UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Vitor Oba

APLICATIVO PARA O GERENCIAMENTO DE REQUISITOS

Florianópolis
2004

VITOR OBA

APLICATIVO PARA O GERENCIAMENTO DE REQUISITOS

Iomani Engelmann Gomes

Florianópolis
2004

1 RESUMO

Por muitos anos, o foco de um bom produto era o próprio produto, e muitas informações da confecção do mesmo eram perdidas, e o processo que o desenvolvia não era avaliado. Atualmente, o produto com qualidade é conseqüência de um projeto bem feito e inspecionado.

Um projeto bem definido deve ter uma série de passos e um deles é o gerenciamento de requisitos. Com o gerenciamento de requisitos é possível gerenciar os requisitos de um projeto assim como identificar inconsistências entre os requisitos.

Este projeto voltou-se ao desenvolvimento de uma ferramenta para auxiliar as práticas de gerenciamento de requisitos baseada no modelo CMMI (Capability Maturity Model Integration), se restringindo ao nível 2 deste modelo. Uma ferramenta operacional e totalmente extensível para projetos futuros adicionarem integrações com ferramentas de padrão no mercado.

SUMÁRIO

1 RESUMO.....02
2 SUMÁRIO.....03
3 LISTA DE TABELAS.....05
4 LISTA DE FIGURAS.....05
5 ACRÔNIMOS E ABREVIATURAS 07
6 INTRODUÇÃO 08
6.1 Tema.....08
6.2 Justificativa 08
6.3 Objetivos 09
6.3.1 Objetivo geral 09
6.3.2 Objetivos específicos 09
7 CMMI (CAPABILITY MATURITY MODEL INTEGRATION) 10
7.1 Introdução 10
7.2 Selecionando um modelo CMMI 10
7.3 Visão geral da estrutura do CMMI 11
7.4 Componentes do modelo CMMI 16
7.5 Gerenciamento de requisitos 17
8 APLICATIVO 22
8.1 Análise 22
8.2 Projeto 25
8.2.1 Struts 25
8.2.2 Arquitetura 27
8.2.2.1 Banco de dados 27
8.2.2.2 Módulo Action 29
8.2.2.3 Módulo Form 30
8.2.2.4 Módulo Rules 31
8.2.2.5 Módulo Broker 32
8.2.2.6 JSP 33
8.2.2.7 Classes utilitárias 33
8.3 Implementação 34

8.4 Teste 37
9 CONCLUSÃO 46
10 REFERÊNCIAS BIBIOLGRÁFICAS 47
11 ANEXOS 48

3 LISTA DE TABELAS

TABELA 1: PRÁTICAS RELACIONADAS AOS OBJETIVOS DO GERENCIMANTO DE REQUISITOS.....18

4 LISTA DE FIGURAS

FIGURA 1: ESTRUTURA DA REPRESENTAÇÃO EM ESTÁGIOS DO MODELO CMMI.....12
FIGURA 2: ESTRUTURA DA REPRESENTAÇÃO CONTÍNUA DO MODELO CMMI...14
FIGURA 3: MÁQUINA DE ESTADOS DO STATUS DE UM REQUISITO COM ESTADO DE ANÁLISE COMO ESTADO INICIAL.....23
FIGURA 4: FLUXO DE CONTROLE DE UMA SOLICITAÇÃO.....26
FIGURA 5: ARQUITETURA BÁSICA.....27
FIGURA 6: MODELAGEM DO BANCO DE DADOS.....28
FIGURA 7: ARQUITETURA DO MÓDULO ACTION.....29
FIGURA 8: CLASSES DO MÓDULO FORM.....30
FIGURA 9: CLASSES DO MÓDULO RULES.....32
FIGURA 10: CLASSES UTILITÁRIAS.....34
FIGURA 11: VALIDAÇÃO DE USUÁRIO.....34
FIGURA 12: DESCRIÇÃO DO PROJETO PARA GERENTE DE PROJETO.....35
FIGURA 13: DESCRIÇÃO DO PROJETO PARA OUTROS PARTICIPANTES.....35
FIGURA 14: DESCRIÇÃO DE UM REQUISITO.....36
FIGURA 15: INSERINDO UM NOVO PROJETO.....37
FIGURA 16: PROJETO EM QUE O USUÁRIO PARTICIPA.....37
FIGURA 17: DESCRIÇÃO DO PROJETO.....38
FIGURA 18: INSERIR UM NOVO USUÁRIO AO PROJETO.....38
FIGURA 19: TRÊS ALTERNATIVAS DE INSERÇÃO DE REQUISITO.....39
FIGURA 20: INSERINDO UM NOVO REQUISITO PURO E SIMPLES.....39
FIGURA 21: DESCRIÇÃO DO PROJETO COM O REQUISITO INSERIDO.....39
FIGURA 22: INTERFACE PARA INSERIR UMA NOVA VERSÃO DE UM REQUISITO.....40
FIGURA 23: INTERFACE PARA IMPORTAR UM REQUISITO.....40
FIGURA 24: INFORMAÇÕES SOBRE O REQUISITO SELECIONADO.....41
FIGURA 25: INTERFACE PARA A INSERÇÃO DE UM REQUISITO.....41
FIGURA 26: INTERFACE PARA A INSERÇÃO DE TAREFAS.....42
FIGURA 27: DESCRIÇÃO DO REQUISITO COM A TAREFA INSERIDA.....42

FIGURA 28: INTERFACE PARA CADASTRO DE HORAS DE UMA ATIVIDADE.....42
FIGURA 29: INTERFACE MOSTRANDO AS HORAS CADASTRADAS.....43
FIGURA 30: MATRIZ DE RASTREABILIDADE.....43

5 ACRÔNIMOS E ABREVIATURAS

- CMM – Capability Maturity Model
- CMMI – Capability Maturity Model Integration
- SEI – Software Engineering Institute
- IPD-CMM – Integrated Product Development Capability Maturity Model
- MVC – Model View Controller
- UFSC – Universidade Federal de Santa Catarina
- JSP – Java Server Pages
- JDBC – Java Database Connectivity
- UML – Unified Modeling Language

6 INTRODUÇÃO

Este trabalho tem como objetivo desenvolver uma ferramenta para a engenharia de requisitos, mais especificamente para uma fração dela, que é o gerenciamento, que ocorre quando a especificação de requisitos já existe. Esta ferramenta cobre todos os objetivos específicos do gerenciamento de requisitos do CMMI nível 2. Inicialmente, será feita uma breve apresentação do CMMI (Capability Maturity Mode Integration) e do gerenciamento de requisitos deste processo de desenvolvimento de sistemas e software. Em seguida, apresenta-se o aplicativo, as conclusões e outras considerações.

