parâmetro não precisa ser informado, pois será sempre obtido pelo sistema.

No caso de parâmetros derivados devem ser observadas as seguintes restrições:

- *...*
- *caso o valor atual tenha sido informado pelo projetista, deve-se esclarecer qual era a expectativa do sistema, porquê, e se ela foi retificada ou ratificada.*

Esta restrição, quando satisfeita, resulta por exemplo (para a pergunta “Por que?”) no texto de exemplo já citado:

“O valor do parâmetro *P* é *(v)* pois assim foi determinado pelo projetista. Isto retifica o valor obtido pelo sistema *(v')* e causa uma inconsistência com as heurísticas que podem ser vistas na Tabela de Heurísticas. ...”

Ela impede a construção do seguinte texto:

“O valor do parâmetro *P* é *(v)* pois assim foi determinado pelo projetista.”

Esta realização poderia causar no usuário a interpretação de que não havia expectativa do sistema, o que obscureceria a tecnologia ADD.

- *...*

....

Em linhas gerais podemos dizer que para a pergunta “Qual o valor do parâmetro *X*?” a meta comunicativa do desenvolvedor do MultiADD é que a resposta seja objetiva e indique o Histórico de Design. Para a pergunta “Por que o valor do parâmetro *X* é *(v)*?” a meta é que o texto mostre o conhecimento utilizado para se chegar ao valor do parâmetro, no caso de o valor ter sido obtido pelo MultiADD. Caso o valor tenha sido informado pelo projetista deve-se fazer uma comparação com a expectativa do sistema. Já para a pergunta “Por que o valor do parâmetro *X* não é *(v')*?” o texto deve fazer uma comparação entre o valor perguntado e o atual. No caso de o valor ter sido obtido pelo MultiADD deve-se referenciar a área da Rede de Dependências que mostra os parâmetros que influenciaram no cálculo do que está em foco. Se o valor tiver sido informado pelo usuário deve-se checar se havia ou não expectativa do sistema para o valor do parâmetro. Se havia expectativa, seu valor deve ser comparado tanto com o valor atual da base (informado pelo usuário) quanto com o valor perguntado. Quanto à área de interface, deve-se enfatizar o Histórico que mostra o contexto onde o usuário tomou a decisão.

É necessário planejar também a atualização dos outros recursos de interface. Mesmo que o texto enfatize ora um recurso, ora outro, todos eles devem ser atualizados para que se mantenha a coesão da resposta explicativa apresentada. O Histórico sempre é atualizado focalizando o parâmetro e mostrando o contexto no qual ele recebeu o valor. A Rede de

Depêndencias também focaliza o parâmetro perguntado e mostra aqueles que podem ter influenciado a obtenção de seu valor. No caso das perguntas “Por que...?” e “Por que não...?” de parâmetros decididos, a Tabela de Avaliação de Alternativas deve ser mostrada. As mesmas perguntas, quando realizadas para um parâmetro derivado cujo valor é obtido através é heurística, devem apresentar na resposta a Tabela de Heurísticas.

Seguindo as heurísticas apresentadas, foram construídos e carregados em biblioteca os esquemas RST. O Planejador de recursos gráficos também foi construído. Veja agora um exemplo de como seria o processo de geração automática de uma resposta explicativa:

Suponha que um usuário está em contexto de Questionamento de *Rationale*, onde ele quer saber por que o valor de determinado parâmetro, de outro subsistema, é diferente do que ele gostaria. O usuário faria então a seguinte pergunta:

“Por que o parâmetro *X* não possui o valor *(v’)*?”

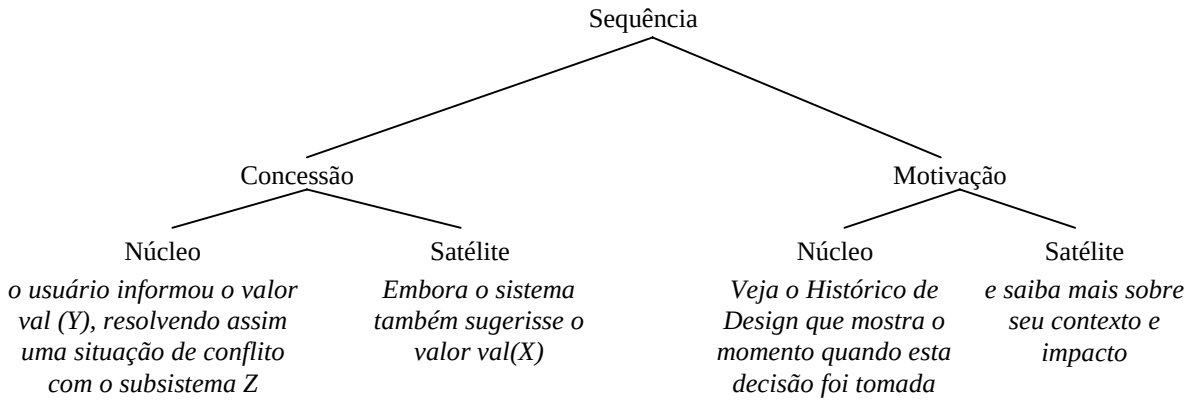
Suponha que este é um parâmetro do tipo Decidido que se encontra no estado de número 21 como mostrado na tabela da figura 16. Seguindo a sequência de planejamento e depois geração de texto temos o seguinte:

- identificação do objetivo comunicativo considerando:
    1. o contexto - Questionamento de *Rationale*
    2. a pergunta realizada -“Por que o parâmetro *X* não possui o valor *(v’)*?”
    3. o estado em que o parâmetro se encontra

Estado	Valor do Parâmetro Decidido	Responsável pelo Valor	Avaliação de Alternativas
21	Sim	Usuário para Resolução de Conflito	Uma alternativa foi selecionada e é diferente da que o usuário escolheu. A alternativa do usuário passou pelas restrições

- objetivo comunicativo Z encontrado para esta configuração reflete que nossa intenção é mostrar o motivo que levou o parâmetro X a possuir o valor (v) e enfatizar o contexto onde a valoração aconteceu (o Histórico é área para a qual se quer direcionar o foco).

- busca do esquema RST na biblioteca, resultando no seguinte esquema:



- realização da lista de atos de fala e demonstração do texto de resposta gerado. O texto em LN gerado seria então: "Embora o sistema também sugerisse o valor *val(X)*, o usuário informou o valor *val(Y)*, resolvendo assim uma situação de conflito com o subsistema *Z*. Veja no Histórico de Design mostra o momento quando esta decisão foi tomada e saiba mais sobre seu contexto e impacto."

- atualização dos recursos de interface

Veja como seria o esquema de interface de Explicação no MultiADD agora seguindo o mesmo exemplo usado na figura 6 (seção 2.4), que mostra a estrutura retórica do texto em LN.

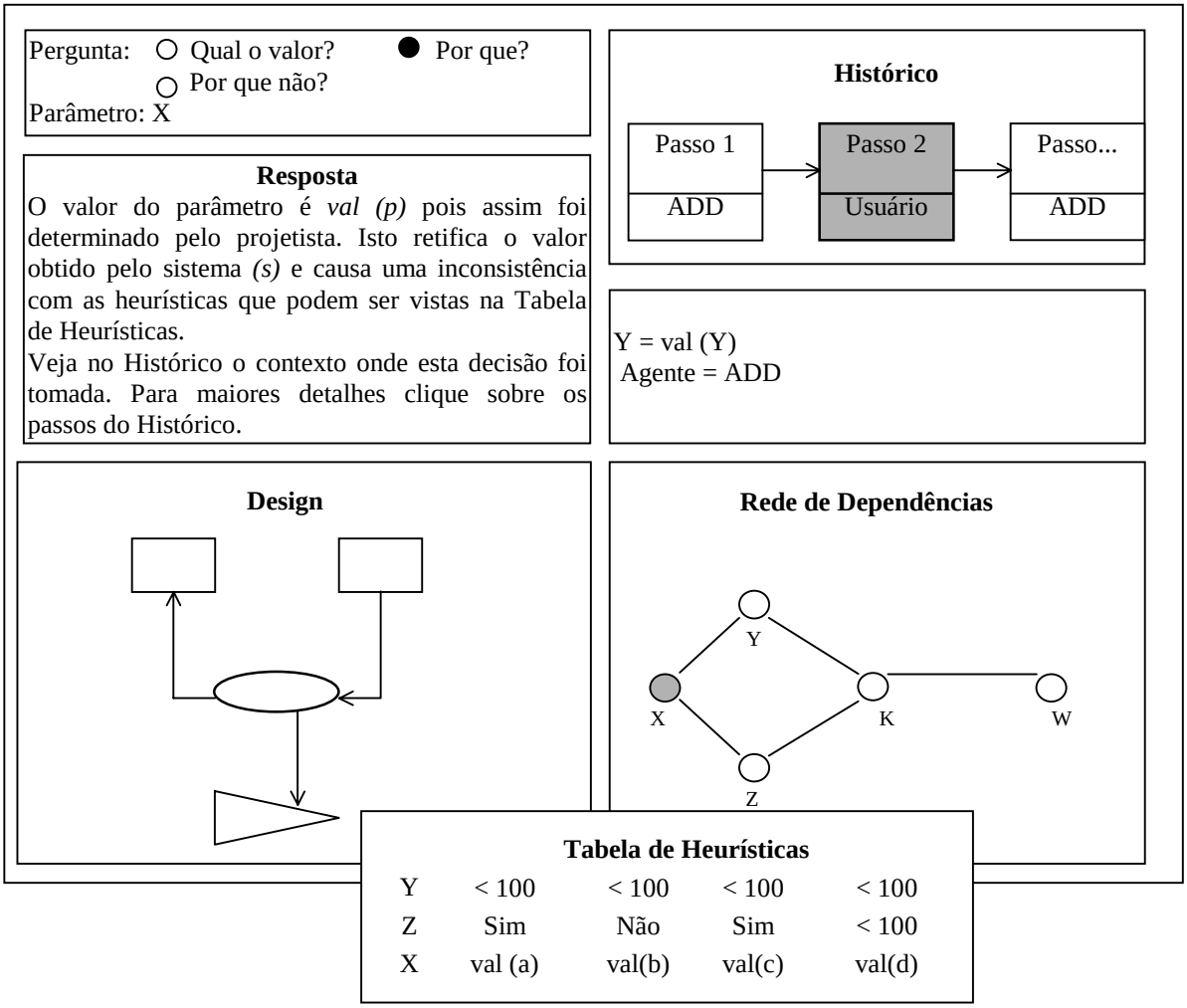


Figura 18: Esquema de Interface do módulo de Explicação do MultiADD

O sistema se encontra atualmente em fase de testes, a menos no que diz respeito às situações de conflito que serão avaliadas no futuro. As fontes de informação para análise de resultados se encontram sob duas formas. A primeira constitui um conjunto de comentários feitos livremente pelos usuários. A segunda corresponde à gravação dos passos de interação do usuário com o sistema, a partir dos quais podemos fazer inferências quanto a efetividade da resposta explicativa. Cabe lembrar que alguns tipos de passos interativos não puderam ser gravados nos *logs* devido a restrições da ferramenta de desenvolvimento de interface utilizada. Por exemplo, um *log* mostrava um usuário que aparentemente havia entrado e saído da explicação sem fazer nenhuma ação, no entanto pode ser que na verdade ele tenha percorrido

o Histórico através de uso da *scroll bar*. Como a ferramenta de interface não é sensível a este tipo de interação, ela não pôde ser captada. O uso de *scroll bar* permitiria também que fosse percebido se os usuários revisaram as Tabelas de Avaliação e de Heurísticas.

