4– Ferramenta de apoio ao método

Uma ferramenta foi desenvolvida para fornecer apoio ao método no gerenciamento dos conflitos. Esta ferramenta analisa as respostas de cada participante para identificar os conflitos e apontar as melhores técnicas para o seu tratamento de acordo com suas causas.

A ferramenta ajuda o líder e os planejadores da reunião em diversas tarefas. A principal delas é automatizar a identificação dos conflitos, fornecendo um relatório com a sua descrição e apontando as técnicas de gerenciamento mais indicadas. A ferramenta ainda permite aos participantes responderem ao questionário on-line de qualquer lugar do mundo através da Internet. Por fim, a ferramenta automatiza o cadastramento dos projetos, usuários da ferramenta, reuniões, participantes de cada reunião, objetivos e requisitos das reuniões.

A ferramenta de apoio está disponível no seguinte endereço:

http://reuniao.les.inf.puc-rio.br

4.1. Novas características da ferramenta

A ferramenta foi desenvolvida em PHP, HTML e Javascript com a base de dados gerenciada através do MySQL e utiliza o servidor Web Apache, ganhando uma nova possibilidade de interatividade e caracterizando desta maneira uma aplicação em 3 camadas.

Além disso, a tarefa de identificação dos conflitos agora é realizada por um sistema especialista integrado à ferramenta de apoio, desenvolvido através da tecnologia do software CLIPS. Falaremos mais detalhadamente sobre a criação do sistema especialista ainda neste capítulo.

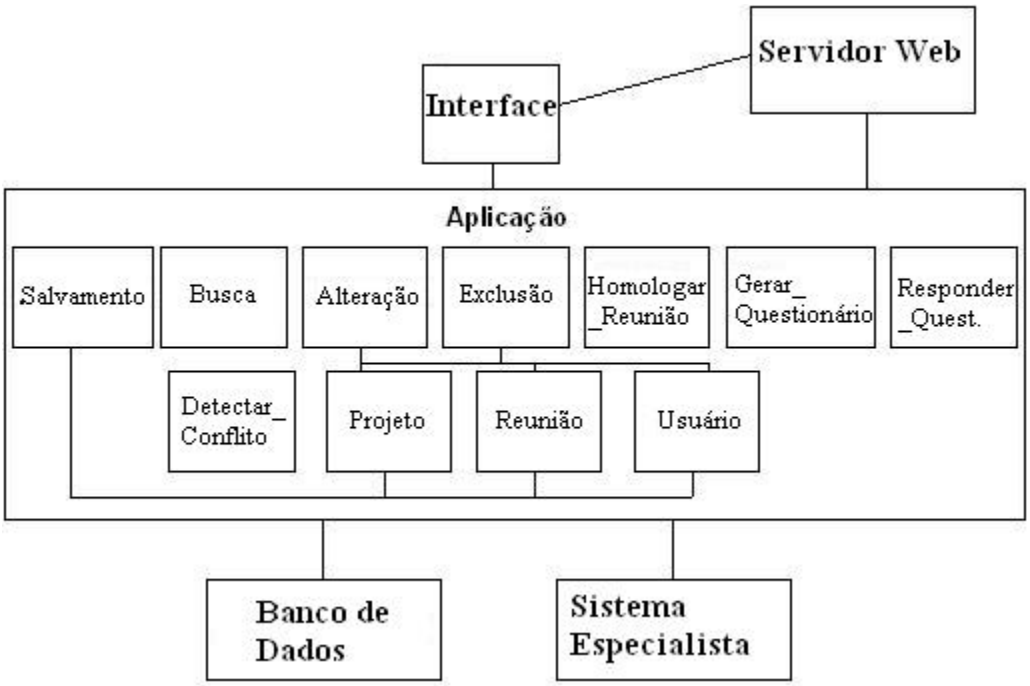

Figura 4 – Arquitetura do sistema

4.2. Interface da ferramenta

A ferramenta é divida em três módulos, cada um responsável pelas operações de um tipo de usuário. Uma para as tarefas do usuário planejador, outra para o líder e uma terceira para o participante da reunião.

A seguir apresentaremos as principais interfaces e funcionalidades da ferramenta.

4.2.1. Tela de acesso ao sistema

Na tela de entrada do sistema, o usuário digita o seu login e sua senha para ter acesso às funcionalidades do sistema que ele for autorizado, dependendo do tipo de usuário que ele represente.

Nesta tela existe uma opção que permite a pessoa interessada em utilizar a ferramenta de apoio se cadastrar como usuário do tipo planejador de reuniões, criando sua senha e seu login. A partir daí esta pessoa terá privilégio de acesso ao módulo deste tipo de usuário e poderá cadastrar o projeto que deseja elicitar os requisitos, os usuários do seu projeto, as reuniões, entre outras funcionalidades.

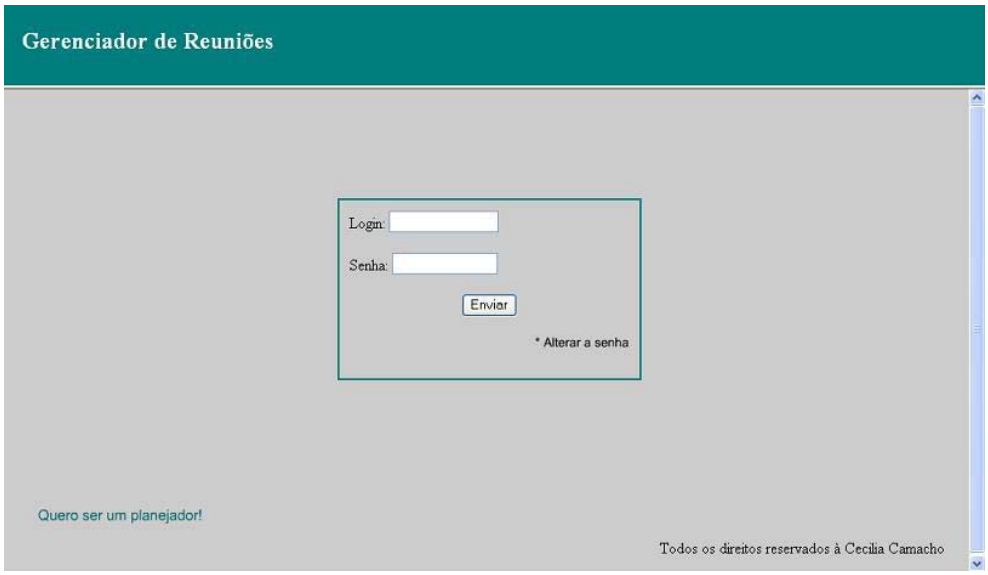

Figura 5 – Tela de acesso ao sistema

4.2.2. Tela de acesso aos módulos do líder e do participante comum

Caso o usuário seja do tipo planejador, ao fazer o login na tela de entrada ele é direcionado diretamente para o módulo de tarefas do planejador. Caso não seja, ele é direcionado então para uma tela intermediária. Nesta tela o usuário deve selecionar o projeto que ele deseja trabalhar dentre os que ele está cadastrado. Após isso, ele deve selecionar a reunião desejada, dentre as reuniões do projeto selecionado e, a seguir, dizer qual foi seu perfil na reunião selecionada, líder (moderador) ou participante comum da reunião.

Ao acionar o botão a ferramenta vai validar as informações para enviar o usuário para o módulo de líder caso ele tenha escolhido uma reunião de um projeto em que ele desempenhou este papel ou para o módulo de participante comum se esta tiver sido sua escolha.

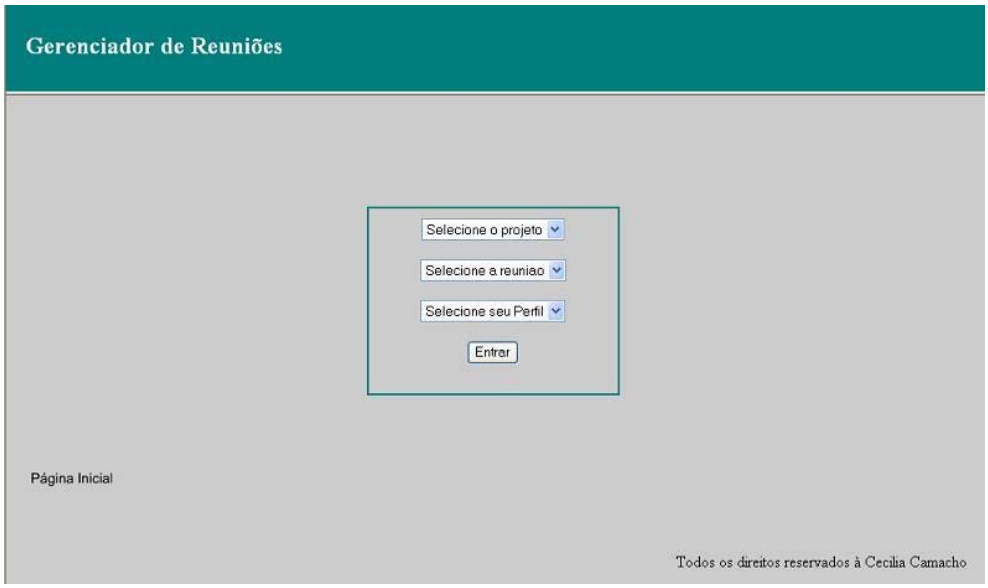

Figura 6 – Tela de acesso aos módulos do líder e do participante comum

4.2.3. Módulo dos planejadores

O usuário do tipo planejador deve entrar com sua senha para ter acesso ao seu módulo. As tarefas que podem ser realizadas pelo planejador são: Cadastro dos projetos, cadastro dos usuários e reuniões de cada projeto, cadastro dos objetivos e participantes das reuniões e operações de avaliação dos conflitos. Vamos detalhar cada uma a seguir.