6.1 Tema

O tema do projeto é engenharia de software, focada no gerenciamento de requisitos, cobrindo todos os objetivos específicos do gerenciamento de requisitos do CMMI nível 2.

6.2 Justificativa

Hoje em dia, os aplicativos que fazem o gerenciamento de requisitos exigem o preenchimento extensivo de *templates* para fazer o gerenciamento dos mesmos. Este entrave torna esta tarefa dispendiosa em tempo e por muitas vezes monótona. Gera uma burocratização que pode acarretar com o abandono do uso deste gerenciamento.

Além disso, dados estatísticos levantados mostram que até 70% do esforço do projeto são gastos com revisões e revalidação das necessidades iniciais dos clientes, principalmente por requisitos não atendidos ou mal executados.

Adicionado a todo este contexto, as boas práticas de engenharia de software na empresas brasileiras, de modo geral, não são bem executadas ou definidas. Muitas destas questões, pelo alto custo das ferramentas e também pela pouca flexibilidade que as mesmas apresentam. Com isto, ter um trabalho aberto para execução deste importante passo dentro de um processo de software, torna-se o principal motivador na execução deste trabalho.

6.3 Objetivos

6.3.1 Objetivo geral

Criação de uma ferramenta que permita o gerenciamento dos requisitos durante diferentes fases do processo de desenvolvimento de software.

6.3.2 Objetivos específicos

- Enfatizar e descrever os principais passos do gerenciamento de requisitos do CMMI nível 2;
- Definir o escopo de atuação que uma ferramenta é capaz de atuar neste processo;
- Apresentar uma proposta de solução para a implementação;
- Explanar a solução proposta e implementada;

7 CMMI (CAPABILITY MATURITY MODEL INTEGRATION)

7.1 Introdução

O CMMI foi desenvolvido no intuito de solucionar problemas decorrentes dos modelos CMM. Por exemplo, existiam diferenças entre disciplinas específicas de cada modelo. Tais diferenças dificultavam os esforços das empresas em executar melhorias. E o emprego de vários modelos resultava em custos adicionais, tais como a necessidade de treinamento. Diante desse cenário, o CMMI foi proposto fazendo a integração dos seguintes modelos[CSR2]:

- Capability Maturity Model for Software (SW-CMM) v2.0 draft C;
- Electronic Industries Alliance Interim Stardard (EIA/IS) 731;
- Integrated Product Development Capability Maturity Model (IPD-CMM) v0.98;

O propósito dos modelos do CMMI é propor uma orientação que permita melhorias de um processo de uma organização, não sendo, portanto, um processo em si ou uma descrição de um processo. O papel dela é fornecer mecanismos para que uma organização gerencie o desenvolvimento, a aquisição e manutenção dos serviços, construindo processos estáveis, capazes e maduros.

7.2 Selecionando um modelo CMMI

Há múltiplos modelos CMMI disponíveis, onde a escolha depende das necessidades da organização. Se o campo de conhecimento está voltado a sistemas, o modelo define a representação contínua. Se for voltada a disciplina de software, pode-se adotar a representação em estágio[TCC3]. Há também a necessidade de escolher qual o campo de atuação, ou simplesmente disciplina, que será incorporado ao modelo escolhido.

A seguir será relatada sucintamente as principais vantagens e possíveis desvantagens das representações contínua e em estágios. Depois será descrito o foco de cada disciplina.

As vantagens da representação contínua são:

- Possibilidade de escolha da melhoria a ser realizada, levando em consideração os objetivos do negócio;
- Possibilidade de comparação de resultados por área de processo;

As vantagens da representação em estágios são:

- Permite que as melhorias sejam executadas em fases, fazendo melhorias progressivas;
- Possibilita comparações através de níveis de maturidade;

As quatro disciplinas e foco de cada uma delas são:

- Engenharia de sistemas: cobre o desenvolvimento do sistema como um todo, podendo tal sistema ser um software ou não. Foca em transformar as necessidades dos clientes e suas expectativas em produto e suporte durante o ciclo de vida do produto;
- Engenharia de software: cobre o desenvolvimento do sistema de software. Seu foco consiste em aplicar abordagens sistemáticas, disciplinadas e quantificável para o desenvolvimento, operação e manutenção do software;
- Desenvolvimento de produtos integrados e processos: é uma abordagem sistemática que permite a colaboração de relevantes *stakeholders* durante o ciclo de vida do produto para satisfazer as necessidades dos clientes, suas expectativas, e requisitos;
- Contrato de fornecedores: a medida que projetos ficam mais complexo, a contratação de fornecedores torna-se interessante. Esta disciplina cobre o monitoramento dos fornecedores;

7.3 Visão geral da estrutura do CMMI

Nesta seção será apresentadas as estruturas das representações contínua e em estágios do CMMI.

Representação em estágios

A estrutura do modelo CCMI em estágios é mostrada a seguir na figura 1.