Ainda assim pudemos concluir alguns resultados qualitativos e de ordem subjetiva. Primeiramente pudemos observar que algumas respostas explicativas não foram esclarecedoras por dois motivos principais:

1. a escolha da forma expressiva não foi adequada, levando o usuário a um interpretante que não converge com o do desenvolvedor.
2. o modelo subjacente que servia de base para as respostas explicativas não estavam completos ou consistentes. Por exemplo: a informação que o modelo passava para a explicação não era consistente com a que era de fato aplicada, ou a informação entre os diferentes modelos não eram coerentes - um parâmetro possuía valor, mas não aparecia no histórico; os parâmetros utilizados na obtenção do valor de um outro não apareciam na rede de dependências.

Como consequência, percebemos através de comentários dos usuários as seguintes reações:

1. o usuário não compreendia a resposta e permanecia com a dúvida.
2. o usuário tentava adaptar seu modelo mental para sustentar o interpretante da nova resposta explicativa. No entanto este era “quebrado” logo em seguida com a apresentação de uma resposta correta. Este processo transmitia a sensação de instabilidade, insegurança e incerteza ao usuário.

Apesar destes pontos, verificamos que a maioria das respostas explicativas estavam corretas e foram compreendidas pelos usuários. Eles puderam entender como o domínio estava ali automatizado e como manipular as fontes de conhecimento apresentadas. Eles captaram o modelo de relevância entre os recursos e tiraram partido das informações apresentadas pelos mesmos, principalmente aquelas que complementavam as do texto em linguagem natural. Eles conseguiram também fazer uma avaliação sobre o funcionamento do sistema, identificando erros na automatização do domínio e chegando inclusive a propor objetivamente (nos termos do modelo do MultiADD) as modificações deveriam ser efetuadas para corrigir os erros

percebidos. Todos estes fatores confirmam potencialmente a realização dos papéis da Explicação do MultiADD: como esclarecedora do funcionamento do sistema e, consequentemente, como auxiliar no processo de testes e no processo de uma futura aquisição automática de conhecimento.

A testagem futura do uso do sistema de Explicação como recurso de negociação poderá revelar erros e acertos nesta perspectiva, integrando assim o conjunto de resultados futuros deste trabalho.

5. Discussão

Neste capítulo faremos uma avaliação do Planejador proposto. Os pontos de referência para esta análise são os requisitos especificados para o mesmo, cujas definições estão aqui repetidas [Moore'95] e seguidas das respectivas avaliações:

1. ***Naturalidade.*** Explicações devem ser estruturadas de acordo com padrões de discurso humanos. Ou seja, o sistema deve possuir conhecimento sobre estruturas de discurso e saber aplicar este conhecimento.

   Este requisito é atendido pelo Planejador, onde os esquemas retóricos representam as estruturas de discurso. O Planejador tem conhecimento sobre quando e como aplicar cada um dos esquemas retóricos. Estes por sua vez, quando linearizados, resultam em explicações estruturadas de acordo com padrões de discurso humanos.

2. ***Reatividade.*** O sistema de explicação deve ser capaz de aceitar *feedbacks* do usuário sobre suas explicações, responder perguntas subsequentes ou oferecer explicações alternativas, se o usuário não se mostrar satisfeito. Além disso o usuário deve poder fazer perguntas não muito bem formuladas.

   Embora este ponto não faça parte do conjunto de requisitos do Planejador, ele está citado entre os requisitos para um bom sistema de Explicação em [Moore'95] e sua função principal é tratar a possibilidade de diálogo. Como dito antes, a realização de diálogo não está prevista para o Planejador pois há um desequilíbrio entre o poder expressivo do gerador de perguntas atual e do gerador de respostas. No entanto parece interessante fazer um exercício sobre quais necessidades seriam impostas ao Planejador caso houvesse maior equilíbrio expressivo.

   Podemos citar a necessidade de representação de contexto de discurso, aumentando assim a sensibilidade do Planejador. Esta representação deveria conter um histórico de discurso que

armazenaria as perguntas feitas pelos usuários com seus respectivos esquemas retóricos de resposta, informações já disponíveis atualmente.

A representação dos esquemas retóricos no contexto de discurso permitiria a produção de explicações alternativas. Primeiramente caberia ao gerador de perguntas permitir ao usuário identificar que parte da explicação não foi compreendida. A partir desta identificação deveria ser possível mapeá-la para o objetivo comunicativo que originou o segmento de texto, como é feito em [Moore'95 e Moore&Mittal'96]. O Planejador por sua vez deveria ser capaz de compreender este novo tipo de pergunta. Para a geração de resposta bastaria possuir na biblioteca mais de uma entrada para atender a um mesmo objetivo comunicativo. Cada entrada teria uma marca indicativa de se ela já tinha sido utilizada ou não, evitando assim a repetição de um texto ou segmento de texto já linearizado.

Para atender a necessidade de perguntas subsequentes e a utilização dos recursos linguísticos decorrentes tais como: anáforas e perguntas elípticas, deveriam estar representados também no contexto de discurso os conceitos de tópico e foco. Como uma abordagem inicial teríamos como candidato a tópico do discurso o objeto de questionamento do usuário. Ainda assim deve ser investigado se outras informações poderiam constituir o tópico de discurso e qual a disponibilidade das mesmas. O conceito de foco precisaria ser calculado durante a geração do texto em linguagem natural e para isso seria necessária a construção dos métodos e heurísticas capazes de fazê-lo. No entanto, atualmente já seria possível se obter a informação de foco nas situações de manipulação direta dos objetos gráficos pelo usuário.

Como se vê ainda há muito o que fazer para atender a este requisito, constituindo-se assim um corpo representativo de pesquisa futura. Cabe lembrar que a solução de planejamento aqui proposta foi construída de forma a sustentar a futura integração deste requisito.

Os próximos dois requisitos são analisados em conjunto devido à interdependência entre eles.

3. ***Flexibilidade.*** Devem existir várias estratégias para se produzir uma explicação - responder a uma pergunta - para os casos em que o usuário não é capaz de entendê-la e para ela poder ser sensível a contexto (conhecimento ou objetivos do usuário).
4. ***Sensibilidade.*** Este item prevê o uso do conceito de contexto - conhecimento ou objetivos do usuário, momento da resolução de um problema ou diálogo precedente.

Estes requisitos são satisfeitos no que diz respeito à sensibilidade ao contexto, que para o Planejador corresponde a um ponto no modelo de tarefas do domínio. Ou seja, o Planejador é sensível a este contexto e possui diferentes estratégias de explicação para uma mesma pergunta quando esta é efetuada sob circunstâncias distintas. Cabe lembrar no entanto que esta sensibilidade ainda não foi totalmente implementada.

Outros tipos de contexto não estão previstos, como por exemplo:

- a sensibilidade a contexto no que tange a modelo de usuário não é possível a partir da abordagem semiótica adotada e pelos argumentos apresentados na seção 2.3 (Sistemas de Explicação); e
- o tratamento de explicações alternativas necessita de contexto de discurso e a indisponibilidade do mesmo foi discutida na apresentação do requisito Reatividade.

O modelo original ADD é sensível também a diferentes papéis de usuário [Garcia'92]. Embora o Planejador ainda não preveja este tipo de sensibilidade ela poderia ser incorporada. O custo de extensão seria ter diferentes visões sobre a base de conhecimento e um mapeamento de diferentes campos semânticos e terminológicos sobre a rede básica do ADD.

Como já explicado anteriormente, o Planejador não prevê modelo de usuário adaptativo, estando em linha com a Engenharia Semiótica. Caso se quisesse contar com esse modelo, além do abandono desta abordagem, seria requerido o desenvolvimento de um módulo inteligente de cálculo de evolução cognitiva.

5. ***Fidelidade.*** Uma explicação deve refletir o conhecimento e raciocínio do sistema.

   Este requisito é atendido pelo Planejador proposto. Ele expõe de forma amigável o conhecimento representado no SBC com vistas a aumentar a usabilidade deste. Algumas consequências deste fato são:

   - uma função tutorial da explicação para uma futura aquisição de conhecimento
   - uma função de validação, pois a exposição do conhecimento real do sistema implica também na exposição dos erros ali contidos.
   - um alto acoplamento entre a função de determinação de objetivos comunicativos e a representação de conhecimento adotada, diminuindo portanto a portabilidade do Planejador como um todo

6. ***Suficiência.*** O sistema deve ser capaz de responder ao conjunto de questões que o usuário deseja fazer. Conhecimento e estratégias de raciocínio do sistema devem estar disponíveis para a explicação.

   Atualmente um subconjunto das perguntas desejadas é atendido. Como plano futuro pretende-se responder a outros tipos de perguntas como por exemplo a pergunta: “E se a situação *X* ocorresse?”. Para viabilizar o planejamento de resposta desta pergunta a base de conhecimento deverá sustentar simulações, informando os impactos que seriam causados a partir da suposição do usuário. Tendo isto, uma abordagem inicial e com baixo custo de realização seria utilizar os mesmos esquemas RST da pergunta “Por que não?” só que com linearização no modo subjuntivo ao invés do indicativo. A Rede de Dependências apresentaria os parâmetros impactados, com seus valores antigos e modificados. Caberia ao usuário pesquisar na Rede de Dependências os parâmetros impactados que ele considera como relevantes.

   Outra forma de abordar simulações pode ser aquela apresentada em [de Souza&Garcia’94] onde o usuário informa de antemão aquilo que para ele é relevante, resultando em uma pergunta do tipo: “O que aconteceria com *Y* se a situação *X* ocorresse?”. Isto acarretaria a construção de novos esquemas RST, assim como um planejamento diferente para a Rede

de Dependências que deveria destacar aquilo que o usuário indicou como relevante. Também outra abordagem seria o próprio sistema possuir um modelo de relevância que seria apresentado ao usuário, como resposta à pergunta “E se a situação *X* ocorresse?”.

Além destas perguntas outras podem ser pensadas, como por exemplo perguntas que referenciassem outras entidades da base de conhecimento que não somente parâmetros, tais como: os mecanismos de inferência ou os métodos de obtenção de valores dos parâmetros. Cabe lembrar que novas perguntas sempre poderão ser imaginadas pelo usuário. O ideal é que a explicação permita a ele formar um modelo mental correto do funcionamento do sistema como um todo e das informações ali representadas. Com esta compreensão o usuário é capaz de responder a muitas de suas perguntas por si só sem a necessidade de aguardar a implementação da mesma no sistema de Explicação. Isto não justifica porém a não-implementação, visto que o esforço mental a ser feito pelo usuário pode ser diminuído através da apresentação automática da explicação, trazendo como consequência um aumento na produtividade do usuário.

7. ***Extensibilidade.*** A explicação deve ser capaz de responder a vários tipos de perguntas, algumas que podem não ter sido previstas na fase de design. O sistema deve poder ser facilmente estendido para responder a novas perguntas.