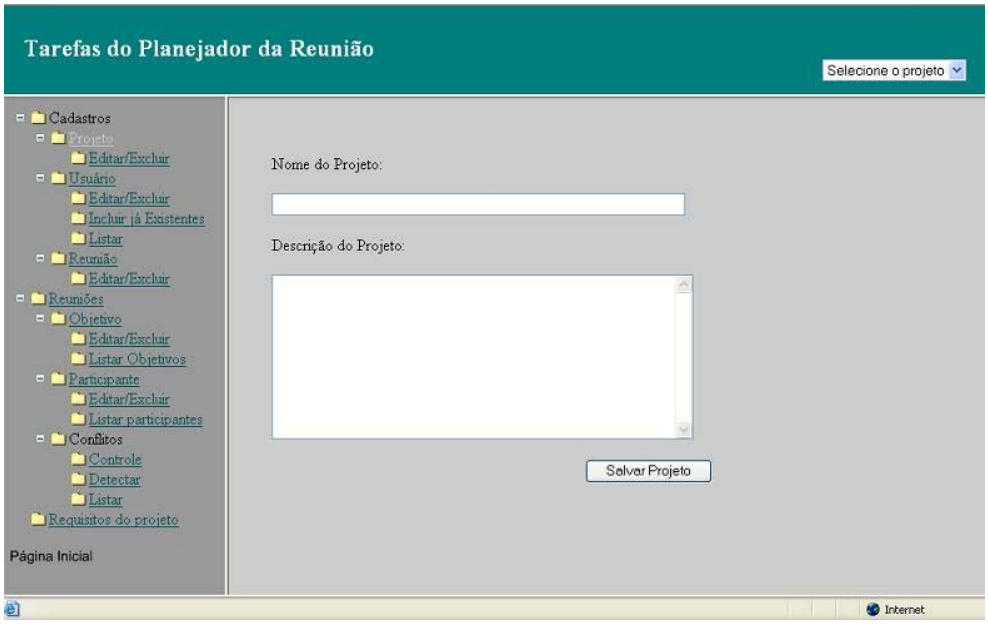

Figura 7 – Módulo planejador

4.2.3.1. Cadastro dos projetos

Esta opção permite o cadastro dos projetos que terão seus requisitos elicitados. Neste cadastro estão disponibilizadas as operações de inclusão, exclusão, alteração e consulta. Para o usuário realizar qualquer tarefa relacionada com um projeto, como incluir participante ou alterar a data de uma reunião, ele deve selecionar o projeto que deseja trabalhar em uma caixa de seleção localizada no alto da página.

4.2.3.2. Cadastro dos usuários dos projetos

Esta opção permite que o planejador cadastre os usuários que terão acesso aos projetos gerenciados por ele. Para isso existem as operações de inclusão, exclusão, busca e alteração.

Para incluir um novo usuário o planejador deverá informar o nome, o login (que deverá ser único, caso contrário a ferramenta informará e solicitará a troca), a senha provisória, o e-mail e a especificação se é um usuário do tipo planejador também ou não. Ao confirmar o cadastro, a ferramenta envia automaticamente um e-mail para o novo usuário informando que ele foi cadastrado na ferramenta e sugerindo que ele a acesse para a troca da senha temporária.

4.2.3.3. Cadastro das reuniões

Nesta opção é que os planejadores fazem o cadastramento de cada reunião a ser realizada para elicitar requisitos do projeto. Nela como nos demais cadastros, são permitidas as operações de inclusão, exclusão, busca e alteração.

Para incluir uma reunião o planejador deve informar o tema, a data e a hora da reunião.

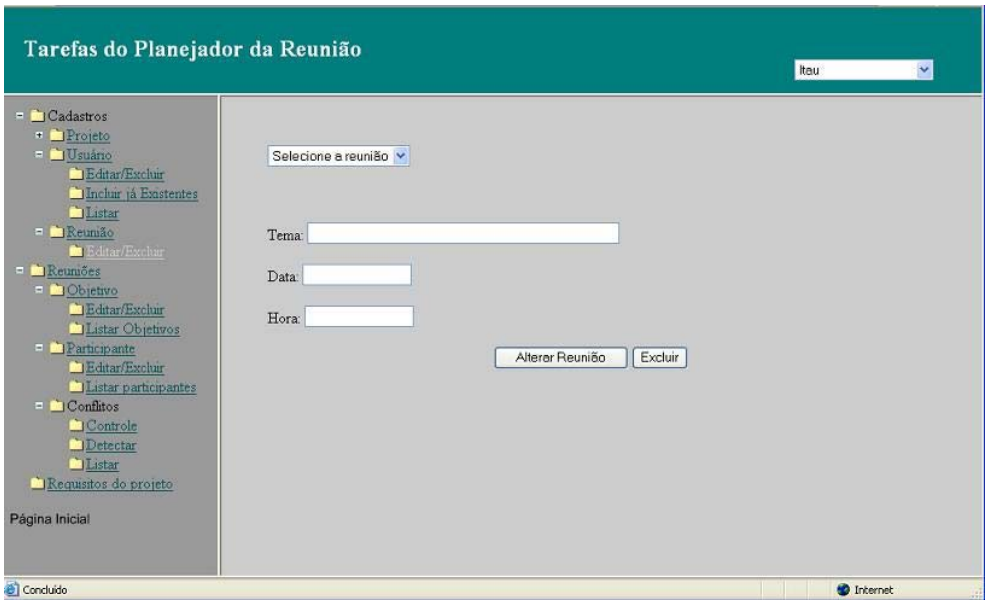

Figura 8 – Módulo do planejador (Cadastro das reuniões)

4.2.3.4. Cadastro dos objetivos da reunião

Para realizar esta tarefa o usuário deve primeiro escolher a reunião que deseja trabalhar no link "Reuniões", pai do link que cadastra os objetivos.

No cadastro dos objetivos da reunião selecionada o usuário pode incluir, excluir, alterar e listar os objetivos. Para incluir um objetivo é necessário apenas entrar com sua descrição.

4.2.3.5. Cadastro dos participantes da reunião

Como explicado na seção anterior, para ter acesso a esta tarefa o usuário deve primeiro selecionar a reunião que deseja trabalhar.

Para cadastrar os participantes da reunião o usuário deve escolher seu nome na lista de usuários do projeto para o qual a reunião será realizada e depois selecionar seu perfil, que poderá ser de participante comum ou de líder. Vale lembrar que uma reunião tem apenas um líder.

Nesta funcionalidade como nas demais de cadastro ele poderá incluir, excluir, alterar e listar os participantes.

4.2.3.6. Operações de avaliação dos conflitos

Existem três operações que são as responsáveis pelo gerenciamento dos conflitos das reuniões por parte do planejador.

A primeira é a de controle, que visa informar quais participantes já responderam ao questionário e quais ainda não responderam. Dentro dela ainda existe uma opção para enviar e-mail para os participantes que ainda não responderam, lembrando-os de realizar esta tarefa.

A segunda operação é a de detecção de conflito. Caso o número mínimo de participantes já tenha respondido ao questionário, esta opção acionará o sistema especialista ligado à ferramenta e realizará a identificação dos conflitos. Esta tarefa pode ser realizada quantas vezes o usuário quiser, pois ele pode querer ir acompanhando os resultados parciais, até todos os participantes finalmente responderem ao questionário e ele poder solicitar o resultado final.

A terceira opção é responsável por listar os conflitos que foram identificados na reunião e que devem ser tratados pelo planejador. O planejador deve tratar apenas dos conflitos relacionados com o líder (moderador) da reunião, desta maneira só estes serão listados. Os demais conflitos serão listados apenas para o líder da reunião, no módulo que contém as tarefas deste tipo de usuário.

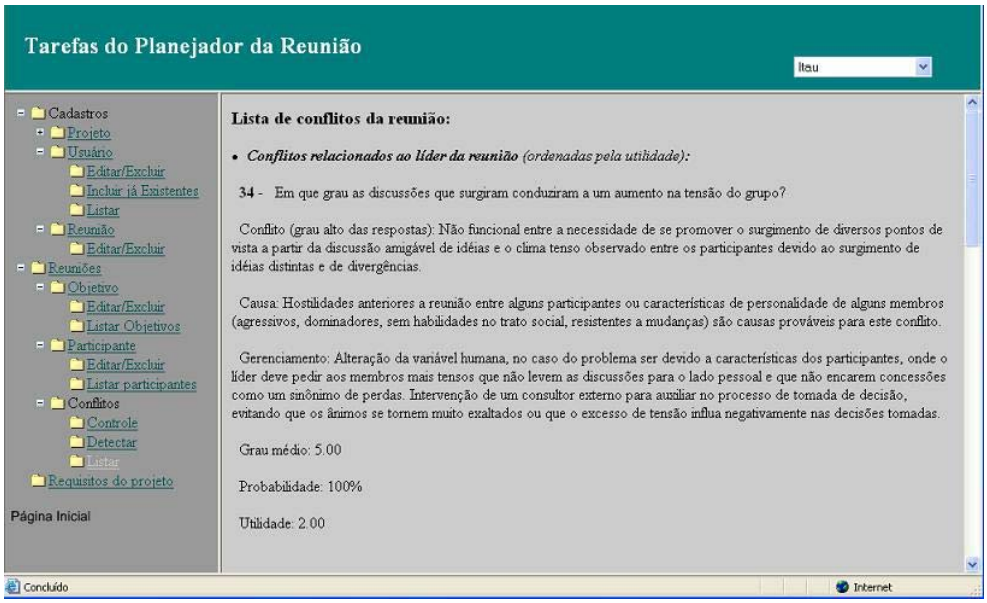

Figura 9 – Módulo do planejador (Listagem dos conflitos)

4.2.4. Módulo dos Líderes

O usuário do tipo líder deve informar o seu login e senha para acessar o sistema. Após o seu acesso ter sido validado ele será direcionado para a tela intermediária para escolher a reunião de um determinado projeto em que ele tenha desempenhado este papel, para desta maneira ter acesso ao seu módulo. As tarefas que podem ser realizadas pelo líder são: Cadastro dos requisitos da reunião, operações sobre o questionário, operações de avaliação dos conflitos e também a homologação da reunião. Vamos detalhar cada uma a seguir.