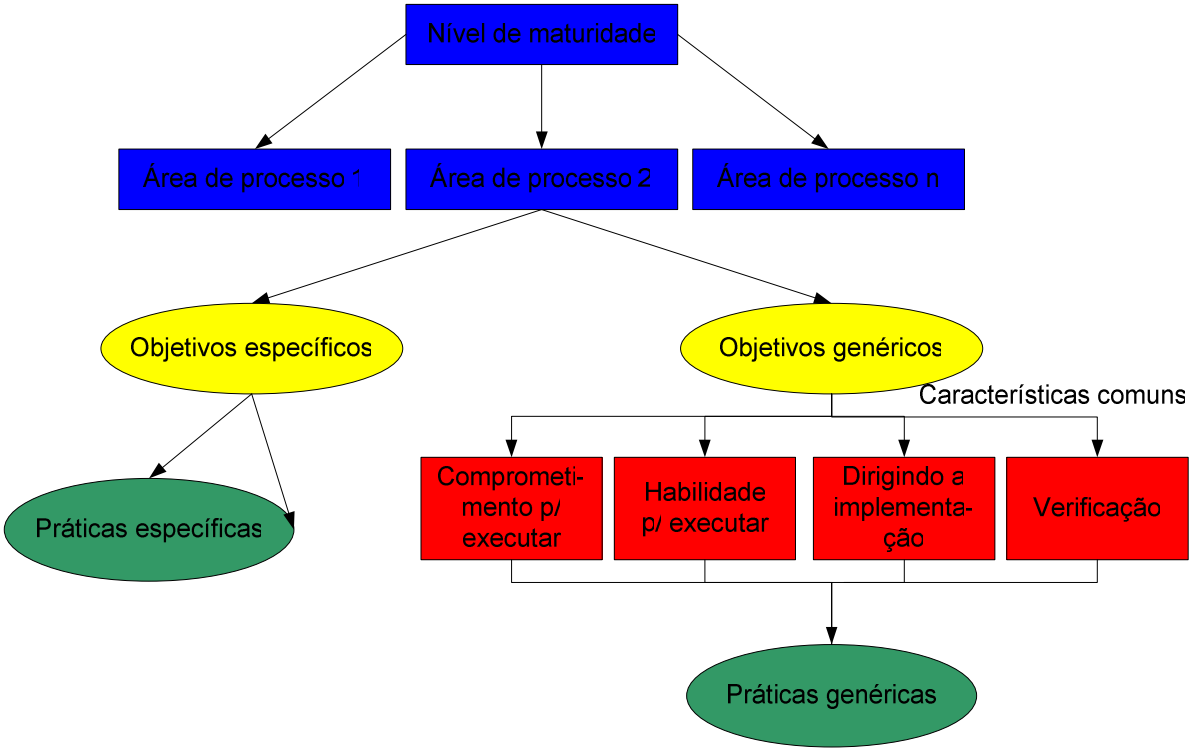


Figura 1: Estrutura da representação em estágios do modelo CMMI

Pode-se observar na figura 1 que cada nível de maturidade é constituído de áreas de processo, onde as áreas de processos possuem objetivos específicos e genéricos. Para cada objetivo especifico ou genérica existe práticas associado a elas. Características comuns organizam as práticas genéricas.

Nível de maturidade

Um nível de maturidade é um platô evolucionário bem definido de um processo de melhoramento[CSR2]. Há cinco níveis de maturidade na representação em estágios do modelo CMMI enumerados de 1 a 5:

1. Inicial;
2. Gerenciado;
3. Definido;
4. Quantitativamente gerenciado;
5. Otimizado;

A seguir serão descritas as características de cada nível de maturidade.

Nível de maturidade 1: inicial

No primeiro nível de maturidade, os processos são caóticos e o ambiente nas organizações não são estáveis. Os processos são abandonados em tempos de crise. Os sucessos dependem da competência dos funcionários e existe a dificuldade de se repetir sucessos do passado.

Nível de maturidade 2: gerenciado

No nível de maturidade 2 ocorre o gerenciamento dos requisitos e processos são planejados, executados, medidos e controlados. Práticas são executadas mesmo em tempos de stress. Compromissos são estabelecidos entre *stakeholders* e revistos se necessário. Produtos e serviços satisfazem os requisitos, padrões e objetivos definidos pelos *stakeholders.*

Nível de maturidade 3: definido

No nível de maturidade 3, processos estão bem caracterizados e entendidos, e definidos em padrões, procedimentos, ferramentas e métodos. A base no nível de maturidade 3 - conjunto de processos padrões, são estabelecidos e melhorados o tempo todo.

Nesse nível de maturidade os processos são mais detalhados e gerenciados mais pró-ativamente. São justamente essas as diferenças entre os níveis de maturidade 2 e 3.

Nível de maturidade 4: quantitativamente gerenciado

A performance é controlada fazendo o uso de estatísticas e métodos quantitativos e é qualitativamente previsível. As medidas analisadas e coletadas são utilizadas para tomadas de decisões futuras.

Nível de maturidade 5: otimizado

O nível de maturidade 5 foca em melhorias continuas na performance do processo. Melhorias dos processos é inerentemente papel de todos, resultando em um ciclo de

melhoria contínua. Melhorias são selecionadas levando em consideração os objetivos das melhorias da organização versus o custo e o impacto para a organização.

Representação contínua

A estrutura do modelo CCMI contínua é mostrado a seguir na figura 2.

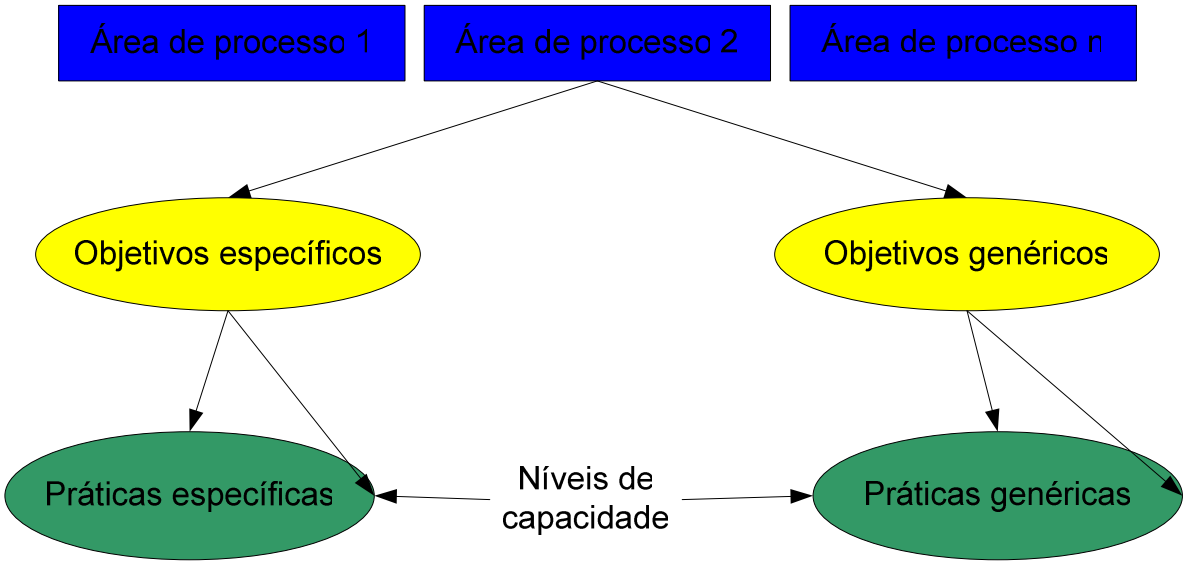


Figura 2: Estrutura da representação contínua do modelo CMMI

Observa-se na figura 2 que a representação contínua é agrupado em áreas de processo. Cada área de processo possui objetivos específicos que mapeiam para práticas específicas e objetivos genéricos que mapeiam para práticas genéricas. E cada prática especifica e genérica corresponde a um nível de capacidade.