   O Planejador pode ser estendido para produzir mais respostas explicativas através da realização de duas atividades básicas: a construção de novos caminhos na árvore de determinação de objetivos comunicativos, e a carga do esquema retórico respectivo ao novo objetivo comunicativo na biblioteca de esquemas RST.

Finalizando a avaliação deste trabalho cabe destacar o tratamento dado ao problema de planejamento de textos como uma instância de problema de classificação. O planejador envolve tarefas que são a rigor tarefas de classificação vista a pré-existência de uma Biblioteca de Planos.

Caso o Planejador devesse calcular um plano RST inteiro com base apenas em raciocínio sobre crenças e intenções (que são o conteúdo de núcleos e satélites de relações RST), ele seria genuina e exclusivamente um raciocinador voltado para planejamento, devendo conter porém uma lógica de crenças e intenções que vai além do escopo deste trabalho. Além disto, como visto no estudo sobre Semiótica Computacional, calcular interpretações alheias, é, em si, uma atividade não parece adequada a tratamento computacional.

6. Conclusão

Durante a análise da solução proposta pudemos destacar duas contribuições deste trabalho. A primeira contribuição é de caráter técnico e corresponde a disponibilidade de uma biblioteca de esquemas RST que pode vir a ser re-utilizada por outros geradores de textos (com os devidos ajustes no que tange a representação de conhecimento).

A outra contribuição corresponde a adoção da perspectiva semiótica para a construção de respostas explicativas, até o momento pouco investigada.

Para trabalhos futuros podemos destacar duas abordagens principais. Em uma abordagem quantitativa ressaltamos os planos de se incorporar ao Planejador um maior número de perguntas e entidades sobre as quais perguntas podem ser realizadas, melhorando portanto o atendimento ao requisito Suficiência. Qualitativamente pretende-se adicionar ao Planejador a capacidade de diálogo, alinhada às questões semióticas e visando satisfazer ao requisito Reatividade.

Uma contribuição que nos parece importante diz respeito ao próprio modelo original ADD [Garcia'92]. Como reiteradamente colocado em trabalhos anteriores [de Souza&Garcia'94, Garcia&de Souza'95 e Garcia&de Souza'97], o modelo original apresenta somente um esboço do módulo explicativo, oferecendo ao usuário de uma aplicação uma baixa organização retórica do conhecimento nela contido. Nosso trabalho, em seu aspecto teórico e prático, representa um avanço da usabilidade deste modelo original, na medida em que o apresenta e transmite melhor para o usuário final de uma de suas aplicações.

Independente do ADD, porém, nossa escolha de RST abre interessantes perspectivas integradoras com outros modelos também baseados em RST. Vemos nisto o embrião de um modelo de explicação que trate diferentes tipos de representação e inclusive os explique homogeneamente para usuários afeiçoados a um ou outro tipo.

Apêndice A. Aspectos de Implementação

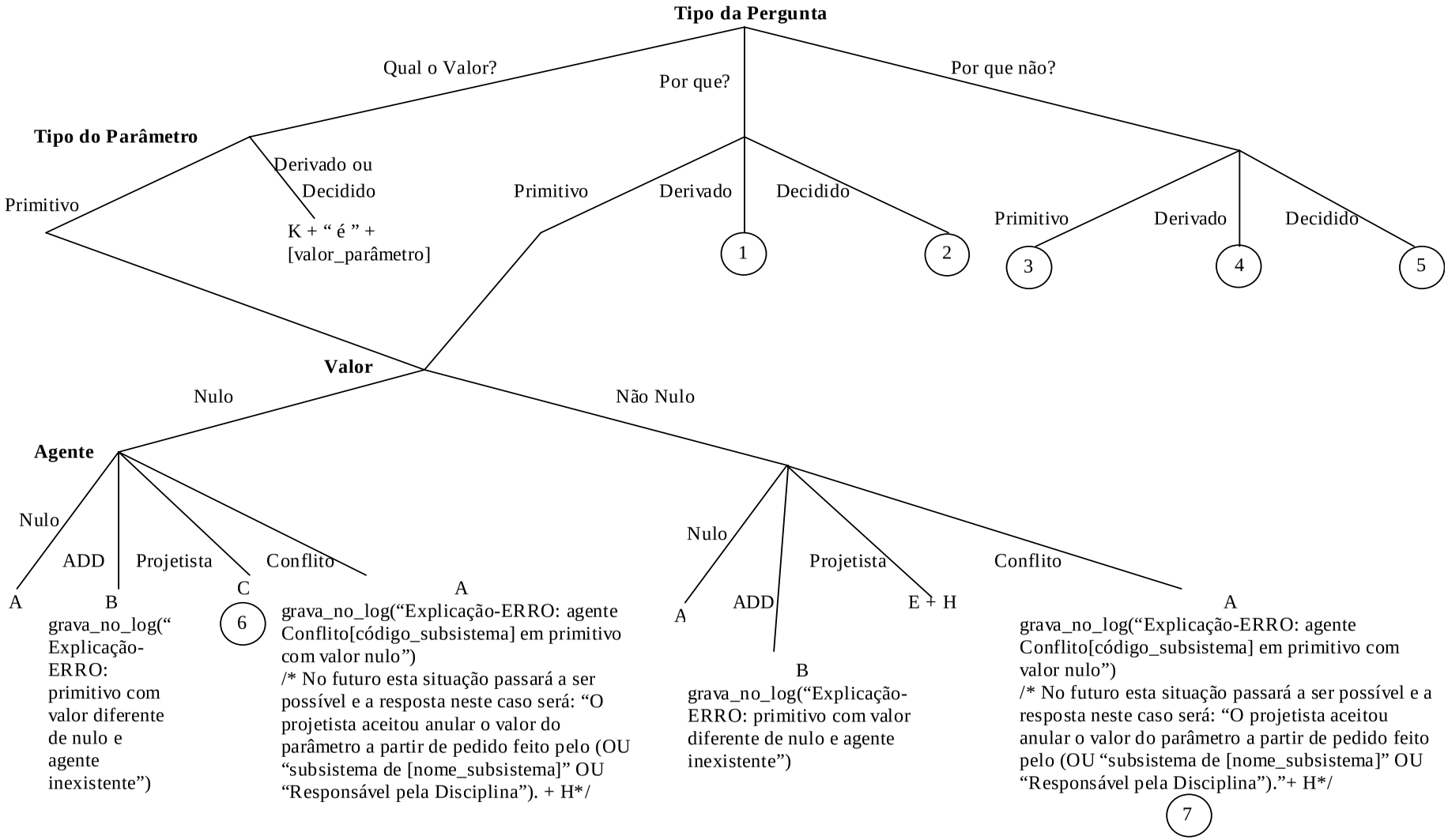
A seguir será apresentada uma representação gráfica das Bibliotecas de Planos implementadas nesta dissertação. Esta representação sintetiza tanto a aplicação da Biblioteca de Esquemas RST como as situações previstas pelo Planejador de Linguagem Gráfica.

A representação segue o formato de grafo (como uma árvore de decisão) e mais o seguinte padrão:

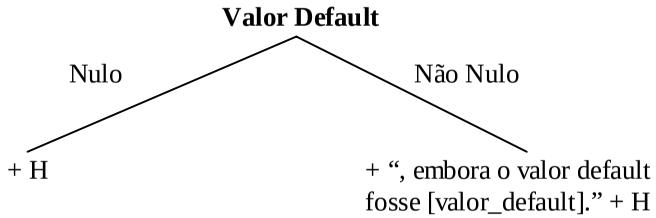
- comentários estão representados da seguinte forma: /* *conteúdo do comentátrio* */ ;
- as informações variáveis obtidas a partir da base de conhecimento para complementar os textos de resposta estão representadas da seguinte forma: [*informação*] ;
- o caracter "+" indica concatenação de *strings* e as letras maiúsculas podem representar variáveis que contém partes de texto (*strings*) fixas. A lista dos conteúdos das variáveis é apresentada no final deste Apêndice ;
- a apresentação de um número envolvido por um círculo indica a presença de uma sub-árvore que será expandida posteriormente ;
- as situações de erro previstas pelo Planejador de Linguagem Natural geram sempre a mesma resposta para o usuário. Funcionalmente, a apresentação desta resposta é acompanhada da gravação de um *log* para facilitar a futura depuração do erro por parte da equipe de desenvolvimento. Tratamentos de erro de interação, como por exemplo uma tentado do usuário de realizar uma pergunta sem informar o seu tipo (Qual o valor?, Por que? ou Por que não?) são tratados na camada de interface e por isto não estão previstos na Biblioteca de Planos. ;
- as situações previstas pelo Planejador de Linguagem Gráfica são apresentadas precedidas do marcador **#** , onde:
  - **#1** indica um comando para amostragem de uma janela não modal contendo a Tabela de Heurísticas que é utilizada para derivar o valor do parâmetro perguntado; e
  - **#2** indica um comando para amostragem de uma janela não modal que contém as avaliações das possíveis alternativas de valor de um parâmetro do tipo decidido.

Cabe lembrar que em todas as situações o Planejador de Linguagem Gráfica é ativado para atualizar a janela principal da Explicação que contém a Rede Dependências do parâmetro perguntado e o Histórico, cujo foco também deve ser o parâmetro perguntado (veja Figura 13). Também é importante notar que a linguagem gráfica utilizada não foi elaborada por este trabalho, mas é remanescente de versões anteriores de sistemas ADD [Garcia'94] com algumas melhorias.

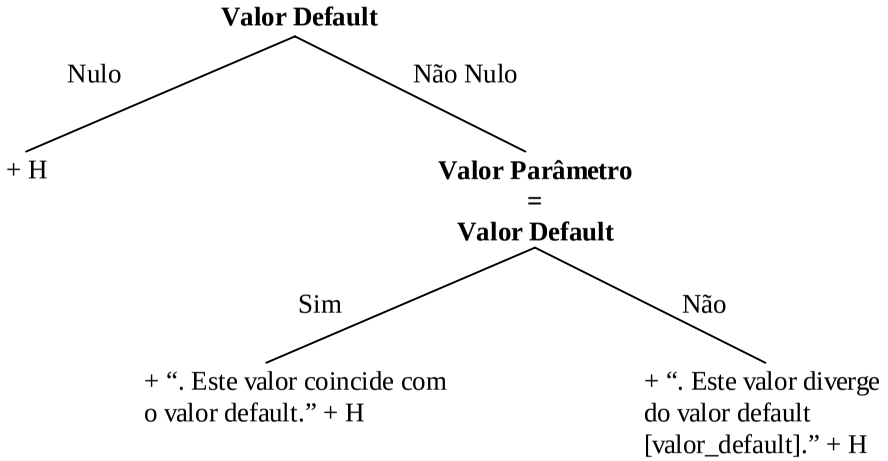
A Biblioteca de Planos atual está implementada na linguagem de programação PKC, que corresponde à uma extensão da linguagem C permitindo comunicação direta com a base de conhecimento, estando esta implementada no *shell* de Inteligência Artificial ProKappa Versão 3.0 (fabricante: Intellicorp, Mountain View, USA).