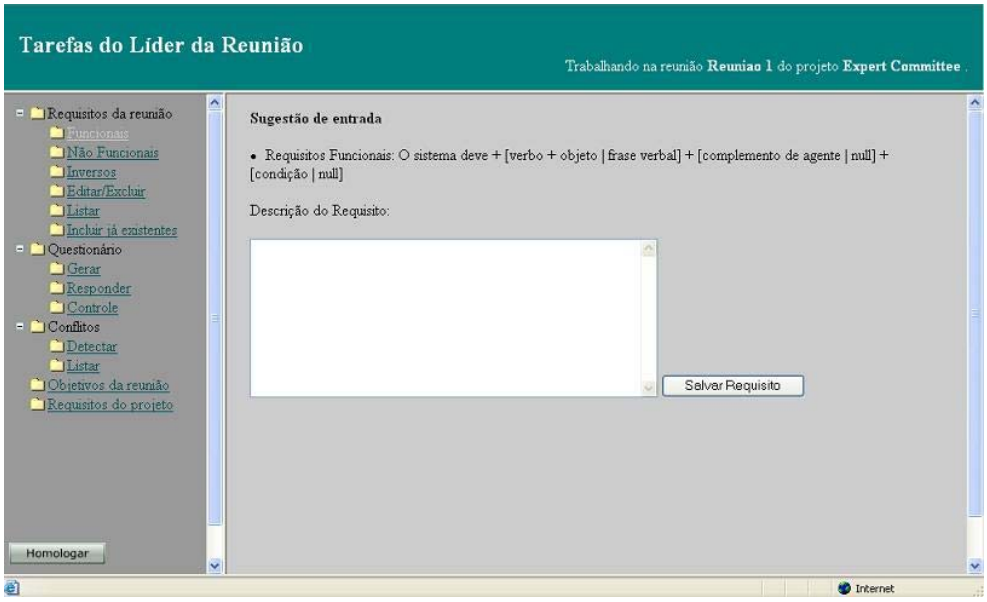

Figura 10 – Módulo do líder

4.2.4.1. Cadastro dos requisitos da reunião

Esta opção permite o cadastro dos requisitos da reunião, que é a soma dos novos requisitos elicitados e os que já foram elicitados anteriormente e que permanecem válidos. Este cadastro permite as operações de inclusão, exclusão, alteração e consulta. Para inclusão de um novo requisito, existem três opções, uma para cada tipo de requisito. Uma para requisitos funcionais, outra para não funcionais e uma terceira para inversos.

Para facilitar a inclusão de requisitos que já tenham sido elicitados anteriormente, é disponibilizada uma opção onde são listados os requisitos da reunião homologada, possibilitando a seleção dos que se deseja incluir na reunião que está sendo trabalhada. Reunião homologada é a reunião cujo conjunto de requisitos retrata os requisitos do projeto.

4.2.4.2. Operações sobre questionários

Existem três tipos de operações que podem ser realizadas sobre o questionário. A primeira delas é a geração das perguntas do questionário, onde nas questões do grupo A (analisa cada requisito individualmente), C (analisa cada objetivo) e E (cada participante) são incluídos conforme o grupo, os requisitos, objetivos e participantes da reunião que devem ser avaliados.

O líder também deve responder o questionário de perguntas, por isso também é disponibilizada esta funcionalidade para ele. Para responder o questionário o usuário deve selecionar um grau entre 1 e 5 inclusive, para avaliar o que está sendo perguntado. O usuário enquanto responde o questionário tem a opção de clicar no link "objetivo", que aparece em cada pergunta, que tem por finalidade explicar ao usuário o objetivo pelo qual está sendo feita tal pergunta.

A opção de controle que também está disponível nas operações do questionário é a mesma explicada na seção 4.2.3.6, que visa informar quais participantes já responderam ao questionário e quais ainda não responderam. Possuindo ainda a opção para enviar e-mail para os participantes que ainda não responderam, lembrando-os desta tarefa.

Mostraremos a seguir um exemplo de um questionário gerado para uma determinada reunião. Para simplificar omitiremos algumas perguntas.

1- Classifique o seu grau de ambigüidade ou falta de clareza e precisão ao final da reunião referente a cada um dos requisitos. Objetivo

1.1- Listar os usuários cadastrados

1 2 3 4 5

1.2- Não demorar mais de 10 segundos para responder à pesquisa.

1 2 3 4 5

2- Qual o grau de necessidade de especialistas adicionais que possam sanar dúvidas ou esclarecer melhor certos pontos que foram discutidos referentes ao requisito? Objetivo

2.1- Listar os usuários cadastrados

1 2 3 4 5

2.2- Não demorar mais de 10 segundos para responder à pesquisa

1 2 3 4 5

3- Qual o grau de consenso surgido entre os participantes no final da reunião em relação a cada requisito obtido? Objetivo

3.1- Listar os usuários cadastrados

1 2 3 4 5

3.2- Não demorar mais de 10 segundos para responder à pesquisa

1 2 3 4 5

11- Para cada objetivo da reunião, apresente o grau de discussão que foi gerada em torno de seus tópicos principais. Objetivo

11.1- Levantar as principais funcionalidades do sistema.

1 2 3 4 5

12- Qual o grau de viabilidade de cada objetivo da reunião? Objetivo

12.1- Levantar as principais funcionalidades do sistema.

1 2 3 4 5

13- Qual o grau de compatibilidade entre os objetivos da reunião? Objetivo

1 2 3 4 5

14- Em que grau os objetivos da reunião foram bem definidos antes do seu inicio ? Objetivo

1 2 3 4 5

16- Qual o grau em que cada membro usou de coerção (ameaças) para fazer prevalecer seus pontos de vista durante a reunião? Objetivo

16.1- Andréa Vilas

1 2 3 4 5

16.2- Carlos José

1 2 3 4 5

17- Qual o grau de participação de cada indivíduo durante a reunião? Objetivo

17.1- Andréa Vilas

1 2 3 4 5

17.2- Carlos José

1 2 3 4 5

35- Qual a qualidade do controle e da cooperação de atividades efetuados pelo líder durante a reunião? Objetivo

1 2 3 4 5

37- Qual o seu grau de satisfação em relação à maneira a qual a reunião foi realizada. Objetivo

1 2 3 4 5

39- Qual o grau em que os compromissos firmados entre os participantes serão suficientes para garantir que os requisitos sejam mantidos e atendidos pelo sistema? Objetivo

1 2 3 4 5

4.2.4.3. Operações de avaliação dos conflitos

Operações semelhantes foram explicadas no módulo do planejador na seção 4.2.3.6. A operação de detecção do conflito é a mesma explicada naquela seção, já a operação de listar os conflitos apresenta uma diferença. Em vez de listar os conflitos relacionados com o líder, como é feito no módulo do planejador, no módulo do líder são listados todos os conflitos identificados na reunião, exceto, os relacionados com o líder que já são listados no módulo do planejador como já foi dito.

O relatório com a listagem dos conflitos é divido em quatro partes: conflitos identificados nas questões que analisam cada requisito individualmente, conflitos identificados nas questões que analisam cada objetivo individualmente, conflitos identificados nas questões que analisam cada participante individualmente e conflito nas demais questões. Dentro de cada divisão desta os conflitos são ordenados pela sua utilidade, para facilitar o líder a identificar os conflitos mais prioritários de cada tipo.

A seguir daremos um exemplo resumido da listagem de conflitos de uma reunião.

- Conflito nas questões sobre cada requisito (ordenadas pela utilidade):

Pergunta número 1 - Classifique o seu grau de ambigüidade ou falta de clareza e precisão ao final da reunião referente a cada um dos requisitos.

Conflito (grau alto das respostas): Não funcional entre a necessidade de se ter requisitos claros e bem definidos e o grau de ambigüidade ou de falta de clareza e precisão detectado.

Causa(s):
2 - Alto grau de necessidade de especialistas adicionais que possam sanar dúvidas ou esclarecer melhor certos pontos que foram discutidos referentes ao requisitos.
15 - Tempo de duração da reunião.

Gerenciamento: Na próxima reunião, o líder deve propor um esclarecimento do(s) requisitos ambíguo(s) e impreciso(s), onde os participantes devem colocar os pontos onde o(s) requisito(s) estão mal definidos visando uma melhora do seu entendimento.

Requisitos onde o conflito foi detectado (ordenados por utilidade):

* O sistema deve atualizar dados do diretor
  Grau_medio: 4.33 Probabilidade: 83% Utilidade: 1.10
* O sistema não deve demorar mais que 20 segundos para dar um resultado.

Grau_medio: 4.33 Probabilidade: 66% Utilidade: 0.88

* O sistema não deve parar de funcionar.

Grau_medio: 4.00 Probabilidade: 50% Utilidade: 0.50

- Conflito nas questões sobre cada objetivo (ordenadas pela utilidade):

Pergunta número 11 - Para cada objetivo do sistema, apresente o grau de discussão que foi gerada em torno de seus tópicos principais.

Conflito (grau baixo das respostas): Ausência do conflito funcional gerado pelas discussões sobre os objetivos traçados.

Causa(s):
15 - Tempo de duração da reunião.
35 - Baixa qualidade do controle e da cooperação de atividades efetuados pelo líder durante a reunião.

Gerenciamento: Na próxima reunião, o líder deve enfatizar as discussões em torno dos objetivos que tiveram pouca ênfase nas reuniões anteriores, cuidando para que os participantes não se desviem do assunto discutido.

Objetivos onde o conflito foi detectado (ordenados por utilidade):

* Criar empregos

Grau_medio: 2.00 Probabilidade: 50% Utilidade: 0.50

* aumentar taxas de valores

Grau_medio: 2.50 Probabilidade: 50% Utilidade: 0.25

- Conflito nas questões sobre cada participante (ordenadas pela utilidade):

Pergunta número 16 - Qual o grau em que cada membro usou de coerção (ameaças) para fazer prevalecer seus pontos de vista durante a reunião?

Conflito (grau alto das respostas): Não funcional entre um dos objetivos do método, o de levar em conta as posições individuais para o alcance de soluções e o uso da coerção, limitando ou impedindo o surgimento de soluções que pudessem levar em conta diferentes posições.

Causa: O participante que usou de coerção possui alguma vantagem em relação aos demais, seja por possuir um status maior ou por ocupar um cargo melhor na organização ou por possuir trunfos que o permite persuadir os demais membros da reunião. Desta forma, o indivíduo utiliza a coerção como forma de fazer valer suas idéias, prejudicando os resultados da reunião.