Um nível de capacidade é constituído de práticas específicas e genéricas relacionada a uma área de processo que permite melhorias no processo da organização associada com a área de processo.

Existe 6 níveis de capacidade enumeradas de 0 a 5:

0. Incompleta
1. Executado
2. Gerenciado
3. Definido
4. Quantitativamente gerenciado
5. Otimizado

A seguir será descrita as características de cada nível de capacidade

Nível de capacidade 0: incompleto

Não é executado ou parcialmente executado.

Nível de capacidade 1: executado

Satisfaz os objetivos específicos da área de processo.

Nível de capacidade 2: gerenciado

É planejado e executado de acordo com políticas, envolvimento de *stakeholders*, treinamento de pessoas, atribuição de responsabilidade, adoção de ações corretivas*;* é monitorado, controlado, e revisto.

Nível de capacidade 3: definido

Processos padrões e o suporte para o uso imediato ou futuro desses processos são estabelecidos e melhorados o tempo todo.

Nível de capacidade 4: quantitativamente gerenciado

Se estabelece e mantém objetivos quantitativos para qualidade e performance do processo.

Nível de capacidade 5: otimizado

Foca em melhorias incrementais e inovadores e tais melhorias são constantemente gerenciados.

7.4 Componentes do modelo CMMI

Os principais componentes do modelo CMMI serão descritos a seguir.

Área de processo

Composto por práticas relacionadas em uma área e que executadas coletivamente satisfazem um conjunto de objetivos possibilitando melhorias nessa área.

Objetivos específicos

Aplicados a área de processo e que devem ser implementados para satisfazer a área de processo.

Prática específica

Atividade considerada importante para que se realize o objetivo especifico associado.

Características comuns

Existem quatro características comuns relacionadas às práticas genéricas de cada área de processo. São elas:

1. **Comprometimento para executar:** relacionados à criação de políticas;
2. **Habilidade para executar:** garante que o projeto e/ou organização tenha os recursos necessários;
3. **Dirigindo a implementação:** gerenciar a performance do processo e a integridade dos seus produtos e envolver *stakeholders* relevantes;
4. **Verificando a implementação:** verificar conformidade das descrições do processo, procedimentos e padrões;

Objetivos genéricos

O mesmo objetivo genérico aparece em varias áreas de processo. Na representação em estágios, cada área de processo tem somente um objetivo genérico. São usados para determinar se uma área de processo é satisfeita.

Práticas genéricas

Garante que processos associados à área de processo serão efetivo, repetível e duradouro.

A seguir será descrita uma das áreas de processo do nível de maturidade 2 que é o gerenciamento de requisitos. A escolha dessa área de processo deve-se ao escopo do aplicativo proposto.

7.5 Gerenciamento de requisitos

O propósito do gerenciamento de requisitos é gerenciar os requisitos dos projetos e garantir que tais requisitos atinjam as expectativas do cliente.

Essa área de processo tem os seguintes objetivos específicos e genéricos:

- **Objetivo específico 1 – Gerenciar requisitos:** requisitos são gerenciados e inconsistências são identificados[CSR2].
- **Objetivo genérico 2 – Institucionalizar um processo gerenciado:** o processo é institucionalizado como um processo gerenciado[CSR2].
- **Objetivo genérico 3 – Institucionalizar um processo definido:** o processo é institucionalizado como um processo definido[CSR2].

As práticas relacionadas aos objetivos citados anteriormente são mostradas abaixo, na tabela 1.

Gerenciamento de requisitos	
OE 1 - Gerenciar requisitos	PE 1.1 Obter um entendimento dos requistos PE 1.2 Obter o comprometimento para os requisitos PE 1.3 Gerenciar mudanças nos requisitos PE 1.4 Manter a rastreabilidade bi-direcional dos requisitos PE 1.5 Identificar inconsistências entre produtos de trabalho e os requisitos
OG 2 - Institucionalizar um processo gerenciado	PG 2.1 Estabelecer uma política organizacional PG 2.2 Planejar o processo PG 2.3 Fornecer recursos PG 2.4 Atribuir responsabilidades PG 2.5 Treinar pessoas PG 2.6 Gerenciar a configuração PG 2.7 Identificar e envolver *stakeholders* relevantes PG 2.8 Monitorar e controlar o processo PG 2.9 Objetivamente avaliar a aderência PG 2.10 Revisar o status com a mais alta gerência
OG 3 - Institucionalizar um processo definido	PG 3.1 Estabelecer um processo definido PG 3.2 Coletar informações de melhoria

Tabela 1: Práticas relacionadas aos objetivos do gerenciamento de requisitos

As práticas do OE 1 são as práticas relacionadas a este projeto. A seguir serão descritas tais práticas.

PE 1.1 Obter um entendimento dos requisitos

O objetivo desta prática é criar critérios que permita o estabelecimento de um canal de comunicação apropriado para garantir que um entendimento compatível e compartilhado seja alcançado, resultando em um conjunto de requisitos que estejam em concordância.

Produtos de trabalho típicos

1. Listas de critérios para distinguir fornecedores de requisitos apropriados[CSR2].
2. Critério para avaliação e aceitação dos requisitos[CSR2].
3. Resultados de análise em contraste aos critérios[CSR2].
4. Conjunto de requisitos de comum acordo[CSR2].

Sub-práticas

1. Estabelecer critérios para distinguir fornecedores de requisitos apropriados[CSR2].
2. Estabelecer critérios objetivos para a aceitação dos requisitos[CSR2].

3. Analisar os requisitos para garantir que o critério estabelecido sejam encontrado[CSR2].
4. Alcançar um entendimento dos requisitos com os fornecedores dos requisitos assim os participantes do projeto podem se comprometer[CSR2].

PE 1.2 Obter o comprometimento para os requisitos

Esta prática lida com a concordância e o comprometimento dos envolvidos encarregados de realizar as atividades que possibilitam a implementação dos requisitos.

Produtos de trabalho típicos

1. Avaliação do impacto dos requisitos[CSR2].
2. Comprometimentos documentados para os requisitos e mudanças nos requisitos[CSR2].