Tipo da Pergunta
Qual o Valor?
Por que?
Por que não?
Tipo do Parâmetro
Derivado ou Decidido
Primitivo
K + " é " + [valor_parâmetro]
Primitivo
Derivado
Decidido
1
2
Primitivo
Derivado
Decidido
3
4
5
Valor
Nulo
Não Nulo
Agente
Nulo
ADD
Projetista
Conflito
A
B
grava_no_log(" Explicação-ERRO: primitivo com valor diferente de nulo e agente inexistente")
C
6
A
grava_no_log("Explicação-ERRO: agente Conflito[código_subsistema] em primitivo com valor nulo")
/* No futuro esta situação passará a ser possível e a resposta neste caso será: "O projetista aceitou anular o valor do parâmetro a partir de pedido feito pelo (OU "subsistema de [nome_subsistema]" OU "Responsável pela Disciplina"). + H*/
Nulo
ADD
Projetista
E + H
Conflito
A
B
grava_no_log("Explicação-ERRO: primitivo com valor diferente de nulo e agente inexistente")
A
grava_no_log("Explicação-ERRO: agente Conflito[código_subsistema] em primitivo com valor nulo")
/* No futuro esta situação passará a ser possível e a resposta neste caso será: "O projetista aceitou anular o valor do parâmetro a partir de pedido feito pelo (OU "subsistema de [nome_subsistema]" OU "Responsável pela Disciplina")."+ H*/
7

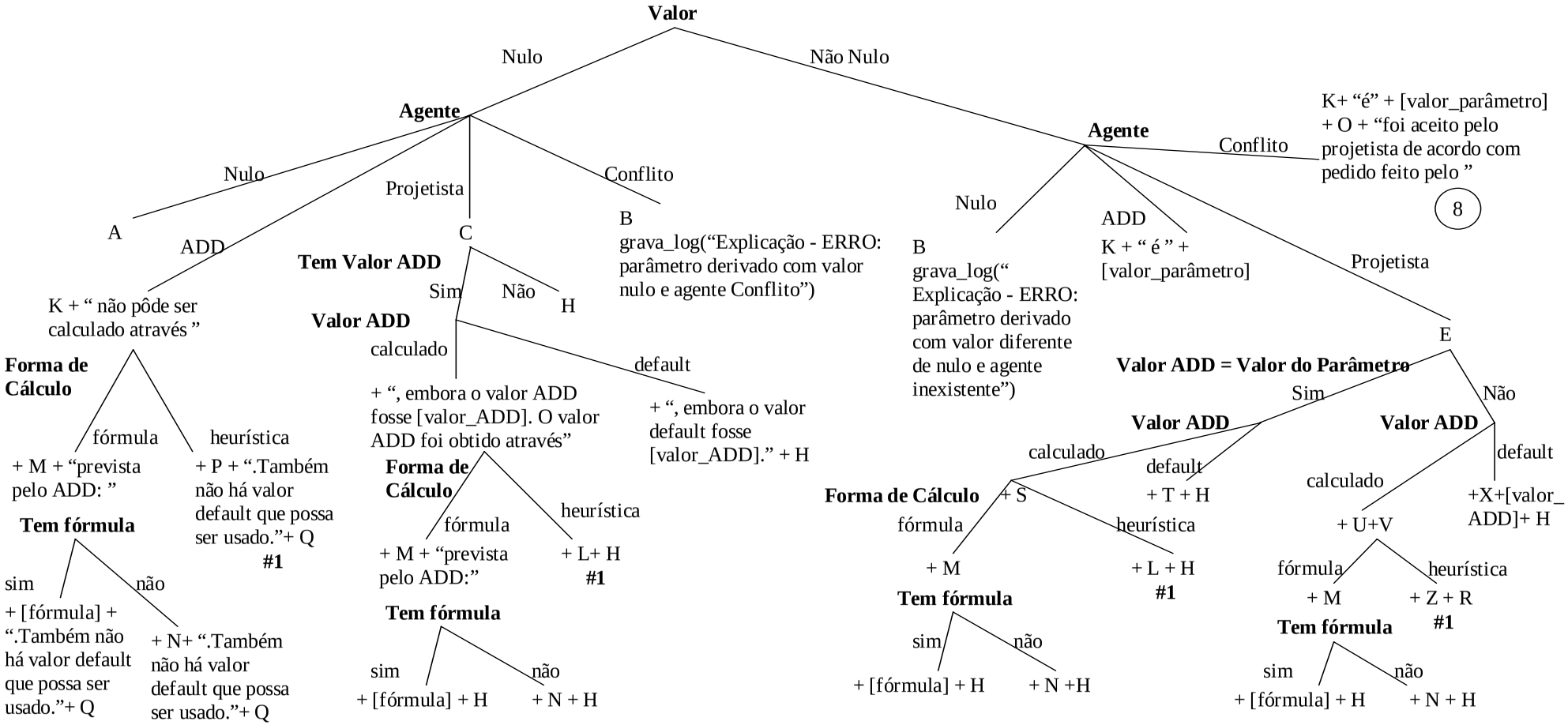
6.
Tipo da Pergunta= “Qual o Valor?” ou “Por que?”
Tipo do Parâmetro= Primitivo
Valor=Nulo
Agente=Projetista



7.
Tipo da Pergunta= “Qual o Valor?” ou “Por que?”
Tipo do Parâmetro= Primitivo
Valor=Nulo
Agente=Conflito

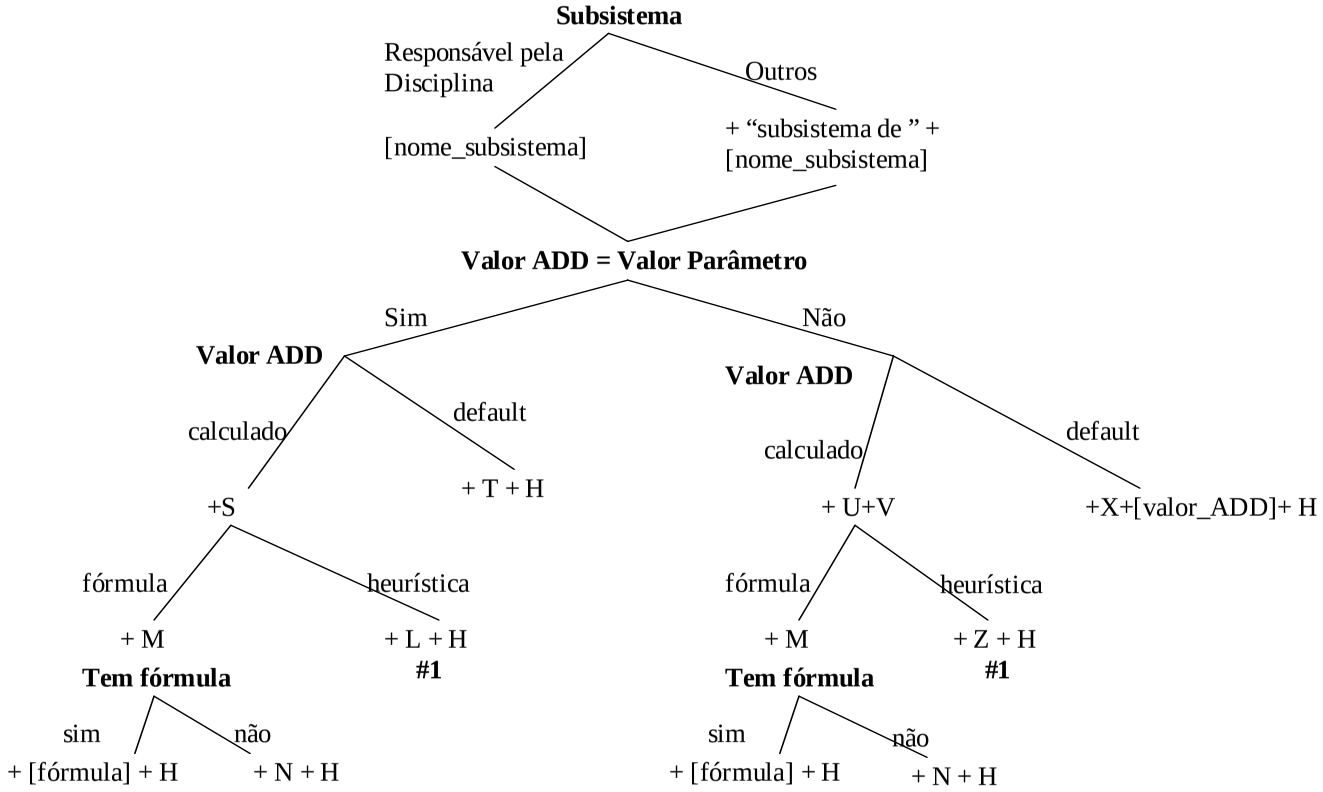


1.
Tipo da Pergunta= “Por que?”
Tipo do Parâmetro= Derivado



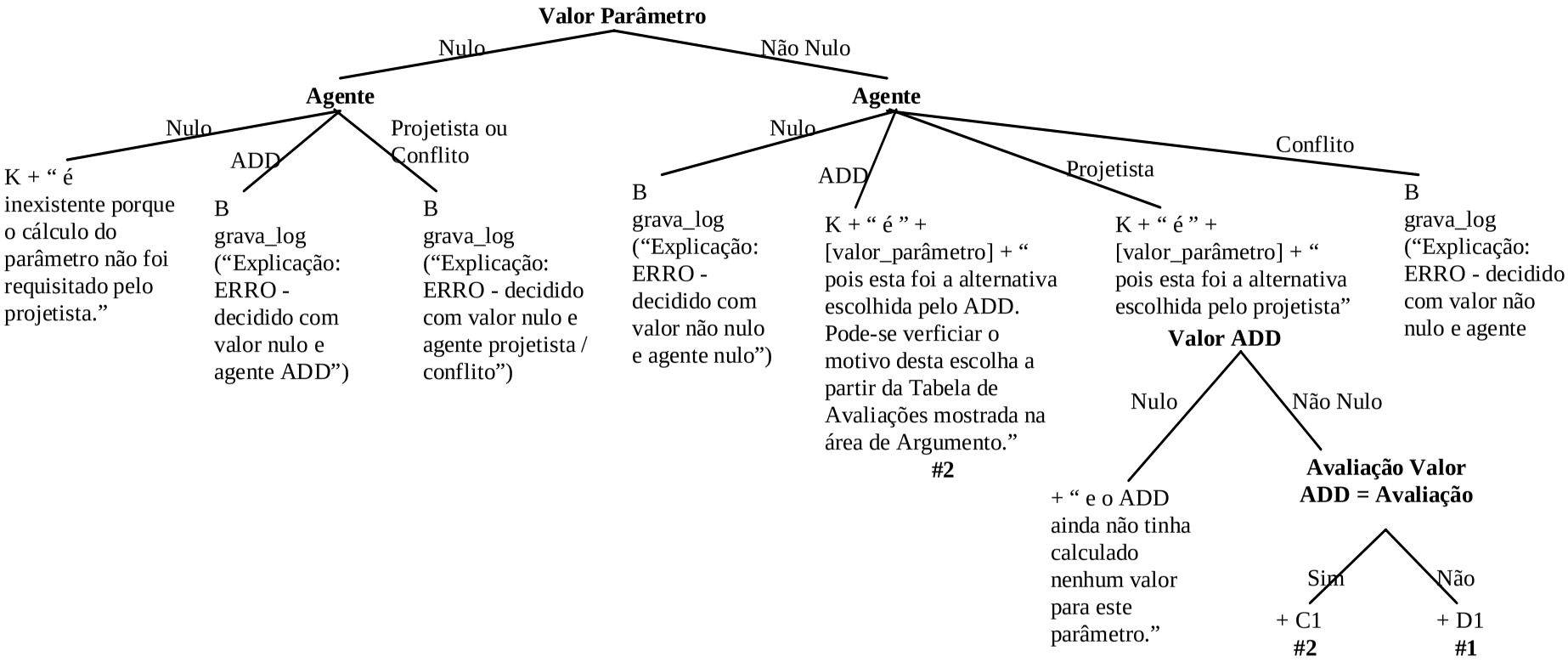
8.