Gerenciamento: O líder deve propor que sejam levadas em conta as posições dos participantes que sofreram coerção no estabelecimento das decisões. Deve, ainda, conscientizar o indivíduo que usou de ameaças de que as idéias de todos devem ser consideradas e que ninguém deve usar tentativas de coerção, pois a opinião dos outros membros também é vital para a obtenção dos requisitos.

Participantes onde o conflito foi detectado (ordenados por utilidade):

* Carlos

Grau_medio: 5.00 Probabilidade: 100% Utilidade: 2

* Andrea Villas

Grau_medio: 2.00 Probabilidade: 50% Utilidade: 0.50

- Conflito nas demais questões (ordenadas pela utilidade):

Pergunta número 8 - Qual o grau de compatibilidade entre os requisitos do sistema obtidos na reunião?

Hipótese 1: A incompatibilidade entre os requisitos é reflexo da má definição ou da incompatibilidade entre os objetivos do sistema.

Conflito (grau baixo das respostas): Não funcional, pois a realização completa de um requisito implica em uma não realização total ou parcial de outros requisitos.

Causa(s): Não detectada!

Gerenciamento: A redefinição dos objetivos auxiliará na resolução da incompatibilidade entre os requisitos, pois alguns requisitos terão que ser revistos e alterados em função da mudança dos objetivos.

Hipótese 2: Os objetivos do sistema estão bem definidos e são compatíveis entre si, mas os requisitos apresentam incompatibilidades.

Conflito (grau baixo das respostas): Não funcional, pois a realização completa de um requisito implica em uma não realização total ou parcial de outros requisitos.

Causa(s):
2 - Alto grau de necessidade de especialistas adicionais que possam sanar dúvidas ou esclarecer melhor certos pontos que foram discutidos referentes ao requisitos.
15 - Tempo de duração da reunião.
16 - Alto grau de coerção (ameaças) usado pelos membros.
35 - Baixa qualidade do controle e da cooperação de atividades efetuados pelo líder durante a reunião.

Gerenciamento: Alterar na próxima reunião, através da negociação entre os participantes, as partes dos requisitos que possuem incompatibilidades, eliminando seus conflitos e permitindo sua completa realização simultaneamente.

Grau médio: 2.17

Probabilidade: 66 %

Utilidade: 0.55

4.2.4.4. Homologar reunião

Esta opção homologa uma reunião, o que significa que a lista de requisitos desta reunião representa a lista de requisitos do projeto.

4.2.5. Módulo do participante comum

O usuário do tipo participante comum deve informar o seu login e senha para acessar o sistema. Após o seu acesso ter sido validado ele será direcionado para a tela intermediária para escolher a reunião de um determinado projeto em que ele tenha desempenhado este papel, para desta maneira ter acesso ao seu módulo. A principal tarefa disponível para o participante comum é a opção de responder o questionário, já explicada na seção 4.2.4.2., além de poder solicitar a visualização da lista de requisitos do projeto e dos objetivos da reunião como também é possível nos demais módulos.

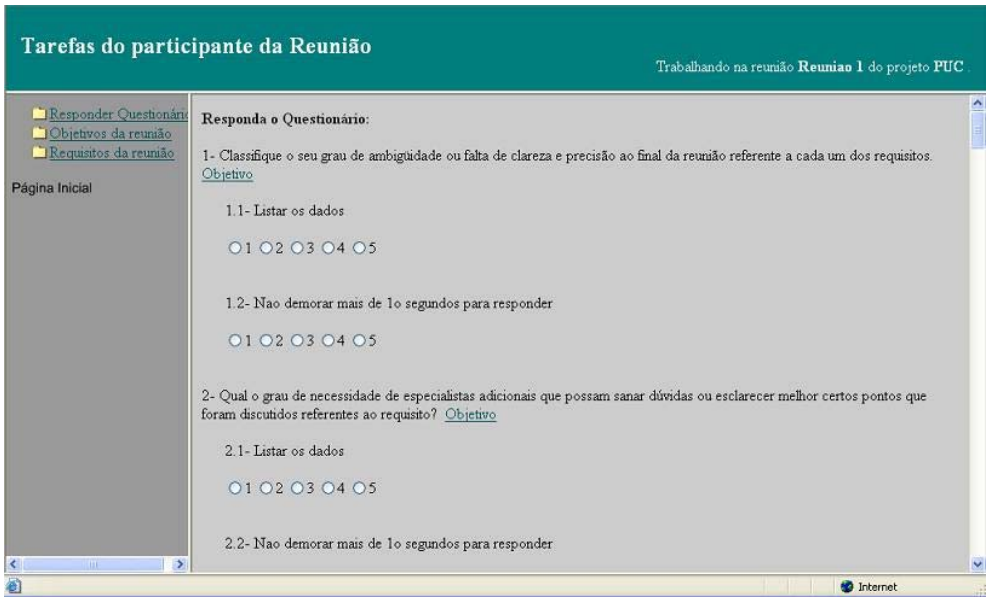

Figura 11 – Módulo do participante comum

4.3. Grafo de relacionamento das tabelas implementadas

Para modelar a base de dados utilizamos um grafo que defini os objetos que compõem o sistema e seus respectivos relacionamentos. As caixas representam as tabelas e as setas os seus relacionamentos.

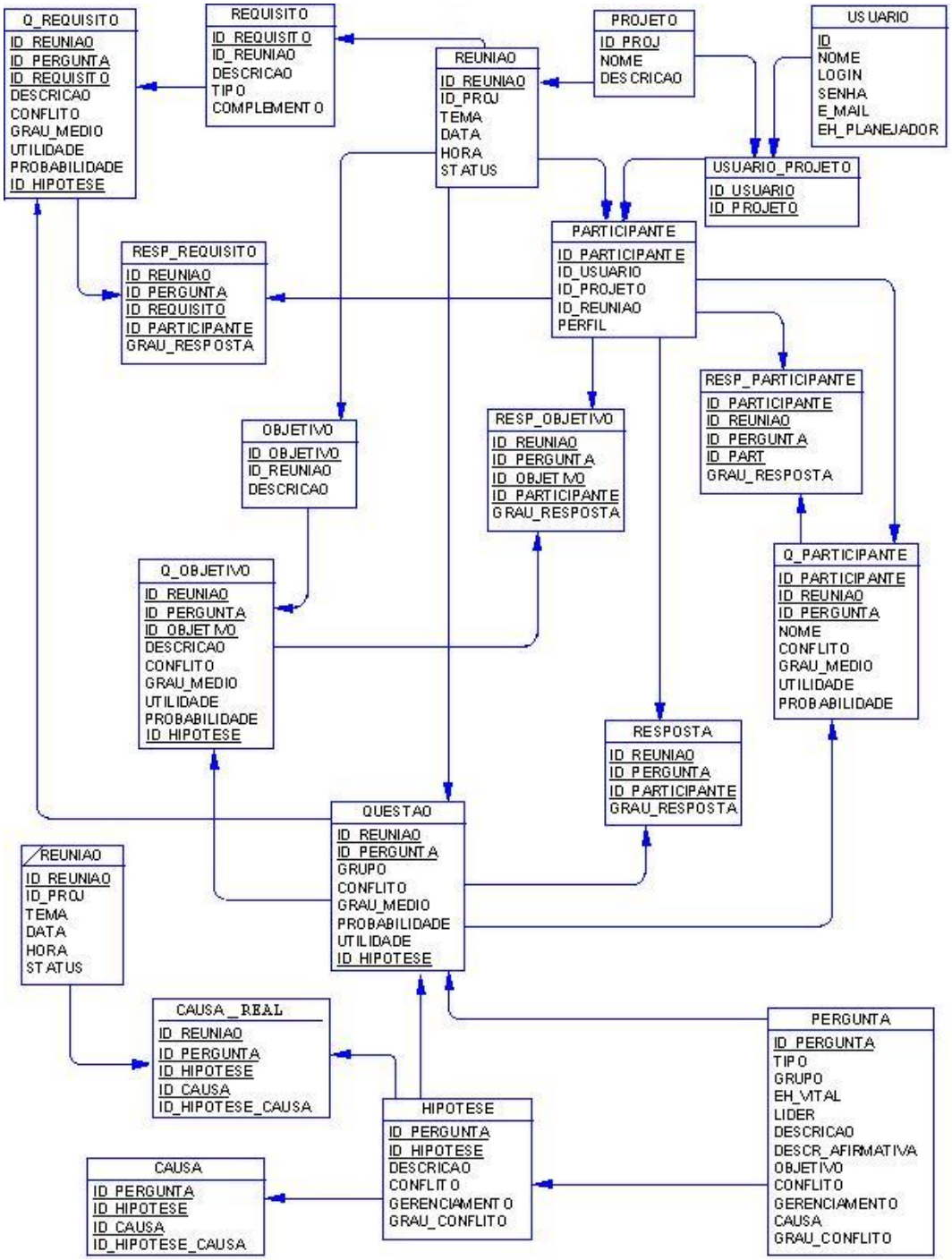

Figura 12 – Grafo de relacionamento das tabelas implementadas

4.4. Sistema especialista para detecção dos conflitos

Nesta seção falaremos sobre o sistema especialista desenvolvido para realizar a detecção dos conflitos da reunião.

4.4.1. O que é um Sistema Especialista

Edward Feigenbaum, professor da Universidade de Stanford, pioneiro na tecnologia de sistemas especialistas - S.E.'s, define-os como "... um programa inteligente de computador que usa conhecimento e procedimentos de inferência para resolver problemas que são difíceis o suficiente para que sua solução necessite de um grau significativo de perícia humana." [Feigenbaum 63] . Isto é, um sistema especialista é um sistema de computação que emula a habilidade de tomar decisões de um especialista humano.

O termo emular significa que se pretende que o sistema especialista aja em todos os sentidos como um especialista humano. Uma emulação é muito mais forte que uma simulação. Simulações agem como o elemento real em apenas alguns aspectos.