Sub-práticas

1. Avaliar o impacto dos requisitos em comprometimentos já existentes[CSR2].
2. Negociar e registrar comprometimentos[CSR2].

PE 1.3 Gerenciar mudanças nos requisitos

Objetivo desta prática é gerenciar a adição e mudanças nos requisitos de uma forma eficiente e efetivo. Há a necessidade de se conhecer código fonte de cada requisito e que análises racionais para qualquer mudança seja documentado para que se possa fazer uma análise do impacto das mudanças.

Produtos de trabalho típicos

1. Status dos requisitos[CSR2].
2. Base de dados dos requisitos[CSR2].
3. Base de dados das decisões dos requisitos[CSR2].

Sub-práticas

1. Capturar todos os requisitos e mudanças nos requisitos que são dados pelo ou gerados pelo projeto[CSR2].
2. Manter o histórico das mudanças dos requisitos com a análise racional para as mudanças[CSR2].

3. Avaliar o impacto das mudanças dos requisitos do ponto de vista dos *stakeholders* relevantes[CSR2].
4. Fazer os requisitos e dados da mudança disponíveis ao projeto[CSR2].

PE 1.4 Manter a rastreabilidade bi-direcional dos requisitos

O objetivo desta prática é manter uma rastreabilidade dos requisitos do código fonte até o nível mais baixo de especificação e do nível mais baixo de especificação até o código fonte. A rastreabilidade ajuda a verificar se todos os requisitos foram endereçados, cobre relacionamentos entre outras entidades tais como mudanças nos documentos do projeto, plano de testes.

Produtos de trabalho típicos

1. Matriz de rastreabilidade de requisitos[CSR2].
2. Sistema de acompanhamento de requisitos[CSR2].

Sub-práticas

1. Manter a rastreabilidade de requisitos para garantir que o código fonte derivado dos requisitos sejam documentados[CSR2].
2. Manter a rastreabilidade dos requisitos de um requisito para seu requisito derivado assim como para a alocação de funções, objetos, pessoas, processos e produto de trabalho[CSR2].
3. Manter a rastreabilidade horizontal de função para função e através das interfaces[TCC3].
4. Gerar a matriz de rastreabilidade de requisitos[CSR2].

1.5 Identificar inconsistências entre produtos de trabalho e os requisitos

O objetivo desta prática é encontrar inconsistências entre os requisitos e os planos de projeto e os produtos de trabalho e tomar ações corretivas.

Produtos de trabalho típicos

1. Documentação de inconsistências incluindo código fonte, condições e análise racional[CSR2].
2. Ações corretivas[CSR2].

Sub-práticas

1. Revisar os planos de projeto, atividades, e produtos de trabalho para consistências com os requisitos e as mudanças feitos a eles[CSR2].
2. Identificar a fonte da inconsistência e a análise racional[CSR2].
3. Identificar mudanças que precisam ser feitas para os planos e os produtos de trabalho[CSR2].
4. Iniciar ações corretivas[CSR2].

Os objetivos específicos aqui apresentados definem o escopo da ferramenta. Além disso, a ferramenta proposta semi-automatiza as práticas relacionadas a estes objetivos. Por exemplo, o usuário não precisa preencher longos documentos para o aplicativo fornecer a matriz de rastreabilidade.

Nas seções a seguir, descreve-se o aplicativo que cobre os objetivos específicos aqui apresentados.

8 APLICATIVO

O aplicativo proposto é uma alternativa aos aplicativos atuais em que se exige o preenchimento extensivo de *templates*, tornando a tarefa do gerenciamento de requisitos monótona e dispendiosa em tempo. Como já dito anteriormente, o escopo do aplicativo está restrito ao gerenciamento de requisitos do CMMI nível 2. Mais especificamente, o aplicativo se restringe aos objetivos específicos do gerenciamento de requisitos do CMMI nível 2. Nos tópicos seguintes serão descritas as etapas de desenvolvimento do aplicativo.

8.1 Análise

A ferramenta tem como objetivo permitir que um gerente de projeto gerencie os requisitos de um projeto.

O aplicativo foi desenvolvido voltado para a *web* para permitir que todos os *stakeholders* (indivíduos que de alguma forma podem influenciar um projeto) de um projeto tenham acesso ao mesmo e na hora que assim o desejar.

O acesso ao sistema é feito através de uma validação de usuário e senha. O administrador do sistema tem a possibilidade de incluir novos usuários e de definir uma hierarquia de usuários. Para cada hierarquia, um gerente de projeto define o que o usuário pode fazer no sistema, como por exemplo, ele pode definir que o usuário tem a permissão de incluir requisitos em um projeto, alterar o status do requisito, visualizar os relatórios, entre outros.

Um usuário tem um papel dentro do projeto, podendo ter diferentes papéis em diferentes projetos. Foram definidos quatro papéis para o aplicativo: gerente de projeto, desenvolvedor, testador e cliente. Vale ressaltar que o aplicativo tem suporte a qualquer papel que o administrador defina posteriormente.

O aplicativo tem um módulo que permite o cadastramento de projetos. No momento do cadastro, deve-se estabelecer quem serão os gerentes de projeto e quais serão as fases de desenvolvimento.

Todo projeto tem sua versão e a versão só pode ser fechada caso todos os seus requisitos estejam fechados.

Cada projeto é composto de um ou mais requisitos. Para a inclusão de um requisito em um projeto, o gerente de projeto tem três alternativas: importar um requisito de algum outro projeto,

criar uma nova versão de um requisito já existente dentro do projeto e que sua versão esteja finalizada ou ser um novo requisito puro e simples.

Se o gerente de projeto optar por um novo requisito, ele tem a opção de colocar os requisitos de dependência. Se optar por importar um requisito, todos os requisitos de que ele depende será importado automaticamente. Caso seja escolhido abrir uma nova versão de um requisito, todos os requisitos que dependiam do requisito que foi aberto uma nova versão, tem uma nova versão automaticamente aberta. Esta *feature* é para obrigar o gerente de requisitos revisar requisito por requisito, analisando o impacto da nova versão aberta sobre os requisitos que dependiam dele.

Cada requisito tem uma versão, um status, um grau de dificuldade de implementação do requisito dentro do projeto, uma estimativa de horas para implementá-lo e um comentário desta estimativa.

O gerente de projeto tem a opção de fechar a versão de um requisito. Se o fizer, ninguém mais altera aquele requisito naquela versão.