Tipo da Pergunta= “Por que?”
Tipo do Parâmetro= Derivado
Valor = Não Nulo
Agente = Conflito



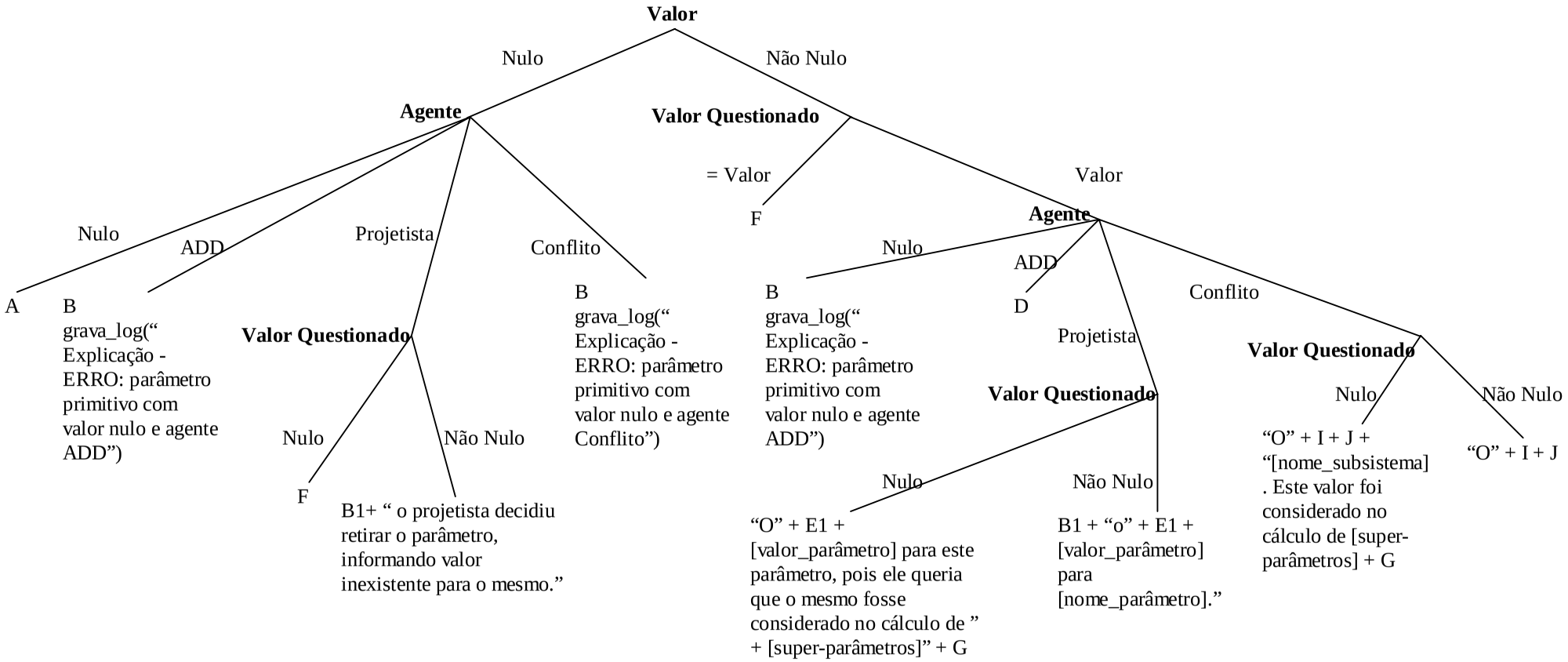
2.

Tipo da Pergunta= “Por que?”
Tipo do Parâmetro= Decidido

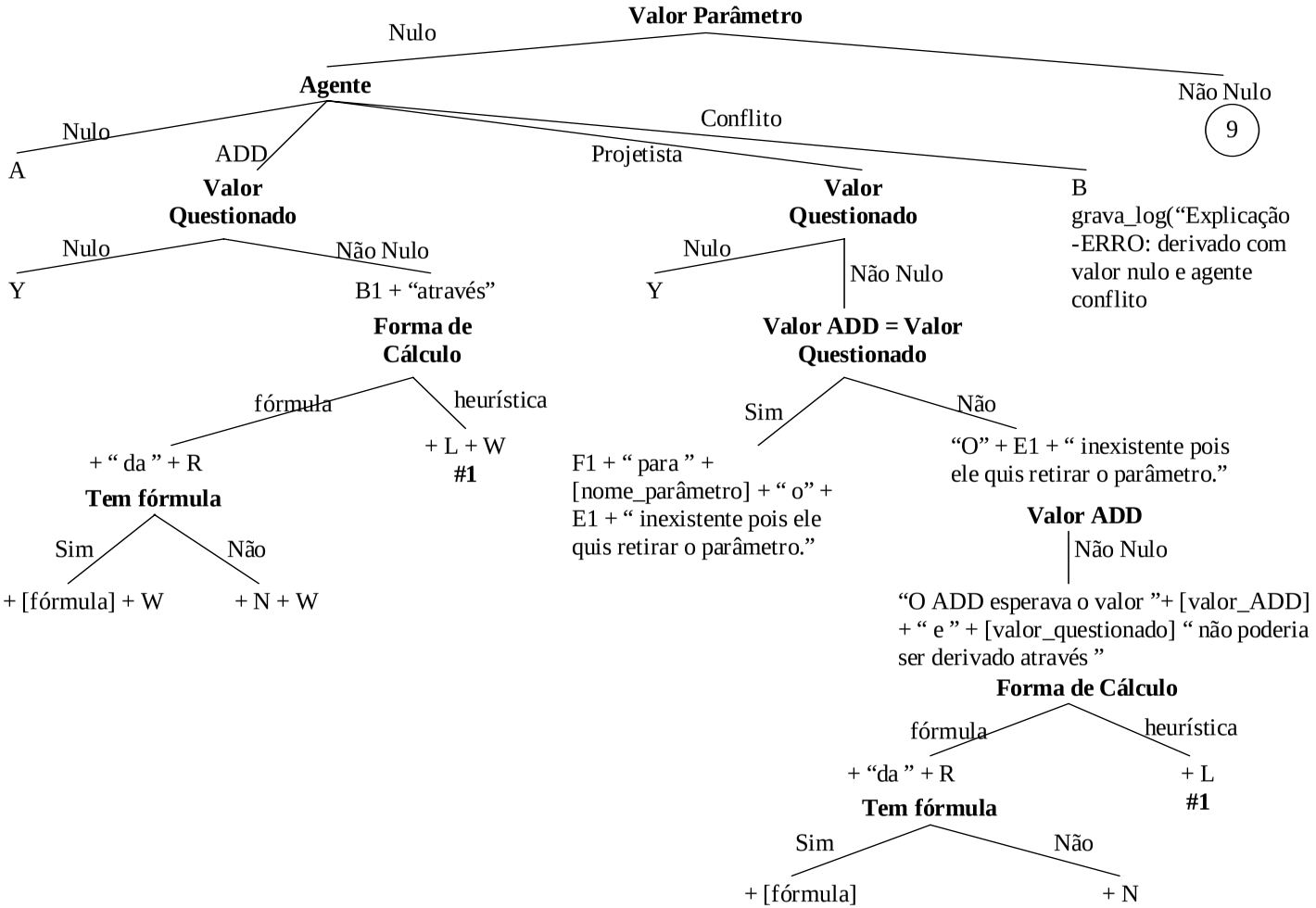


3.

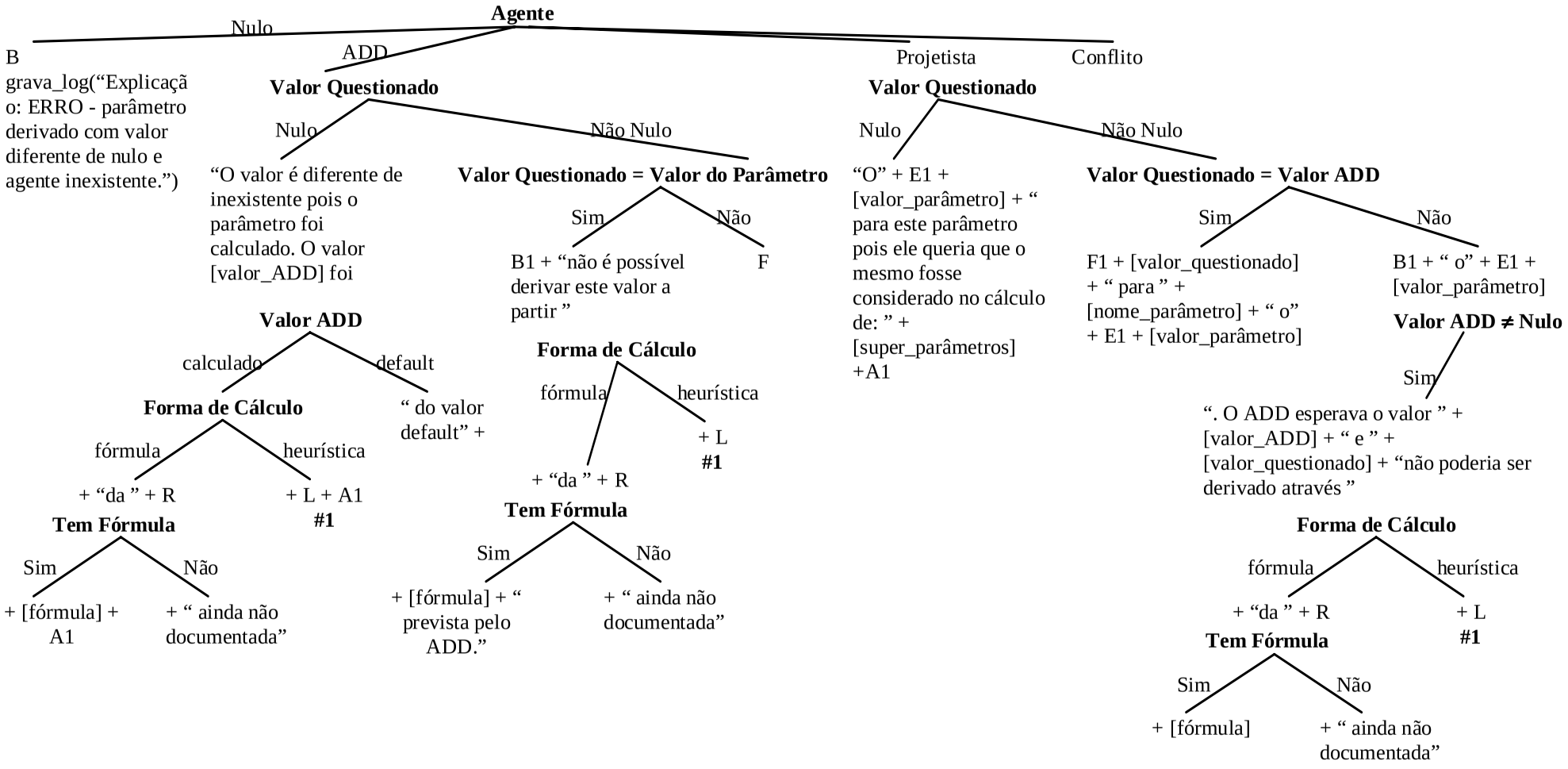
Tipo da Pergunta= “Por que não?”
Tipo do Parâmetro= Primitivo