Uma característica fundamental em sistemas especialistas é que o conhecimento do(s) especialista(s) é armazenado através de regras. Regras, neste contexto, são construções simples na forma: Se alguma condição é verdadeira, então faça alguma coisa. O uso de regras é interessante em algumas situações tendo em vista que permite a construção da definição de um determinado comportamento de maneira compacta.

Se analisado em sua forma mais simples, um determinado comportamento caracteriza-se por um conjunto de ações e um conjunto de condições sob as quais as ações devem acontecer. Um sistema especialista continuamente analisa um conjunto de elementos condicionais (conhecidos como regras) em relação a uma base de fatos (também conhecida como base de conhecimento) para verificar se alguma regra pode ser aplicada para aquele conjunto de fatos. Se a premissa é verdadeira, então o sistema especialista executa as ações associadas gerando novos fatos na base de conhecimento. Este processo de analisar a base e disparar regras é realizado por um componente denominado *máquina de inferência*.

Um sistema especialista (SE) deve possuir alguns componentes essenciais [Farreny 1985]:

• uma linguagem de expressão dos conhecimentos fornecidos pelos especialistas;

• uma base de conhecimentos, para armazenar o conhecimento específico de determinada aplicação, que pode ser diretamente fornecido por um especialista, ou acumulado pelo sistema ao fim dos experimentos;

• um motor de inferência, programa relativamente geral que explora o conhecimento da base de conhecimentos, considerando-a como fonte de informações.

A aplicação desta tecnologia apresenta uma série de vantagens, dentre as quais, algumas são citadas por Gardner [Gardner 92] a seguir:

• solucionar problemas importantes que, de outro modo, deveriam ser solucionados por um perito humano;

• flexibilidade na integração de novos conhecimentos ao conhecimento já armazenado;

• capacidade de mostrar seu conhecimento de uma forma facilmente compreensível;

• capacidade de tratar sentenças simples em linguagens naturais.

4.4.2. Ambiente de Desenvolvimento

Utilizamos como ambiente de desenvolvimento o software CLIPS (C Language Integrated Production System) desenvolvido pela NASA/Johnson Space Center.

Por volta de 1985, a NASA decidiu unificar o desenvolvimento interno de sistemas especialistas. O sistema usado até então, ART, não estava definido para todas as plataformas utilizadas pela companhia, especificamente PCs e Macintoshs. Como resultado, o setor de tecnologia de software do Johnson Space Center criou um clone das capacidades de encadeamento progressivo e da sintaxe de ART, introduzindo o CLIPS (C Language Integrated Production System) [Giarratano 1998, Lopez 04] no domínio público, com a finalidade de gerar soluções que apresentassem alta portabilidade, baixo custo e fácil integração com sistemas externos.

CLIPS rapidamente conseguiu um grande número de usuários, principalmente no meio acadêmico. Sua sintaxe em nada se assemelha à da linguagem hospedeira (C). Entretanto, a sua semelhança com a sintaxe de outras

linguagens usadas pela comunidade de Inteligência Artificial, notadamente LISP, permitiu o aumento do número de seus usuários [Giarratano 1998].

A representação de conhecimento em CLIPS pode ser feita de três formas: regras (de produção), usadas principalmente para a representação de conhecimento heurístico baseado em experiência; funções e funções genéricas, para representação de conhecimento procedimental; ou programação orientada a objetos, também usada na representação de conhecimento procedimental através da linguagem de programação própria (COOL, ou CLIPS Object-Oriented Language). No nosso trabalho nós utilizamos a representação do conhecimento na forma de regras de produção.

Uma outra característica importante é a forma como é realizada a unificação entre os objetos da base de fatos e as regras. O sistema utiliza o algoritmo Rete [Forgy 1982], que elimina redundâncias de avaliações das pré-condições das regras. Por exemplo, se uma regra pode ser disparada com um conjunto de objetos, e uma segunda regra é disparada e não altera nenhum dos objetos do primeiro conjunto, então não há necessidade de se testar novamente as pré-condições da primeira, pois ela ainda está ativa com o mesmo conjunto de instâncias. A idéia por trás do algoritmo é que apenas os objetos modificados no disparo da última regra tenham que ser testados novamente para verificar se tornam alguma outra regra ativa (ou desativa). O algoritmo Rete é usado na maioria dos sistemas de produção existentes, apresentando um ótimo desempenho para aplicações de porte razoável.

4.4.3. Representação do Conhecimento

Em CLIPS há uma diferença entre tipos estruturados de dados e classes, de forma similar à diferença entre os construtores struct e class em C++. Tipos, ou templates definem um agrupamento de valores, possibilitando o valor dos atributos serem fatos, e não apenas valores atômicos.

Cada estrutura se subdivide em campos que identificam um atributo importante das perguntas e das causas.

No caso das estruturas de perguntas temos:

número - o número da pergunta;

tipo - se ela é absoluta(1) ou relativa(2);

a hipótese - o número da hipótese se for pergunta relativa ou 0 se for absoluta

grau_medio - o grau médio das respostas a esta pergunta;

grau_conflito – (1) Se identificar conflito com o grau alto das respostas ou (0) se identificar conflito com grau baixo das respostas.

conflito – se ocorreu conflito (1) ou não (nil);

utilidade – o valor da utilidade, ou seja, da prioridade do conflito;

probabilidade – valor da probabilidade que pode ter ocorrido o conflito;

identificador – no caso de pergunta_requisito ou pergunta_objetivo ou pergunta_participante o identificador é o número que identifica o requisito ou objetivo ou participante respectivamente.

No caso da estrutura de causa temos:

pergunta – o número da pergunta relativa

hipótese_pergunta – número da hipótese da pergunta relativa

causa – número da pergunta que é uma possível causa do conflito identificado pela pergunta relativa citada acima.

hipótese_causa – caso a pergunta possua mais de uma hipótese, o número da hipótese que identifica a possível causa aparecerá neste campo, senão aparecerá o número 0.

eh_causa - (1) se a causa ocorrer, ou permanece nil se não ocorrer.

Segue abaixo o script de criação das estruturas de dados:

```
(deftemplate pergunta
(slot numero ) (slot tipo ) (slot hipotese) (slot grau_medio) (slot grau_conflito)
(slot conflito) (slot utilidade) (slot probabilidade)
)
(deftemplate pergunta_requisito
(slot numero ) (slot tipo ) (slot hipotese) (slot identificador) (slot grau_medio)
(slot grau_conflito) (slot conflito) (slot utilidade) (slot probabilidade)
)
(deftemplate pergunta_objetivo
```

```
(slot numero ) (slot tipo ) (slot hipotese) (slot identificador) (slot grau_medio)
(slot grau_conflito) (slot conflito) (slot utilidade) (slot probabilidade)
)
(deftemplate pergunta_participante
(slot numero ) (slot tipo ) (slot hipotese) (slot identificador) (slot grau_medio)
(slot grau_conflito) (slot conflito) (slot utilidade) (slot probabilidade)
)

(deftemplate causa
(slot pergunta) (slot hipotese_pergunta) (slot causa) (slot hipotese_causa)
(slot eh_causa)
)
```


Fatos são colocados na base de fatos através da utilização do comando assert, ou através do comando load que carrega um arquivo txt com os fatos. Os atributos não inicializados recebem o valor nulo (nil).

Segue abaixo um exemplo do conjunto de fatos para identificar a pergunta 9 e suas causas:

Pergunta no formato texto ->

9 - Qual a qualidade da documentação que foi elaborada para representar os requisitos.

Objetivo: Verificar a qualidade da documentação dos requisitos, pois uma documentação insuficiente dificultará a transmissão de conhecimento para os demais membros da equipe de desenvolvimento do sistema além de sujeitar a distorções e a esquecimentos dos requisitos que foram obtidos.

Tipo da questão: relativa.

Conflito: Não funcional entre a qualidade desejada da documentação dos requisitos e a documentação apresentada, comprometendo seriamente a qualidade do projeto.

Gerenciamento: Documentar os requisitos obtidos com a participação de todos os membros da reunião, para evitar distorções e tendências na documentação que for elaborada.

Causas: 15 – Primeira hipótese: Baixo tempo de duração da reunião em função dos objetivos do sistema ou **35 –** Baixa qualidade do controle e da coordenação de atividades efetuadas pelo líder ou **37 –** Baixo grau de satisfação em relação à maneira a qual a reunião foi realizada.

Intensidade das notas recebidas na reunião x: 2

Probabilidade da ocorrência deste conflito na reunião x: 60%

Pergunta com seus dados em formato de fatos:

```
(pergunta (numero 9)(tipo 2)(hipotese 1)(grau_medio 2)
(grau_conflito nil) (conflito nil) (utilidade nil) (probabilidade
60))
(causa (pergunta 9) (hipotese_pergunta 1) (causa
15)(hipotese_causa 1 ))
(causa (pergunta 9) (hipotese_pergunta 1) (causa
35)(hipotese_causa 1 ))
(causa (pergunta 9) (hipotese_pergunta 1) (causa
37)(hipotese_causa 0 ))
```

O campo eh_causa da estrutura de dados da causa não aparece, pois é inicializado com valor nulo, já que é durante a execução das regras no sistema especialista que é acertado seu valor.

4.4.4. A Implementação

Para um melhor entendimento, explicaremos um pouco sobre a estrutura de criação de uma regra. As regras no CLIPS são definidas através do comando defrule. Para construção das regras, você pode utilizar variáveis. As variáveis são utilizadas para casar nos pré-requisitos de uma regra os valores dos fatos contidos na base de conhecimento, para serem usados depois pela regra em testes ou como base para outros cálculos. Vejamos por exemplo, a regra que verifica se uma pergunta que identifica o conflito com grau alto das respostas detectou conflito:

```
(defrule verif_conflito_alto
(or   ?id <- (pergunta (grau_medio ?x)(grau_conflito 1)(conflito
nil))
      ?id <- (pergunta_objetivo (grau_medio ?x)(grau_conflito 1)
(conflito nil))
```

```
      ?id <- (pergunta_requisito (grau_medio ?x)(grau_conflito 1)
(conflito nil))
      ?id    <-    (pergunta_participante    (grau_medio    ?x)
(grau_conflito 1) (conflito nil))
)
      (test (>= ?x 3.5))
=>
     (modify ?id (conflito 1)))
```

Como podemos observar, o que está contido antes da seta são os pré-requisitos e após são as atividades que devem ser realizadas caso o fato satisfaça os pré-requisitos.