O aplicativo faz o controle do status do requisito, ou seja, dado um status inicial, o aplicativo exibe as possibilidades seguintes. Para tal, o aplicativo segue a máquina de estados mostrada na figura abaixo:

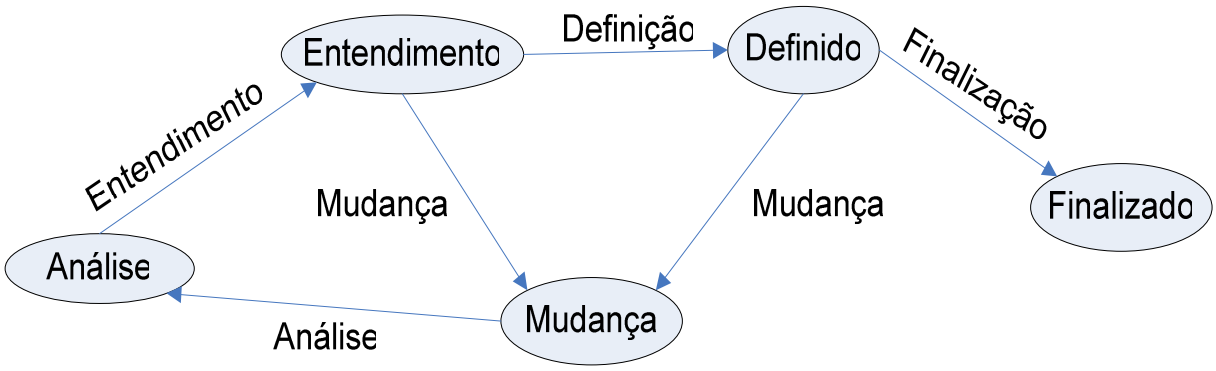


Figura 3: Máquina de estados do status de um requisito com estado de análise como estado inicial

Nota-se pela máquina de estados que tudo começa pelo estado análise. Deste estado, pode-se transitar para o estado entendimento. Do entendimento, pode-se transitar para os estados definido ou mudança. De mudança para o estado análise. De definido para os estados mudança e finalizado. O estado finalizado não transita para nenhum outro estado. Para voltar ao estado análise somente criando uma nova versão do requisito.

Tarefas são atribuídas aos requisitos e tais tarefas são atividades inseridas pelo gerente de projeto para implementar o requisito. Uma tarefa é atribuída automaticamente ao gerente de projeto, podendo ele atribuí-la a mais alguém que participe do projeto. A tarefa é atribuída automaticamente

para permitir que o gerente de projeto visualize, por exemplo, o histórico de horas cadastradas daquela tarefa, visto que elas só são visíveis para quem foi atribuída.

Cada vez que alguém alterar um requisito ou fazer alguma atividade para implementá-lo, o usuário deve informar qual fase do desenvolvimento do projeto pertence a atividade que ele deseja cadastrar, quantas horas foram gastas e selecionar qual artefato foi realizado. Os artefatos que o usuário pode selecionar são: diagramas de casos de uso, de classe, de colaboração, de seqüência, de estado, de atividade, de componente, de *deployment* e código fonte.

A ferramenta permite também a geração de uma matriz de rastreabilidade dos requisitos mapeando bidirecionalmente os requisitos do projeto. Para a composição da matriz, a ferramenta leva em consideração as últimas versões dos requisitos que estão dentro do projeto. Esta matriz irá responde às seguintes perguntas[TCC3]:

- Quais os impactos de uma mudança de um determinado requisito[TCC3]?
- Onde um requisito foi mapeado e/ ou implementado[TCC3]?
- Todos os requisitos foram cobertos e/ou implementados[TCC3]?
- O requisito é mesmo necessário[TCC3]?
- Quais os testes utilizados na verificação de um requisito[TCC3]?

Todas as informações relativas ao uso da ferramenta, tais como a inserção de um novo projeto, de um requisito a um projeto, de tarefas para implementar o requisito, são armazenadas em um banco de dados.

Nesta seção foi descrita as funcionalidades do aplicativo para cobrir os objetivos específicos do CMMI nível 2, sem exibir as telas de implementação. A ferramenta com suas telas de implementação serão mostradas na seção 8.3 onde é discutido a implementação do aplicativo e na seção 8.4 onde é feito um estudo de caso.

8.2 Projeto

Nesta seção seão descritas as atividades relacionadas à fase de projeto do desenvolvimento do aplicativo. Antes da discussão da arquitetura, é feita uma breve explanação do *framework* Struts, visto que esta foi utilizada extensamente para que a ferramenta seguisse o padrão MVC.

8.2.1 Struts

O Struts é um framework de software livre utilizado por desenvolvedores de componentes web em Java.

Sua arquitetura é extensível e adaptável[JMA01], dando suporte à diferentes camadas de apresentações, tais como o JSP – utilizado no aplicativo proposto, e também à diferentes camadas de persistências, tais como o JDBC – também utilizado no aplicativo proposto.

Nos padrões de projeto definidos para a linguagem de programação Java pela Sun, existem dois tipos de MVC (Model-View-Controller): o *Model 1* e o *Model 2*.

No *Model* 1 existem apenas dois componentes (JSP e *Javabeans*) para desempenhar três papéis, sendo que o papel do controlador está totalmente disperso entre os dois componentes. Já no *Model* 2 existem três componentes(JSP, *Javabeans* e Controlador), cada um desempenhando seu papel.

O *framework* implementa o *Model* 2, fornecendo ela, o papel do controlador. Com isso é garantido um controle centralizado e se usado corretamente é garantido também que ambos desenvolvedores e *designers* não tenham que editar, por exemplo, páginas JSP.

O controlador é constituído de três elementos principais[JMA1]: ActionServlet(servlet controlador), RequestProcessor e os Actions(classes de ação), sendo as duas primeiras implementadas pelo Struts.

O controlador interage com a camada de negócios através de ações e tais ações são definidas estendendo a classe abstrata Action. O Struts também fornece o ActionForm. Estendendo tal classe, ele representará, dentro do *framework*, um formulário HTML, sendo este preenchido automaticamente pelo Struts.

É exigido que uma aplicação tenha um arquivo de configuração chamado *struts-config.xml*. Neste arquivo, o desenvolvedor define qual o Action que está associado à requisição do usuário.