4.
Tipo da Pergunta= “Por que não?”
Tipo do Parâmetro= Derivado

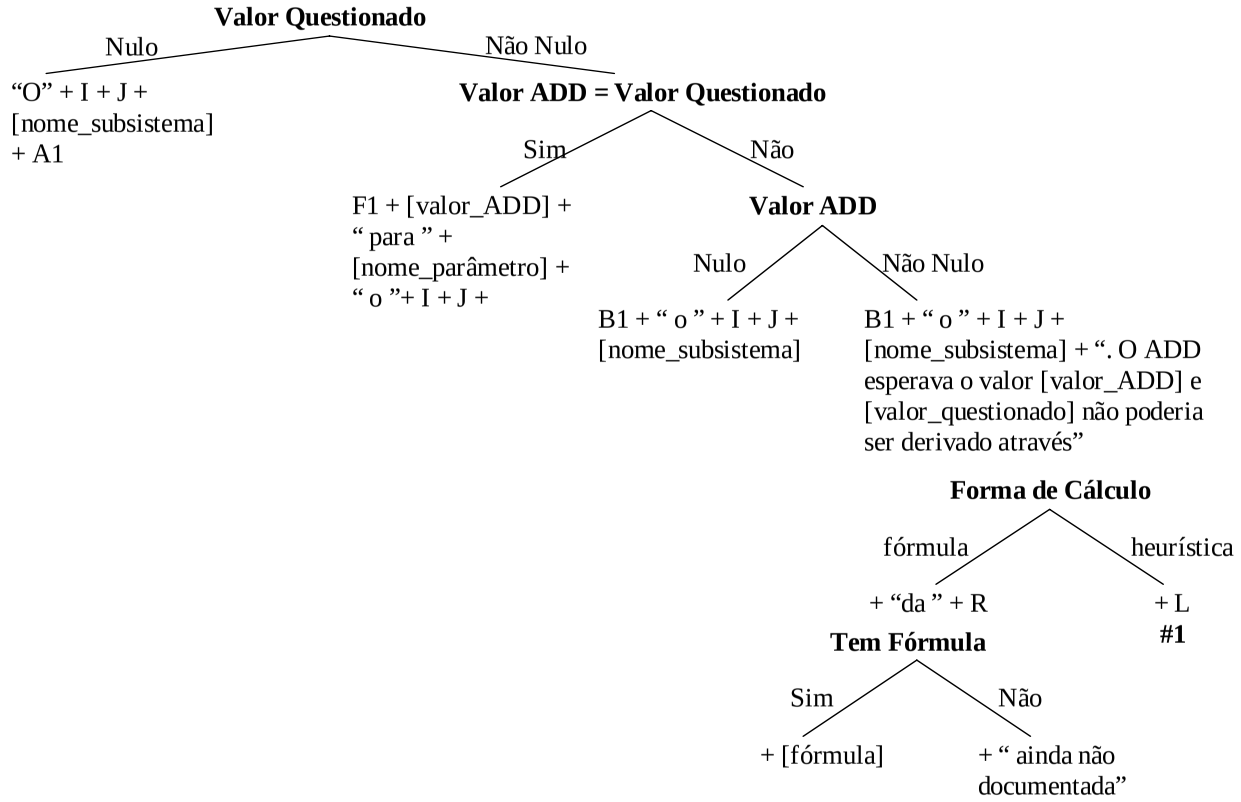


9.
Tipo da Pergunta= “Por que não?”
Tipo do Parâmetro= Derivado
Valor do Parâmetro= Não Nulo

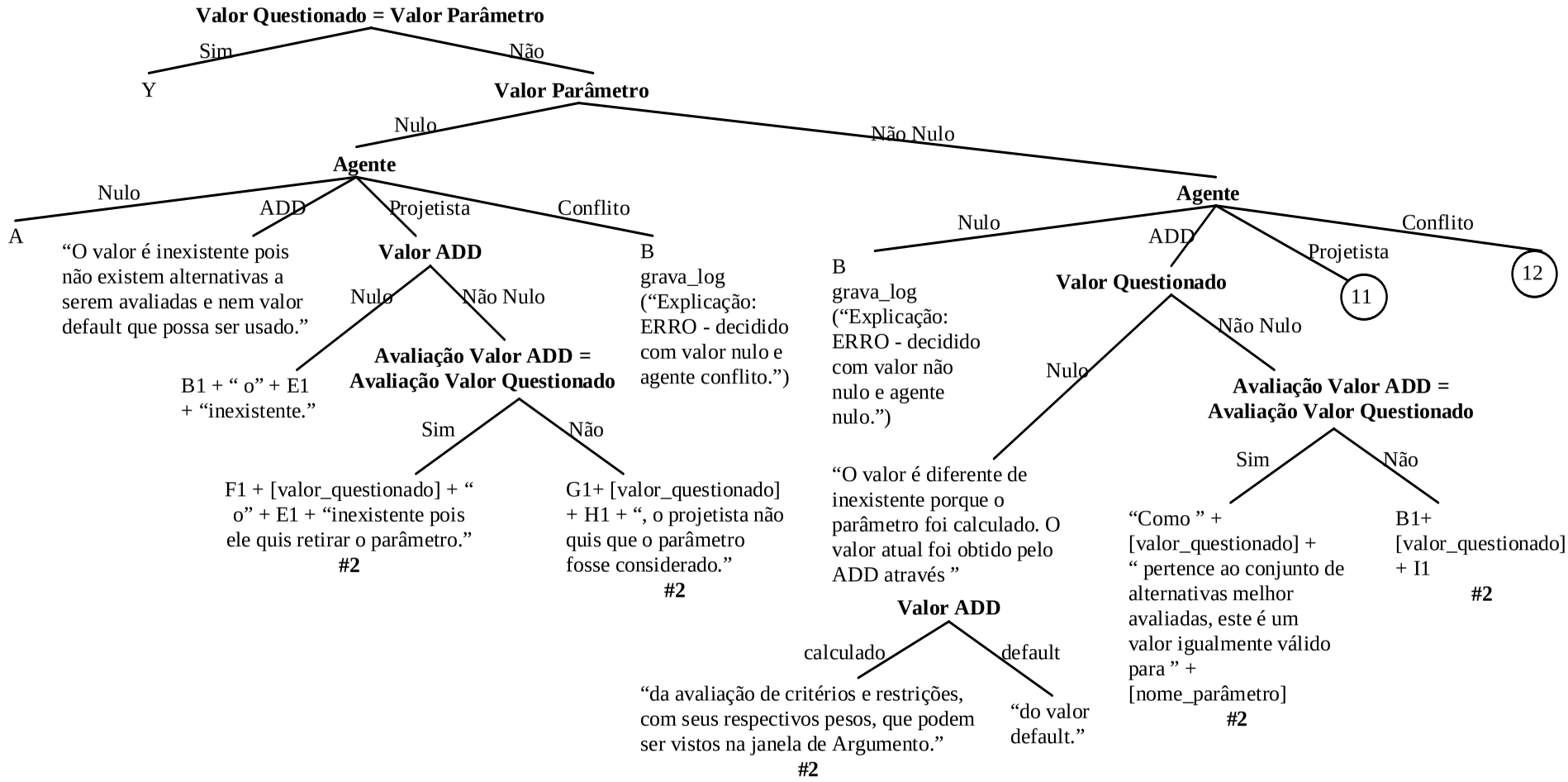


10.

Tipo da Pergunta= “Por que não?”
Tipo do Parâmetro= Derivado
Valor do Parâmetro= Não Nulo
Agente= Conflito