Todos os fatos vão passar pelas linhas de pré-requisitos, testando se os satisfazem. No caso do exemplo, os fatos vão tentar se encaixar em um dos tipos de estruturas de dados solicitados, não em todos, já que estes estão dentro de uma expressão "ou". Para satisfazer este pré-requisito o fato do tipo pergunta ou pergunta_requisito ou pergunta_objetivo ou pergunta_participante deve ter o grau do conflito igual a 1, ou seja alto, e a informação sobre o conflito igual a nil, ou seja, não foi detectado conflito ainda para esta pergunta.

Para o fato que conseguir este casamento iremos usar as variáveis ?x para guardar o valor do campo grau médio. E utilizaremos também a variável ?id, que guardará o valor do registro daquele fato na base de conhecimento.

Depois, como mostra na regra exemplificada acima, o fato deverá passar pelo teste (test) que verifica se a variável ?x, que guarda o valor do campo grau médio, é maior ou igual a 3.5. Se este pré-requisito também for satisfeito o fato sofrerá a modificação imposta pela regra, que neste caso é modificar o fato identificado pelo registro guardado na variável ?id, atualizando o campo conflito.

Para a implementação do sistema foram criadas onze regras, elas permitem obter o grau do conflito de cada pergunta, verificar se a pergunta identificou algum conflito e verificar as causas dos conflitos identificados por perguntas do tipo relativas. Além de possibilitar a limpeza do conjunto de fatos resultante.

A seguir apresentaremos cada regra que foi construída, e junto mostraremos através de um exemplo, como os fatos são modificados por elas.

Para isso selecionamos dois conjuntos de exemplos de fatos, o grupo 1 são fatos que caracterizam a pergunta número 9, que é uma pergunta relativa, sem

subdivisão e contém apenas uma hipótese. Sua causa esta relacionada com os resultados de outras 3 perguntas e suas respectivas hipóteses (pergunta 15 - hipótese 1, pergunta 35 - hipótese 1, pergunta 37 - hipótese 0).

O grupo 2 são os fatos referentes a pergunta número 4, que é uma pergunta absoluta, com subdivisão e com causa pré-determinada, ou seja, sua causa não depende da correlação com outras perguntas, por isso não aparece como fato. Ela é uma pergunta com subdivisão, pois pertence ao grupo A, perguntas que analisam cada requisito, então será subdividida conforme o número de requisitos resultantes da reunião, para que cada requisito seja analisado separadamente baseado em suas avaliações individuais (vide capítulo 3, seção 3.4). O campo "identificador" da estrutura pergunta_requisito corresponde ao identificador do requisito, como já foi explicado anteriormente.

Estado inicial dos fatos:

Grupo 1

```
(pergunta (numero 9 ) (tipo 2 ) (hipotese 1 ) (grau_medio 2 ) (grau_conflito nil) (conflito nil) (utilidade nil) (probabilidade 50) )
(causa (pergunta 9 ) (hipotese_pergunta 1 )(causa 15 )(hipotese_causa 1 ))
(causa (pergunta 9 ) (hipotese_pergunta 1 )(causa 35 )(hipotese_causa 1 ))
(causa (pergunta 9 ) (hipotese_pergunta 1 )(causa 37 )(hipotese_causa 0 ))
```

Grupo 2

```
(pergunta_requisito (numero 4) (tipo 1) (hipotese 0) (identificador 7) (grau_medio 4 ) (grau_conflito nil) (conflito nil) (utilidade nil) (probabilidade 75))
(pergunta_requisito (numero 4) (tipo 1) (hipotese 0) (identificador 8) (grau_medio 3.4 ) (grau_conflito nil) (conflito nil) (utilidade nil) (probabilidade 57))
```

Segue o conjunto de regras criadas para a realização do sistema, na ordem em que são acionadas:

1) As regras grau_conflito_alto e grau_conflito_baixo são as primeiras a serem acionadas, pois são as únicas que no momento os fatos satisfazem os pré-requisitos. O objetivo é detectar qual o grau da resposta que identifica um conflito na pergunta, grau alto ou baixo.

- Regra para identificar quais perguntas irão detectar conflito com o grau alto das respostas

```
( defrule grau_conflito_alto
(or         ?id  <-  (pergunta  (numero  ?x)  (hipotese  ?y)
(grau_conflito nil) )
            ?id <- (pergunta_objetivo (numero ?x) (hipotese ?y)
(grau_conflito nil))
            ?id  <-  (pergunta_requisito  (numero  ?x)  (hipotese
?y)(grau_conflito nil))
            ?id <- (pergunta_participante (numero ?x) (hipotese
?y)(grau_conflito nil))
)
(or
(test (= ?x 1)) (test (= ?x 2)) (test (= ?x 4)) (test (= ?x
7)) (test (= ?x 16))
(test (= ?x 18)) (test (= ?x 19)) (test (= ?x 20)) (test (=
?x 27))(test (= ?x 28))
(test (= ?x 29)) (test (= ?x 34)) (test (= ?x 36))
(and (test (= ?x 15)) (or (test (= ?y 2)) (test (= ?y 3))))
)
=>
(modify ?id (grau_conflito 1))
)
```

- Regra para identificar quais perguntas irão detectar conflito com o grau baixo das respostas

```
( defrule grau_conflito_baixo
(or         ?id  <-  (pergunta  (numero  ?x)  (hipotese  ?y)
(grau_conflito nil))
            ?id  <-  (pergunta_objetivo  (numero  ?x)  (hipotese
?y)(grau_conflito nil))
            ?id  <-  (pergunta_requisito  (numero  ?x)(hipotese
?y)(grau_conflito nil))
```

```
            ?id <- (pergunta_participante (numero ?x) (hipotese
?y) (grau_conflito nil))
     )
     (or
     (test (= ?x 3)) (test (= ?x 5)) (test (= ?x 6))
     (and (test (>= ?x 8)) (test (<= ?x 14)))
     (test (= ?x 17))
     (and(test (>= ?x 21)) (test (<= ?x 26)))
     (and (test (>= ?x 30)) (test (<= ?x 33)))
     (test (= ?x 35)) (test (= ?x 37)) (test (= ?x 38)) (test (=
?x 39))
     (and (test (= ?x 15)) (test (= ?y 1)))
     )
     =>
     (modify ?id (grau_conflito 0))
     )
```

Estado dos fatos:

Grupo 1

```
     (pergunta (numero 9) (tipo 2) (hipotese 1) (grau_medio 2)
(grau_conflito 0) (conflito nil) (utilidade nil) (probabilidade
50))
     (causa (pergunta 9 ) (hipotese_pergunta 1 )(causa 15
)(hipotese_causa 1 ))
     (causa (pergunta 9 ) (hipotese_pergunta 1 )(causa 35
)(hipotese_causa 1 ))
     (causa (pergunta 9 ) (hipotese_pergunta 1 )(causa 37
)(hipotese_causa 0 ))
```

Grupo 2

```
     (pergunta_requisito (numero 4) (tipo 1) (hipotese 0)
(identificador 7) (grau_medio 4) (grau_conflito 1) (conflito nil)
(utilidade nil) (probabilidade 75))
     (pergunta_requisito (numero 4) (tipo 1) (hipotese 0)
(identificador 8) (grau_medio 3.6) (grau_conflito 1) (conflito
nil) (utilidade nil) (probabilidade 57))
```

2) As regras que são acionadas neste momento são a verif_conflito_alto, verif_conflito_baixo, verif_conflito_alto_probabilidade e verif_conflito_baixo_

probabilidade, estas regras tem como objetivo identificar em quais perguntas ocorreram conflito.

- Regra que verifica qual pergunta detectou conflito, para as perguntas que detectam conflito com grau alto das respostas, verificando apenas se o grau médio é maior ou igual a 3.5.

```
(defrule verif_conflito_alto
 (or   ?id <- (pergunta (grau_medio ?x) (grau_conflito 1)
(conflito nil))
            ?id    <-    (pergunta_objetivo    (grau_medio    ?x)
(grau_conflito 1) (conflito nil))
            ?id    <-    (pergunta_requisito    (grau_medio    ?x)
(grau_conflito 1) (conflito nil))
            ?id   <-   (pergunta_participante   (grau_medio   ?x)
(grau_conflito 1) (conflito nil))
)
(test (>= ?x 3.5))
=>
(modify ?id (conflito 1)))
```

- Regra que verifica qual pergunta detectou conflito, para as perguntas que detectam conflito com grau baixo das respostas, verificando apenas se o grau médio é menor ou igual a 2.5.

```
(defrule verif_conflito_baixo
(or   ?id <- (pergunta (grau_medio ?x) (grau_conflito 0)
(conflito nil))
           ?id    <-    (pergunta_objetivo    (grau_medio    ?x)
(grau_conflito 0) (conflito nil))
           ?id    <-    (pergunta_requisito    (grau_medio    ?x)
(grau_conflito 0) (conflito nil))
           ?id   <-   (pergunta_participante   (grau_medio   ?x)
(grau_conflito 0) (conflito nil))
)
  (test (<= ?x 2.5))
=>
(modify ?id (conflito 1)))
```

- Regra que verifica qual pergunta detectou conflito, para as perguntas que detectam conflito com grau alto das respostas, verificando se o grau médio é maior ou igual a 3.4 e se a probabilidade de ocorrência do conflito é maior ou igual a 55 %.

```
(defrule verif_conflito_alto_probabilidade
 (or   ?id <- (pergunta (grau_medio ?x) (grau_conflito 1)
(conflito nil) (probabilidade ?y))
        ?id   <-   (pergunta_objetivo   (grau_medio   ?x)
(grau_conflito 1) (conflito nil) (probabilidade ?y))
        ?id   <-   (pergunta_requisito   (grau_medio   ?x)
(grau_conflito 1) (conflito nil) (probabilidade ?y))
        ?id   <-   (pergunta_participante   (grau_medio   ?x)
(grau_conflito 1) (conflito nil) (probabilidade ?y))
)
(and (test (>= ?x 3.4))
     (test (>= ?y 55))
)

=>
(modify ?id (conflito 1) ) )
```