Toda solicitação do usuário de uma aplicação que faça uso do Struts segue o seguinte fluxo de controle:

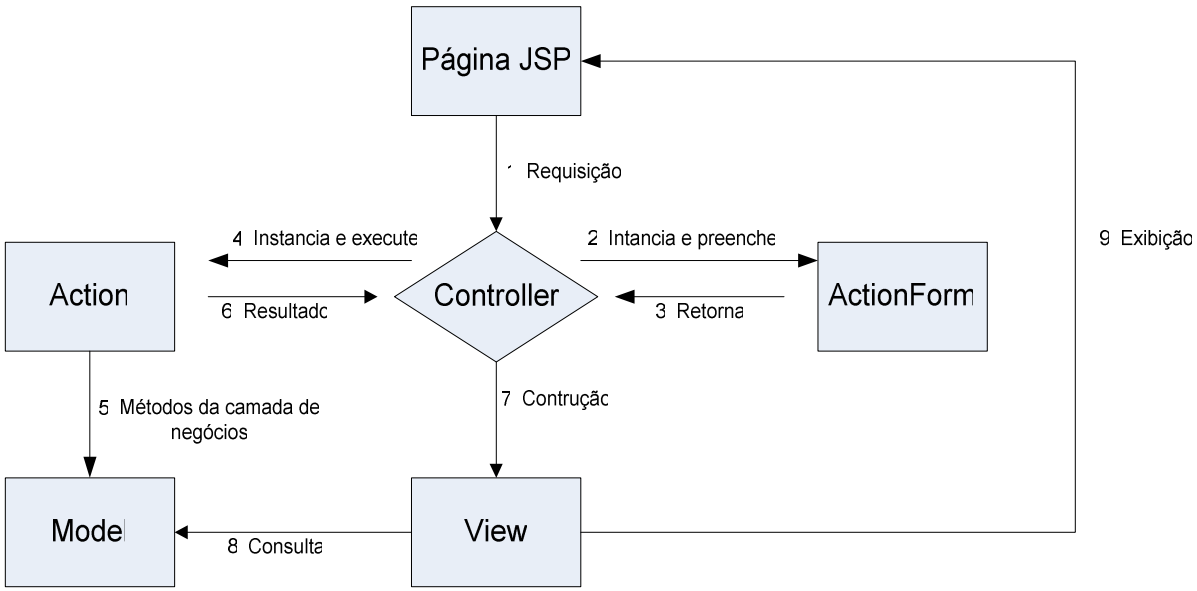


Figura 4: Fluxo de controle de uma solicitação

A solicitação de um usuário segue nove passos de acordo com a figura acima. As descrições de cada passo são:

1. Dada uma requisição, o servlet verifica no arquivo de configuração qual Action está associado a essa requisição e se há a necessidade de um ActionForm;
2. Se é necessário um ActionForm, o *framework* instancia um e preenche com os atributos vindos da requisição;
3. O ActionForm é retornado. Se ocorrer um erro, o passo 7 é executado; senão, o processamento segue normalmente;
4. O Action associado à requisição é instanciado e seu método *execute* é chamado;
5. A instância do Action chama os métodos da camada de negócios necessários;
6. No final do método *execute*, o desenvolvedor diz se deve ser executada uma outra ação ou a resposta deve ser exibida;
7. Uma página é construída com erro ou com resposta, dependendo do processamento da requisição;
8. A página de resposta consulta algum objeto de resposta para exibir uma página apropriada;
9. A página de resposta de exibida ao usuário;

8.2.2 Arquitetura

Durante a fase de projeto, todos os diagramas feitos para a modelagem do sistema segue a metodologia UML [CRA 02].

A ferramenta utilizou o Struts para que seguisse o padrão MVC, onde o Struts fez o papel do *controller*. A arquitetura do aplicativo possui cinco módulos básicos: JSP, Action, Rules, Broker e o Form, como mostra a figura abaixo.

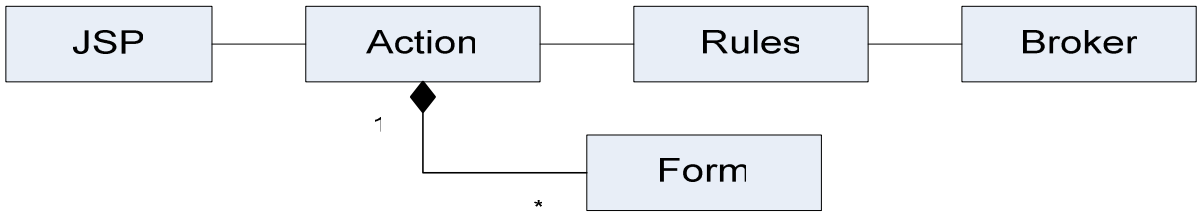


Figura 5: Arquitetura básica

Esta arquitetura básica possui algumas classes utilitárias que foram criadas para evitar a replicação de código. Uma descrição mais detalhada de cada módulo da arquitetura, das classes utilitárias, assim como a modelagem do banco de dados será feita nas seções seguintes.

8.2.2.1 Banco de dados

O banco de dados foi modelado de forma que o aplicativo proposto fosse o mais extensível possível. O administrador do sistema pode inserir no banco de dados, por exemplo, quantos artefatos ele desejar. A ferramenta dará suporte a todas elas.

A modelagem do banco resultou num total de 13 tabelas. A figura 6 mostra as tabelas e seus atributos.