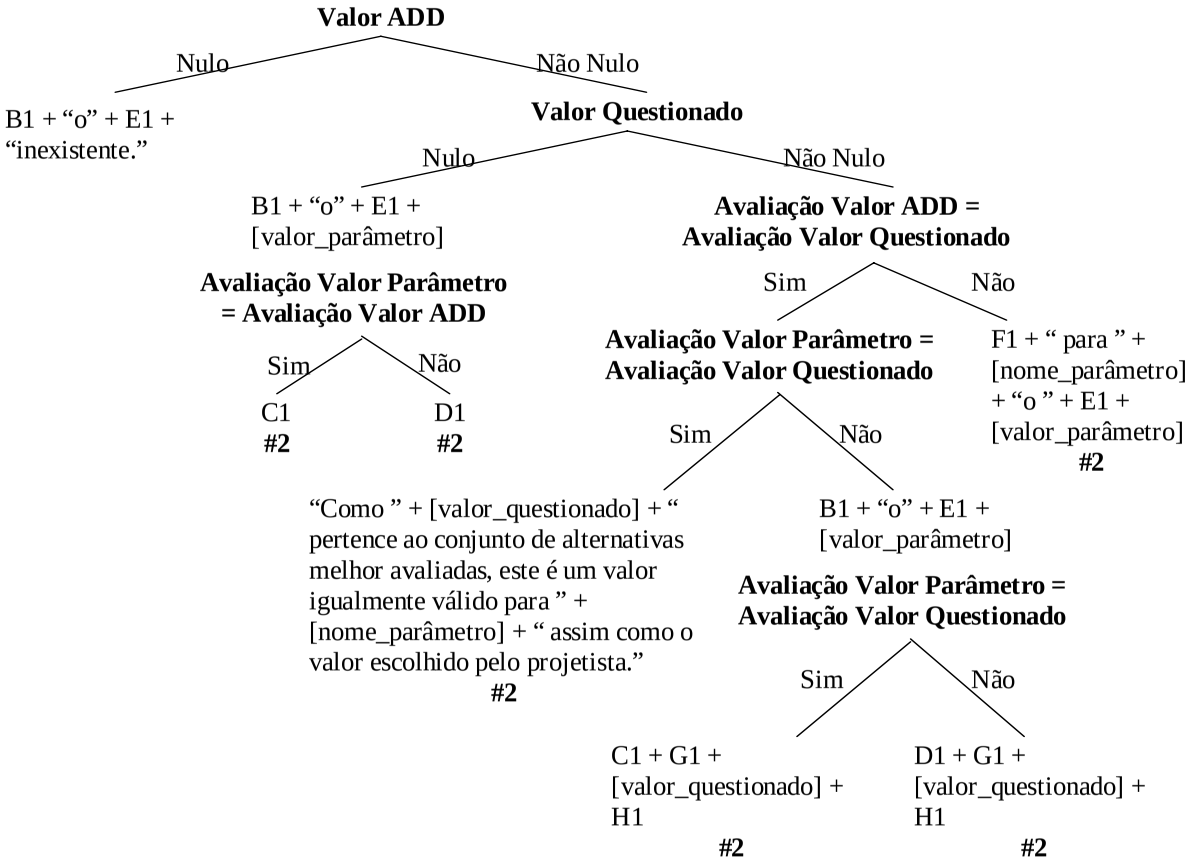


5.
Tipo da Pergunta= “Por que não?”
Tipo do Parâmetro= Decidido



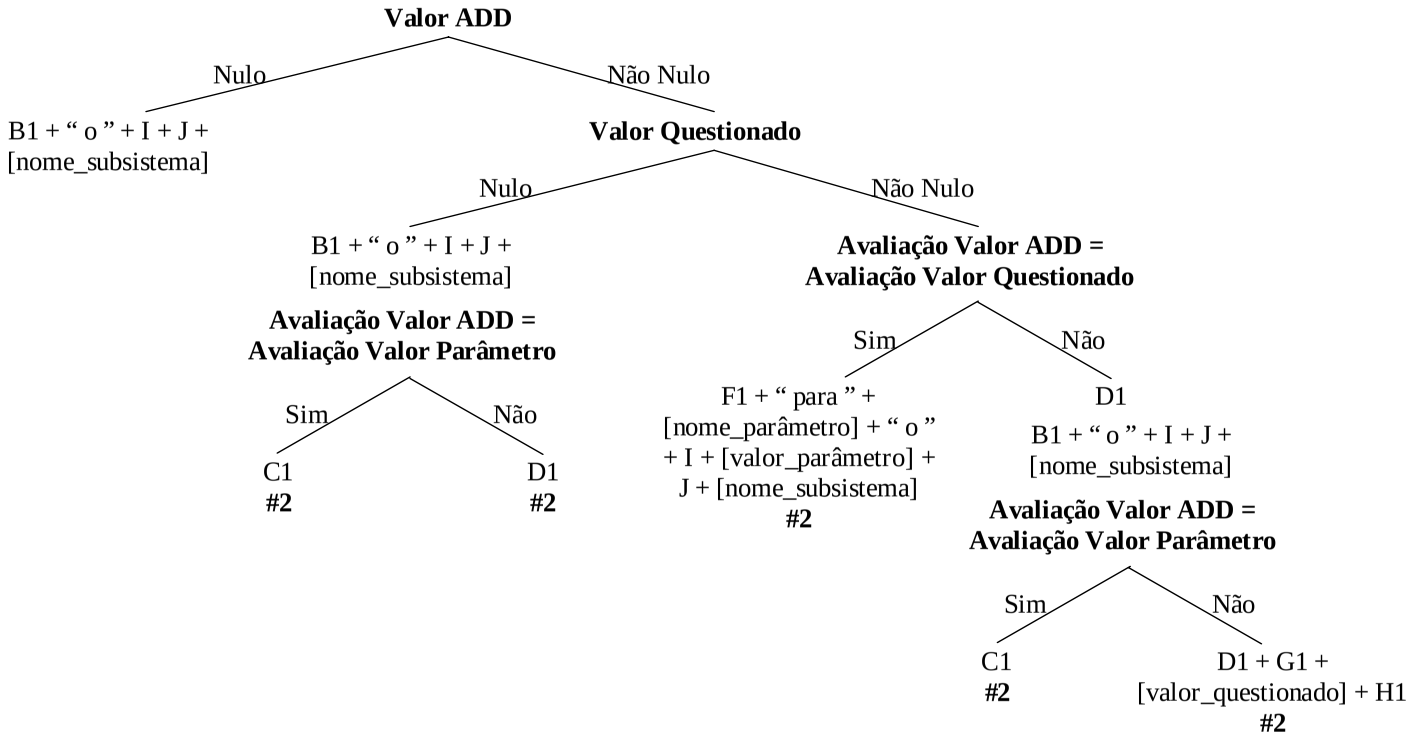
11.

Tipo da Pergunta= “Por que não?”
Tipo do Parâmetro= Decidido
Agente= Projetista



12.

Tipo da Pergunta= “Por que não?”
Tipo do Parâmetro= Decidido
Agente= Conflito



A=“A pergunta não pode ser respondida no estágio atual de projeto pois este parâmetro ainda não foi completamente analisado pelo projetista ou pelo ADD.”
B=“Infelizmente há um erro no modelo do sistema. Por favor contacte a equipe de desenvolvimento do ADDLabs.”
C=“O projetista anulou o valor do parâmetro”
D=“O valor do parâmetro [nome_do_parâmetro] é [valor_do_parâmetro], pois este é seu valor default, usado porque nenhuma informação foi colocada pelo projetista.”
E=“O valor do parâmetro [nome_do_ parâmetro] é [valor_do_ parâmetro] pois foi determinado pelo projetista.”
F=“Este já é o valor atual do parâmetro”
G=“Escolher o valor inexistente é o que se deve fazer quando não se quer que um parâmetro seja considerado.”
H=“Veja no Histórico o contexto onde esta decisão foi tomada.
Para maiores detalhes clique sobre os passos do Histórico.”
I=“ projetista aceitou o valor [valor_parâmetro] ”
J=“para este parâmetro a partir de um pedido de ”
K=“O valor de [nome_parâmetro] ”
L=“de uma das heurísticas que podem ser vistas na Tabela de Heurísticas”
M=“ da fórmula ”
N=“ ainda não documentada ”
O=“pois”
P=“de nenhuma das heurísticas previstas pelo ADD, que podem ser vistas na Tabela de Heurísticas”
Q=“Veja na Rede de Dependências os parâmetros que influenciaram este resultado. Clique sobre os mesmos para conhecer seus valores atuais.”
R=“ fórmula: ”
S=“Isto ratifica o valor calculado pelo sistema, que foi obtido através ”
T=“, coincidindo com o valor default.”
U=“Isto retifica o valor calculado pelo sistema - [valor_ADD]”
V=“e causa uma inconsistência com ”
W=“ não foi possível calcular um valor válido para este parâmetro e também não existe valor default que possa ser usado”
X= “, divergindo do valor default - ”
Y=“Este já é o valor atual do parâmetro”
Z= “as heurísticas que podem ser vistas na Tabela de Heurísticas”
A1=“Escolher valor inexistente é o que se deve fazer quando não se quer que um parâmetro seja considerado”
B1=“Porque ”
C1=“ ratificando o valor ADD”
D1=“ retificando o valor ADD - [valor_ADD]”
E1=“ projetista escolheu o valor ”
F1=“Embora o ADD também esperasse o valor [valor_questionado]”
G1=“Além de ”
H1=“ não poder ser obtido a partir dos critérios e restrições, com seus respctivos pesos, previstos pelo ADD ”

I1=“ não poderia ser obtido a partir dos critérios e restrições, com seus respctivos pesos, previstos pelo ADD”

Referências Bibliográficas

[ADDLabs] “Laboratório de Documentação Ativa e Design Inteligente”. Página internet: http://www.inf.puc-rio.br/~addlabs

[Adler&Winograd’92] Adler, P. S. e Winograd, T. (1992) “The Usability Challenge” em “Usability: Turning Technology into Tools”. New York. Oxford University Press.

[Appelt’82] Appelt, D. G. (1982) “Planning Natural-Language Utterances”. Proceedings of the National Conference on Artificial Intelligence - AAAI’82. pp. 59-62. Pittsburgh, Pensylvania.

[Appelt’83] Appelt, D. G. (1983) “Telegram: A Grammar Formalism for Language Planning”. Proceedings of the Eighth IJCAI Conference. pp. 595-599.

[Appelt’85] Appelt, D. G. (1985) “Planning English Sentences”. Cambridge. Cambridge University Press.

[Baddely’86] Baddely, A. D. (1986) “Working Memory”. Oxford: Oxford University Press.

[Bannon’86] Bannon, L. J. (1986) “Computer-Mediated Communication” em “User Centered System Design”. Editado por Norman, D. A. e Draper, S.. Hillsdale, New Jersey. Lawrence Erlbaum and Associates. Capítulo 21.

[Brennan’90] Brennan, S. E. (1990) “Conversation as Direct Manipulation: An Iconoclastic View” em “The Art of Human-Computer Interface Design”. Reading, MA. Addison-Wesley.

[Buchanan&Shortliffe’84] Buchanan, B. G. e Shortliffe, E. H. (1984) “Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project”. Reading, Ma. Addison-Wesley.

[Cawsey'92] Cawsey, A. (1992) "Explanation and Interaction - The Computer Generation of Explanatory Dialogues". Cambridge, Ma. MIT Press.

[Clayton&Norman'86] Clayton, L. e Norman, D. A. (1986) "Designing for Error" em "User Centered System Design". Editado por Norman, D. A. e Draper, S.. Hillsdale, New Jersey.Lawrence Erlbaum and Associates. Capítulo 20.

[Daneman&Carpenter'80] Daneman, M. e Carpenter, P. A. (1980) "Individual Differences in Working Memory and Reading." Journal of Verbal Learning and Verbal Behavior, 19, 450-466.

[de Souza'93] de Souza, C. S. (1993) "The Semiotic Engineering of User Interface Languages". International Journal of Man-Machine Studies. No. 39. pp. 753-773.

[de Souza'94] de Souza, C.S. (1994) "Testing Predictions of Semiotic Engineering in Human-Computer Interaction". Curitiba, Pr. Anais do VII Simpósio Brasileiro de Engenharia de Software. SBC-CITS-PUC/PR-UFRGS. pp.51-62.

[de Souza&Barbosa'96] de Souza, C. S. e Barbosa, S. D. J. (1996) "End-User Programming Environments: The Semiotic Challenges". PUC-RioInf MCC19/96. Rio de Janeiro, RJ.

[de Souza&Garcia'94] de Souza, C. S. e Garcia, A. C. B. G. (1994) "Towards a Rethoric of Active Design Documents". Fortaleza, Ce. Procceedings SBIA'94. pp. 523-534.

[de Souza&Leite'97] de Souza, C. S. e Leite, J. C. (1997) "Projeto de Interfaces de Usuários". PUC-RioInf MCC11/97. Rio de Janeiro, RJ.

[De Smedt'90] De Smedt, K. J. M. J. (1990) "IPF: An Incremental Parallel Formulator" em "Current Research in Natural Language Generation". Editado por Dale, R. e Mellish, C. e Zock, M.. San Diego, CA.. Academic Press. Capítulo 7.

[Dias'94] Dias, M. C. P. (1994) "O Léxico em Sistemas de Análise e Geração Automática de Textos em Língua Portuguesa". Rio de Janeiro, RJ. Tese de Doutorado. Departamento de Letras, PUC-Rio.

[Draper'86] Draper, S. (1986) "Display Managers as the Basis for User-Machine Communication". em "User Centered System Design". Editado por Norman, D. A. e Draper, S.. Hillsdale, New Jersey. Lawrence Erlbaum and Associates. Capítulo 16.