- Regra que verifica qual pergunta detectou conflito, para as perguntas que detectam conflito com grau baixo das respostas, verificando se o grau médio é menor ou igual a 2.6 e se a probabilidade de ocorrência do conflito é maior ou igual a 55 %

```
(defrule verif_conflito_baixo_probabilidade
(or   ?id <- (pergunta (grau_medio ?x) (grau_conflito 0)
(conflito nil) (probabilidade ?y))
        ?id   <-   (pergunta_objetivo   (grau_medio   ?x)
(grau_conflito 0) (conflito nil) (probabilidade ?y))
        ?id   <-   (pergunta_requisito   (grau_medio   ?x)
(grau_conflito 0) (conflito nil) (probabilidade ?y))
        ?id   <-   (pergunta_participante   (grau_medio   ?x)
(grau_conflito 0) (conflito nil) (probabilidade ?y))
)
(and  (test (<= ?x 2.6))
      (test (>= ?y 55))
)
=>
```

```
(modify ?id (conflito 1) ))
```

Estado dos fatos:

Grupo 1

```
(pergunta (numero 9) (tipo 2) (hipotese 1) (grau_medio 2) (grau_conflito 0) (conflito 1) (utilidade nil) (probabilidade 50.0) (qtdrespostas 4) (qtdfavoraveis 2))
(causa (pergunta 9 ) (hipotese_pergunta 1 ) (causa 15 ) (hipotese_causa 1 ))
(causa (pergunta 9 ) (hipotese_pergunta 1 ) (causa 35 ) (hipotese_causa 1 ))
(causa (pergunta 9 ) (hipotese_pergunta 1 ) (causa 37 ) (hipotese_causa 0 ))
```

Grupo2

```
(pergunta_requisito (numero 4) (tipo 1) (hipotese 0) (identificador 7) (grau_medio 4) (grau_conflito 1) (conflito 1) (utilidade nil) (probabilidade 75.0))
(pergunta_requisito (numero 4) (tipo 1) (hipotese 0) (identificador 8) (grau_medio 3.4) (grau_conflito 1) (conflito 1) (utilidade nil) (probabilidade 57))
```

Podemos observar que o campo conflito foi atualizado, recebeu valor 1, ou seja, ocorreu conflito nestas perguntas.

3) As regras ativadas agora foram as que calculam a utilidade do conflito, chamadas calcula_utilidade_alto, que calcula a utilidade dos conflitos detectados com grau alto das respostas e calcula_utilidade_baixo, para os detectados com grau baixo das respostas. Apenas as perguntas onde os conflitos foram detectados tiveram a sua utilidade detectada, pois satisfizeram os pré-requisitos da regra.

- Regra que calcula a utilidade, para as perguntas que detectaram conflito com grau médio alto das respostas.

```
(defrule calcula_utilidade_alto
 (or    ?id <- (pergunta (conflito 1) (grau_conflito 1) (grau_medio ?w) (probabilidade ?x) (utilidade nil))
```

```
            ?id    <-    (pergunta_requisito    (conflito    1)
(grau_conflito 1) (grau_medio ?w) (probabilidade ?x) (utilidade
nil))
            ?id <- (pergunta_objetivo (conflito 1) (grau_conflito
1) (grau_medio ?w) (probabilidade ?x) (utilidade nil))
            ?id    <-    (pergunta_participante    (conflito    1)
(grau_conflito 1) (grau_medio ?w) (probabilidade ?x) (utilidade
nil))
       )
     =>
     (modify ?id (utilidade(/(*(- ?w 3) ?x)100)))
     )
```

- Regra que calcula a utilidade, para as perguntas que detectaram conflito com grau médio baixo das respostas.

```
     (defrule calcula_utilidade_baixo
      (or    ?id <- (pergunta (conflito 1) (grau_conflito 0)
(grau_medio ?w) (probabilidade ?x) (utilidade nil))
            ?id    <-    (pergunta_requisito    (conflito    1)
(grau_conflito 0) (grau_medio ?w) (probabilidade ?x) (utilidade
nil))
            ?id <- (pergunta_objetivo (conflito 1) (grau_conflito
0) (grau_medio ?w) (probabilidade ?x) (utilidade nil))
            ?id    <-    (pergunta_participante    (conflito    1)
(grau_conflito 0) (grau_medio ?w) (probabilidade ?x) (utilidade
nil))
       )
     =>
     (modify ?id (utilidade(/(*(- 3 ?w) ?x)100)))
     )
```

Estado dos fatos:

Grupo 1

```
     (pergunta (numero 9) (tipo 2) (hipotese 1) (grau_medio 2)
(grau_conflito 0) (conflito 1) (utilidade 0.5) (probabilidade
50.0))

     (causa (pergunta 9 ) (hipotese_pergunta 1 ) (causa 15 )
(hipotese_causa 1 ))
```

```
    (causa (pergunta 9 ) (hipotese_pergunta 1 ) (causa 35 )
(hipotese_causa 1 ))

    (causa (pergunta 9 ) (hipotese_pergunta 1 ) (causa 37 )
(hipotese_causa 0 ))
```

Grupo 2

```
    (pergunta_requisito (numero 4) (tipo 1) (hipotese 0)
(identificador 7) (grau_medio 4) (grau_conflito 1) (conflito 1)
(utilidade 0.75) (probabilidade 75.0))
    (pergunta_requisito (numero 4) (tipo 1) (hipotese 0)
(identificador 8) (grau_medio 3.4) (grau_conflito 1) (conflito 1)
(utilidade 0.23) (probabilidade 57))
```

4) A regra acionada agora é a regra denominada descobre_causas_perg_relativas, que calcula quais as causas, dentre o conjunto de causas de cada pergunta, se tornaram ativas. Isso é verificado observando se a pergunta contida na causa de uma pergunta que foi detectado o conflito, também detectou conflito.

```
    (defrule  descobre_causas_perg_relativas
        (or (pergunta (numero ?y) (conflito 1) (tipo 2) (hipotese
?x))
            (pergunta_objetivo (numero ?y) (conflito 1) (tipo 2)
(hipotese ?x))
            (pergunta_requisito (numero ?y) (conflito 1) (tipo
2) (hipotese ?x))
            (pergunta_participante (numero ?y) (conflito 1)
(tipo 2) (hipotese ?x))
         )
        ?id<-(causa (pergunta ?y)(hipotese_pergunta ?x) (causa
?c) (hipotese_causa ?h) (eh_causa nil))
        (or (pergunta (numero ?c) (hipotese ?h) (conflito 1))
             (pergunta_objetivo (numero ?c) (hipotese ?h)
(conflito 1))
             (pergunta_requisito (numero ?c) (hipotese ?h)
(conflito 1))
             (pergunta_participante (numero ?c) (hipotese ?h)
(conflito 1))
         )
```

```
=>
( modify ?id (eh_causa 1))
)
```

Estado dos fatos:

Grupo 1

```
(pergunta (numero 9) (tipo 2) (hipotese 1) (grau_medio 2) (grau_conflito 0) (conflito 1) (utilidade 0.5) (probabilidade 50.0))
(causa (pergunta 9) (hipotese_pergunta 1) (causa 35) (hipotese_causa 1) (eh_causa 1))
(causa (pergunta 9) (hipotese_pergunta 1) (causa 15) (hipotese_causa 1) (eh_causa nil))
(causa (pergunta 9) (hipotese_pergunta 1) (causa 37) (hipotese_causa 0) (eh_causa nil))
```

Grupo 2

```
(pergunta_requisito (numero 4) (tipo 1) (hipotese 0) (identificador 7) (grau_medio 4) (grau_conflito 1) (conflito 1) (utilidade 0.75) (probabilidade 75.0))
(pergunta_requisito (numero 4) (tipo 1) (hipotese 0) (identificador 8) (grau_medio 3.4) (grau_conflito 1) (conflito 1) (utilidade 0.23) (probabilidade 57))
```

Como observado, apenas a causa 35 da pergunta 9 se tornou ativa. Isso ocorreu porque não foram detectados conflitos nas perguntas 15, hipótese 1 e na pergunta 37.

5) As regras remove_perguntas_sem_conflito, remove_causas_nao_satisfeitas foram criadas para limpar o conjunto final de fatos. Elas são as últimas a regras ativadas.

- Limpa o conjunto de fatos, retirando as perguntas que não detectaram conflito.

```
(defrule  remove_perguntas_sem_conflito
    (or  ?id <- (pergunta (conflito nil) )
         ?id <- (pergunta_objetivo (conflito nil))
         ?id <- (pergunta_requisito (conflito nil))
```

```
            ?id <- (pergunta_participante (conflito nil))
)
=>
(retract ?id)
)
```

- Limpa o conjunto de fatos, retirando as causas que não se tornaram ativas.

```
 (defrule  remove_causas_nao_satisfeitas
  ?id <- (causa (eh_causa nil) )
=>
(retract ?id)
)
```

Estado final dos fatos:

Grupo 1

```
(pergunta (numero 9) (tipo 2) (hipotese 1) (grau_medio 2) (grau_conflito 0) (conflito 1) (utilidade 0.5) (probabilidade 50.0) (qtdrespostas 4) (qtdfavoraveis 2))
(causa (pergunta 9) (hipotese_pergunta 1) (causa 35) (hipotese_causa 1) (eh_causa 1))
```

Grupo 2

```
(pergunta_requisito (numero 4) (tipo 1) (hipotese 0) (identificador 7) (grau_medio 4) (grau_conflito 1) (conflito 1) (utilidade 0.75) (probabilidade 75.0))
(pergunta_requisito (numero 4) (tipo 1) (hipotese 0) (identificador 8) (grau_medio 3.4) (grau_conflito 1) (conflito1) (utilidade 0.23) (probabilidade 57))
```

O controle de disparo das regras no CLIPS é baseado em um mecanismo simples de prioridades, ou saliência de regras. O usuário pode especificar uma saliência para uma regra, e em caso de mais de uma regra estar ativa no momento, será escolhida para o disparo a regra com a maior prioridade.

O CLIPS contém a propriedade de refração, ou seja, uma regra não será disparada mais de uma vez para o mesmo fato, ou conjunto de fatos.