[Eco'76] Eco, Umberto (1976) "Tratado Geral de Semiótica". São Paulo, SP. Editora Perspectiva.

[Fisher'87] Fisher, G. (1987) "A Critic of LISP". Proceedings of the Tenth International Joint Conference on Artificial Intelligence. Milão, Itália. Morgan Kaufman, Los Altos, California. pp. 177-184.

[Garcia'92] Garcia, A. C. B. (1992) "Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design". Stanford, Ca. Dissertação PhD.

[Garcia'94] Garcia, A. C. B. (1994) "ADDVac: Sistema de Apoio a Projetos de Ventilação, Ar Condicionado e Refrigeração de Plataformas Offshore". Rio de Janeiro, RJ.. Petrobrás/CENPES. ME-3029.

[Garcia'95] Garcia, A. C. B. (1995) "ADDProc: Sistema de Apoio a Projetos de Planta de Processo de Plataformas Offshore". Rio de Janeiro, RJ.. Petrobrás/CENPES. RL.3521.01, RL.3521.02, RL.3521.03 e RL.3755.02.

[García'95] García, L. S. (1995) "LINX: Um Ambiente Integrado de Interface para Sistemas de Informação Baseados em Conhecimento". Rio de Janeiro, RJ. Tese de Doutorado. Departamento de Informática, PUC-Rio.

[Garcia&de Souza'95] Garcia, A. C. B. e de Souza, C. S. (1995) "Rhetorical Structures in the Delivery of Design Intent". Proceedings of II Congress of Computing in Civil Engineering. Atlanta, GA.. pp. 1443-1452.

[Garcia&de Souza'97] Garcia, A. C. B. e de Souza, C. S. (1997) "ADD$^+$: Including Rhetorical Structures in the Active Documents". A ser publicado no AIEDAM: Special Issue in Design Rationale.

[Gorry&Silverman&Pauker'78] Gorry, G. A. e Silverman, H. e Pauker, S. G. (1978) "Capturing Clinical Expertise: A Computer Program that Considers Clinical Responses to Digitalis". American Journal of Medicine. 64:452-460.

[Hutchins&Hollan&Norman'86] Hutchins, E. L. e Hollan, J. D. e Norman, D. A. (1986) "Direct Manipulation Interfaces" em "User Centered System Design". Editado por Norman, D. A. e Draper, S.. Hillsdale, New Jersey. Lawrence Erlbaum and Associates. Capítulo 5.

[Ioerger'94] Ioerger, T. R. (1994). "The Manipulation of Images to Handle Indeterminacy in Spatial Reasoning". Cognitive Science, 18. pp. 513-549. 1994.

[Kammersgaard'88] Kammersgaard, J. (1988) "Four Different Perspectives on Human-Computer Interaction". International Journal of Man-machine Studies. No. 28. pp. 343-362. 1988.

[Kobsa&Wahlster'89] Kobsa, A. e Wahlster, W. (1989) (Eds.) "User Models in Dialog Systems". Springer-Verlag Berlin Heidelberg.

[Lang&Lang &Auld'81] Lang, T., Lang, K. N. e Auld, R. (1981) "Support for Users of Operating Systems and Applications Software". International Journal of Man-Machine Studies, 14, pp. 269-282.

[Lang&Auld&Lang'82] Lang, K. N., Auld, R. e Lang, T. (1982) "The Goals and Methods of Computer Users". International Journal of Man-Machine Studies, 17, pp. 375-399

[Laurel'86] Laurel, B. K. (1986) "Interface as Mimesis" em "User Centered System Design". Editado por Norman, D. A. e Draper, S.. Hillsdale, New Jersey. Lawrence Erlbaum and Associates. Capítulo 4.

[Mann&Thompson'87] Mann, W. C. e Thompson, S. A. (1987) "Rhetorical Structure Theory: A Theory of Text Organization". Information Science Institute. University of Southern California.

[Maybury'93] Maybury, M. T. (1993) "Intelligent Multimedia Interfaces". Menlo Park, Ca. AAAI Press, MIT Press.

[McKeown'85] MacKeown, K. R. (1985) "Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Texts". Cambridge, Ma. Cambridge University Press.

[Moore&Mittal'96] Moore, J. D. e Mittal, V. O. (Julho, 1996) "Dynamically Generated Follow-Up Questions". IEEE Computer: Special Issue in Natural Language Processing. pp. 75-86.

[Moore'95] Moore, J. D. (1995) "Participating in Explanatory Dialogues: Interpreting and Responding to Questions in Context.". Cambridge, Ma. MIT Press.

[Moore&Swartout'91] Moore, J. D. (1991) "A Reactive Approach to Explanation" em "Natural Language Generation in Artificial Intelligence and Computational Linguistics". Editado por Paris, C. L. e Swartout, W. R. e Mann, W. C.. Norwell, Ma.. Klewer Academic Publishers Group.

[Nadin'88] Nadin, M. (1988) "Interface Design and Evaluation - Semiotic Implications" em "Advances in Human-Computer Interaction", Vol. 2. Editado por Hartson, H. R. e Hix, D.. Norwood, NJ.. Ablex Publishing Corporation. Capítulo 2.

[Norman'86] Norman, D. A. (1986) "Cognitive Engineering" em "User Centered System Design". Editado por Norman, D. A. e Draper, S.. Hillsdale, New Jersey.Lawrence Erlbaum and Associates. Capítulo 3.

[Norman&Draper'86] Norman, D. A. e Draper, S. (1986) (Eds.) "User Centered System Design". Hillsdale, New Jersey. Lawrence Erlbaum and Associates.

[Nunes'91] Nunes. M. G. V. (1991) "A Geração de Respostas Cooperativas em Sistemas Baseados em Lógica". Rio de Janeiro, RJ. Tese de Doutorado. Departamento de Informática, PUC-Rio.

[O'Malley'86] O'Malley, C. E. (1986) "Helping Users Help Themselves" em "User Centered System Design". Editado por Norman, D. A. e Draper, S.. Hillsdale, New Jersey. Lawrence Erlbaum and Associates. Capítulo 18.

[Oliveira'92] Oliveira, D. A. S. (1992) "Um Provador de Teoremas em Dedução Natural Capaz de Complementar seu Conhecimento". Rio de Janeiro, RJ. Dissertação de Mestrado. Departamento de Informática. PUC-Rio.

[Oliveira'95] Oliveira, D. A. S. (1995) "Transformação de Provas em DN com vistas à Geração de Explicações Interativas". Rio de Janeiro, RJ. Monografias da Ciência da Computação. Departamento de Informática. PUC-Rio.

[Oliveira&de Souza&Haeusler'96] Oliveira, D. A. S., de Souza, C. S. e Haeusler, E. H. (1996) "Structured Argument Generation in a Logic Based KB Systems". Proceedings of ITALLC'96 (Second Conference on Information-Theoretic Approaches to Logic, Language and Computation. Regent's College, London. pp. 173-181.

[Owen'86] Owen, D. (1986) "Answers First, Then Questions". em Norman e Draper (Eds.) "User Centered System Design". Hillsdale, New Jersey. Lawrence Erlbaum and Associates. Capítulo 17.

[Paris'91] Paris, C. L. (1991) "Generation and Explanation" em "Natural Language Generation in Artificial Intelligence and Computational Linguistics". Editado por Paris, C. L. e Swartout, W. R. e Mann, W. C.. Norwell, Ma.. Klewer Academic Publishers Group.

[Peirce'31] Peirce, C. S. (1931) "Collected Papers.". Cambridge, Ma. Harvard University Press.

[Pollack&Hirschberg&Webber'82] Pollack, M. e Hirschberg, J. e Webber, B. (1982) "User Participation in the Reasoning Processes of Expert Systems". Proceedings do AAAI. Pittsburg, PA.. American Association of Artificial Intelligence.

[Quental'95] Quental, V. S. T. D. B. (1995) "Uma Gramática para Compreensão e Geração Automática de Textos de Língua Portuguesa". Rio de Janeiro, RJ. Tese de Doutorado. Departamento de Letras, PUC-Rio.

[Scott&Clancey&Davis&Shortliffe'84] Scott, A. C. e Clancey, W. J. e Davis, R. e Shortliffe, E. H. (1984) "Methods for Generating Explanations" em "Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project". Editado por Buchanan, B. G. e Shortliffe, E. H.. Addison-Wesley. Reading, Ma.. pp. 338-362.

[Scott&de Souza'90a] Scott, D. R. e de Souza, C. S. (1990) "Getting the Message Across in RST-based Text Generation" em "Current Research in Natural Language Generation". Editado por Dale, R. e Mellish, C. e Zock, M.. San Diego, CA.. Academic Press. Capítulo 3.

[Scott&de Souza'90b] Scott, D. R. e de Souza, C. S. (1990) "Planejamento Conciliatório para Textos Descritivos Extensos". Revista Brasileira de Computação. Vol. 5, No. 3, Janeiro/Março 1990. Rio de Janeiro, RJ.. Sociedade Brasileira de Computação (SBC) e Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro (NCE/UFRJ).

[Scharer'83] Scharer, L. L. (Julho, 1983) "User Training: Less is More". Datamation, pp. 175-182.

[Sparck Jones'84] Sparck Jones, K. (1984) "User Models and Expert Systems". Technical Report 61, Computer Laboratory. Cambridge University.

[Sparck Jones'89] Sparck Jones, K. (1989) "Realism about User Modeling" em "User Models in Dialog Systems". Editado por Kobsa, A. e Wolfgang, W.. Springer-Verlag Berlin Heidelberg. Capítulo 12.

[Stefik'95] Stefik, M. (1995) "Introduction to Knowledge Systems". San Francisco, CA. Morgan Kaufmann Publishers.

[Swartout'77] Swartout, W. R. (1977) "A Digitalis Therapy Advisor with Explanations". Technical Report TR-176. Labortory of Computer Science. Massachusetts Institute of Technology.

[Swartout'90] Swartout, W. R. (1990) "Evaluation Criteria for Expert System Explanation". Apresentado no *Workshop on Evaluation of Natural Language Generation Systems* do AAAI90. Boston

[Vieira da Cunha&de Souza'97] da Cunha, C. K. V. e de Souza, C. S. (Agosto, 1997) "The Role of Explanation Systems in MultiAgent Applications". International Conference on Human-Computer Interaction. San Franciso, CA. A ser publicado na seção *Interactive Poster Sessions*.

[Vivacqua&Garcia'96a] Vivacqua, A. S. & Garcia, A. C. B. (1996) "MultiADD: Uma Extensão do Modelo ADD para Viabilizar Design de Grupo Interdisciplinar". Proposta de Tese de Mestrado. Rio de Janeiro, RJ.

[Vivacqua&Garcia'96b] Vivacqua, A. S. e Garcia, A. C. B. (1996) "The Use of Active Design Documents to Assist Conflict Mitigation in Concurrent Engineering". Toronto, Canada. Third Conference on Concurrent Engineering and Research Applications.

[Wahlster&Kobsa'89] Wahlster, W. e Kobsa, A. (1989) "User Models in Dialog Systems" em "User Models in Dialog Systems". Editado por Kobsa, A. e Wahlster, W.. Springer-Verlag Berlin Heidelberg. Capítulo 1.