4.4.5. O Processo

O processo para fazer a ligação entre a aplicação desenvolvida em PHP e o sistema especialista desenvolvido em CLIPS é exemplificado através da figura abaixo:

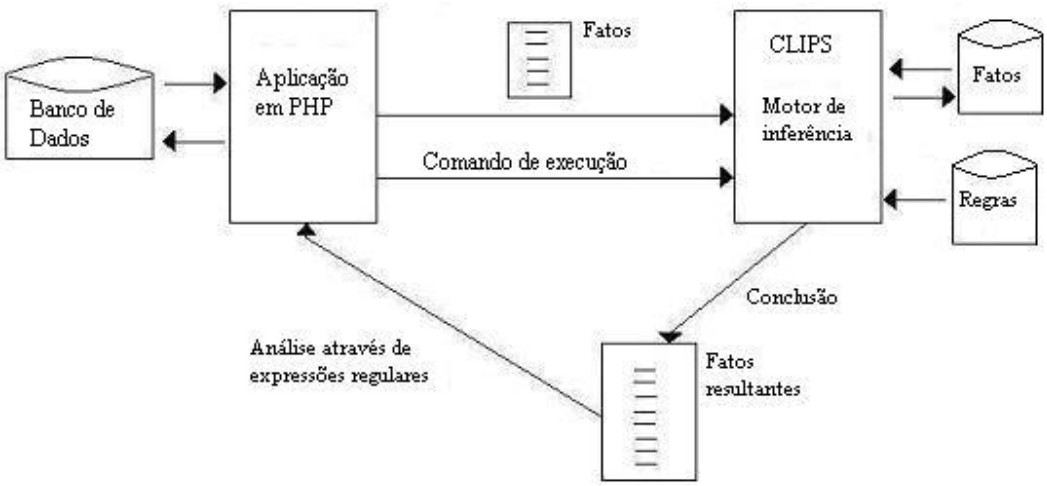

Figura 13 – O processo

A figura 13 mostra a troca de informações que é realizada entre o banco de dados, a aplicação e o sistema especialista. Este processo começa quando a aplicação busca no banco de dados as informações para construir o conjunto de fatos iniciais que será salvo em um arquivo texto (Fatos), e posteriormente lido pelo sistema especialista. A aplicação também aciona a execução de comandos para o funcionamento deste, que para isso utiliza o conjunto de regras e a base de fatos.

Após o sistema especialista trabalhar o conjunto de fatos enviado, ele grava os fatos resultantes em um outro arquivo texto (Fatos resultantes) que é lido pela aplicação através de expressões regulares e depois as informações necessárias salvas no banco de dados.

As informações da pergunta que não serão trabalhadas pelo sistema especialista, como a descrição da pergunta, do conflito e do gerenciamento não entram como informações na estrutura de fatos. Apenas as informações que serão modificadas pelo sistema especialista ou que serviram de informação no pré-requisito das regras, é que são passadas como fatos para o sistema. Isto vale também para a causa das perguntas absolutas, já que esta não é variável conforme

a reunião, como é o caso das causas das perguntas relativas. As informações que não variam conforme a reunião são manipuladas apenas na aplicação PHP.

4.4.6. Vantagens na utilização do sistema especialista

Com a utilização do sistema especialista (SE) estamos flexibilizando não só a manutenção das regras de identificação dos conflitos que nele estão armazenadas, pela maior facilidade de compreensão, mas também a base de conhecimento (questionário). Pois uma nova pergunta pode ser criada e também alterada sem que o SE perca sua funcionalidade, já que o conhecimento do especialista contido nele independe do conjunto de perguntas e seus relacionamentos.

Além das vantagens já apresentadas acima e na seção 4.4.1, nós gostaríamos de ressaltar a capacidade de expressar o conhecimento de uma forma facilmente compreensível entre elas. Como exemplo, podemos fazer a comparação de como algumas regras que foram escritas na linguagem do CLIPS ficariam escritas em PHP. O código a seguir é para identificar quais perguntas detectaram conflitos, e para as que detectarem, calcular sua utilidade:

```
    // Busca no BD as informações sobre as perguntas sem subdivisão
    $pergunta=mysql_query("select id_pergunta, grupo, tipo, grau_conflito, grau_medio, qtd, qtdfav from questao");
    while ($linha=mysql_fetch_array($pergunta))       //Enquanto existir pergunta
    {
    // Dando valores as variáveis
    if (($grau_conflito==1) and ($grau_medio<=2.5))
     {
          $probabilidade = 100*$favoravel/$qtd;
          $utilidade = (3-$grau_medio)*$probabilidade/100;
          $conflito = 1;
    }else
      {
         if (($grau_conflito==2) and ($grau_medio>=3.5))
         {
             $probabilidade = 100*$favoravel/$qtd;
```

```
            $utilidade = ($grau_medio-3)*$probabilidade/100;
            $conflito = 1;
          }
           else
            {
                 $probabilidade = 0;
                 $utilidade=0;
                 $conflito = 0;
            }

    //Atualiza informações no BD
    mysql_query("UPDATE questao SET conflito= '$conflito',
grau_medio='$grau_medio', probabilidade='$probabilidade',
utilidade = '$utilidade' WHERE id_reuniao='$reuniao'and
id_pergunta ='$id_pergunta' ") or print (mysql_error());
    }
```

A seguir analisaremos as perguntas com subdivisão. Primeiro as perguntas do grupo a, onde cada pergunta analisa todos os requisitos individualmente.

```
    // Primeiro seleciona todos os requisitos da reunião
    $requisitos=mysql_query("select id_requisito from requisito
where     id_reuniao='$reuniao'");
    while ($linha_req=mysql_fetch_array($requisitos))
    {
    // Seleciona as informações das perguntas que analisam o
requisito da vez
    $pergunta=mysql_query("select id_pergunta, grupo, tipo,
grau_conflito, grau_medio, qtd, qtdfav from q_requisito where
id_requisito = $linha_req[id_requisito]");
    while ($linha=mysql_fetch_array($pergunta))       //Enquanto
existir pergunta
    {
    if (($grau_conflito==1) and ($grau_medio<=2.5))
     {
          $probabilidade = 100*$favoravel/$qtd;
          $utilidade = (3-$grau_medio)*$probabilidade/100;
          $conflito = 1;
    }else
      {
```

```
        if (($grau_conflito==2) and ($grau_medio>=3.5))
        {
            $probabilidade = 100*$favoravel/$qtd;
            $utilidade = ($grau_medio-3)*$probabilidade/100;
            $conflito = 1;
         }
          else
           {
                 $probabilidade = 0;
      $utilidade=0;
                $conflito = 0;
           }

    //Atualiza informações no BD
    mysql_query("UPDATE q_requisito SET conflito= '$conflito',
grau_medio='$grau_medio',          probabilidade='$probabilidade',
utilidade    =    '$utilidade'    WHERE    id_reuniao='$reuniao'and
id_pergunta ='$id_pergunta' and id_requisito ='$linha_req[0]'") or
print (mysql_error());
    }
    }
```

O conjunto acima se repete para as perguntas com subdivisão de objetivos e participantes

Para fazer a análise acima em regras de produção utilizamos as regras grau_conflito_alto, grau_conflito_baixo, verif_conflito_alto, verif_conflito_baixo, calcula_utilidade_alto e calcula_utilidade_baixo, que podem ser visualizadas na seção 4.4.4.

No próximo código-fonte em PHP veremos como ficaria para detectar as causas das perguntas relativas que detectaram conflito.

```
    // Busca no BD as informações sobre as perguntas sem
subdivisão
    $pergunta=mysql_query("select id_pergunta, id_hipotese from
questão where conflito ='1' and tipo = '2'");
    while ($perg=mysql_fetch_array($pergunta))        //Enquanto
existir pergunta
    {
    // Busco as causas da pergunta
```

```
    $causas=mysql_query("select id_causa, id_hipotese_causa from
causa where id_pergunta = $perg[id_pergunta] and id_hipotese =
$perg[id_hipotese]" );
      while ($causa=mysql_fetch_array($causas))     // Para cada
causa
     {
    // Busco seu grupo
    $grupo_causa = mysql_query("select grupo from pergunta where
id_pergunta =              $causa[id_causa] and id_hipotese =
$causa[id_hipotese_causa]" );
    $grupo=mysql_fetch_array($grupo_causa)

    if ($grupo = 'a')
    {
       $verifica = mysql_query("select id_pergunta, id_hipotese
from q_requisito where conflito ='1' and id_pergunta =
$causa[id_causa] and id_hipotese = $causa[id_hipotese_causa]" );
    }else
     {
       if ($grupo = 'c')
       {
    $verifica = mysql_query("select id_pergunta, id_hipotese from
q_objetivo where conflito ='1' and id_pergunta = $causa[id_causa]
and id_hipotese = $causa[id_hipotese_causa]" );
        }
         else
         {
           if ($grupo = 'e')
           {
     $verifica = mysql_query("select id_pergunta, id_hipotese
from q_participante where       conflito ='1' and id_pergunta =
$causa[id_causa] and id_hipotese = $causa[id_hipotese_causa]" );
            }
           else
          {
                $verifica   =   mysql_query("select   id_pergunta,
id_hipotese from questao where     conflito ='1' and id_pergunta =
$causa[id_causa] and id_hipotese = $causa[id_hipotese_causa]" );
          }
        }
     }
```

```
$num_linhas=mysql_num_rows($verifica);

if ($num_linhas<>0)
{
mysql_query ("insert into causa_ativa (id_pergunta,
id_hipotese, id_causa, id_hipotese_causa) values
($perg[id_pergunta] , $perg[id_hipotese], $causa[id_causa],
$causa[id_hipotese_causa])" );
}
}
}
```

O código acima se repetirá para verificar as causas das perguntas relativas dos grupos com subdivisão em requisitos, objetivos e participantes.

Para realizar a análise feita acima em regras de produção utilizamos apenas a regra descobre_causas_perg_relativas, vide seção 4.4.4.

Verificamos então que expressar o conhecimento através de regras de produção é mais vantajoso, levando em conta a flexibilidade, a clareza do código e a manutenibilidade.