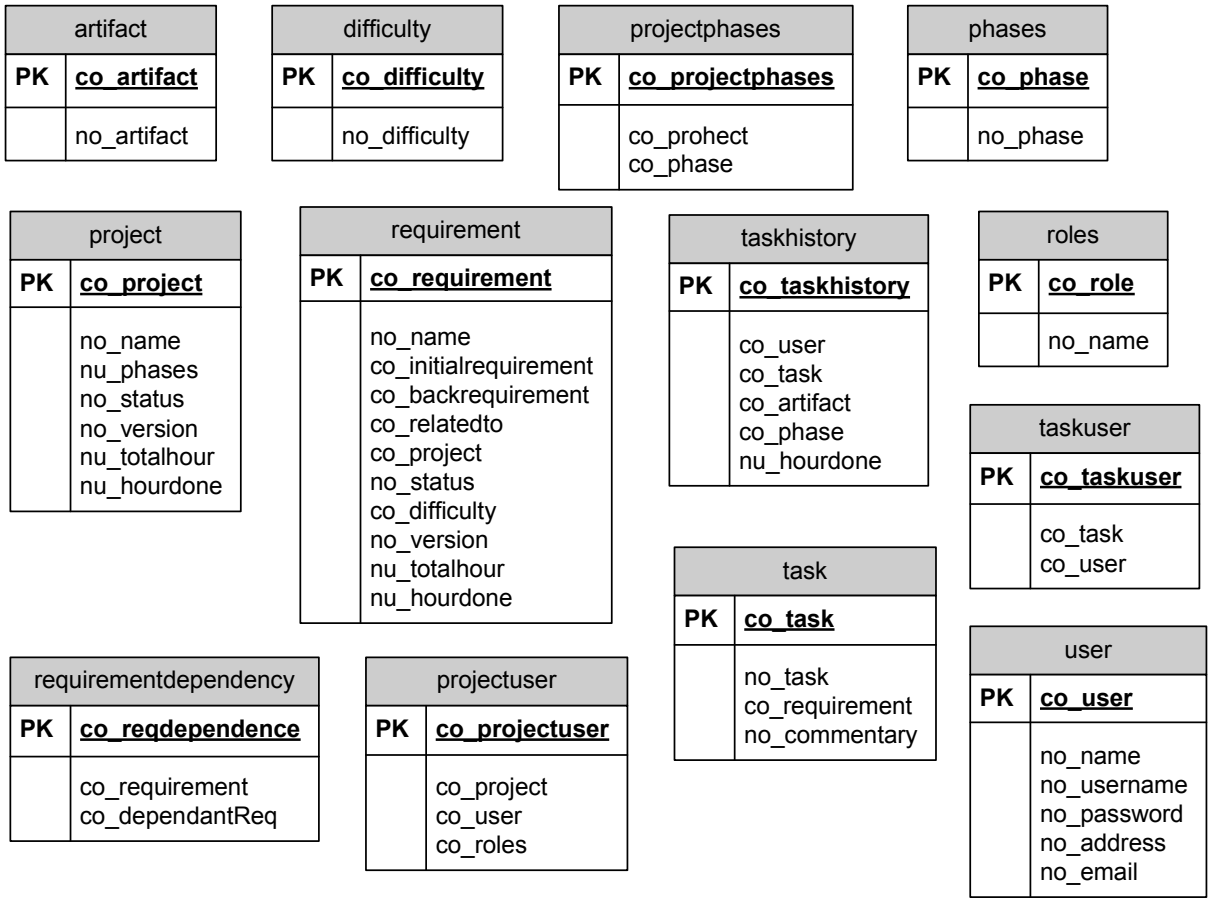


Figura 6: Modelagem do banco de dados

Segue abaixo, uma breve descrição de cada uma das tabelas:

- artifact: representa os artefatos que o aplicativo mostrará para o usuário, por exemplo, na hora de cadastrar uma atividade feita para cumprir uma tarefa;
- difficulty: representa os níveis de dificuldade de um requisito;
- phases: representa as fases que um projeto poderá ter;
- project: armazena os projetos inseridos pelo administrador do sistema;
- projectphases: faz o mapeamento entre o projeto e suas fases;
- projectuser: faz o mapeamento entre o projeto e seus *stakeholders*;
- requirement: armazena os requisitos de um projeto;
- requirementdependency: guarda as dependências de um determinado requisito;
- roles: representa os papéis que um usuário pode ter em um projeto;
- task: armazena as tarefas que são criadas para implementar um requisito;
- taskhistory: guarda as atividades feitas dentro de uma tarefa;
- taskuser: guarda os responsáveis por cumprir uma tarefa;
- user: representa todos os usuários que farão uso do sistema;

8.2.2.2 Módulo Action

O módulo Action faz a ponte entre a camada de apresentação e a camada de negócio. Todas as classes que pertençam a este módulo necessariamente estendem a classe abstrata Action, que é fornecida pelo Struts. E todas elas implementam o método *execute* herdado da classe Action, possuindo somente este método e nenhuma delas possuem atributos.

Para cada requisição de serviço de um usuário existe um Action associado. Como dito anteriormente, o *framework* Struts conhece o Action que está associado à requisição através do arquivo de configuração do aplicativo que faz uso do Struts.

Na figura abaixo é mostrada a arquitetura deste módulo.

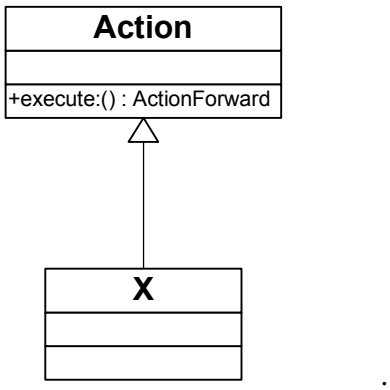


Figura 7: Arquitetura do módulo Action

Na arquiterura mostrada na figura acima, as subclasses de Action estão simbolizadas pela classe X. Esta omissão se deve à grande quantidade de classes que fazem parte deste módulo, onde a inclusão destes tornaria o diagrama poluído.

As subclasses do módulo Action são: AddDependencyAction, AddTaskAction, ChangeStatusAction, GenerateMatrixAction, ImportRequirementAction, InsertDependencyAction, InsertNewRequirementAction, InsertUserAtProjectAction, LoginAction, NewRequirementAction, NewVersionRequirementAction, ProjectsAction, RemoveDependencyAction, RequirementSelectedAction, RequirementViewHistoryAction, SubmitDependencyAction, SubmitImportRequirementAction, SubmitNewVersionRequirementAction, SubmitTaskAction, SubmitTaskHistoryAction, SubmitUserAtProjectAction, TaskSelectedAction, TaskViewHistoryAction.

8.2.2.3 Módulo Form

O módulo Form possui somente classes *Javabeans*. Ou seja, classes que tem somente propriedades e métodos para acessar ou modificar essas propriedades. Instâncias dessas classes só contêm dados, não podendo executar nenhuma ação. Todas essas classes estendem a classe ActionForm, que é também fornecido pelo *framework* Struts.

As instâncias das classes desse módulo representam tuplas de uma tabela do banco de dados, podendo ser uma tupla já existente no banco ou uma tupla que ainda vai ser inserida. Se uma requisição necessitar de uma instância de alguma classe do módulo Form, o *Struts* cria uma e automaticamente preenche os atributos.

A seguir é mostrado uma figura contendo as classes desse módulo.

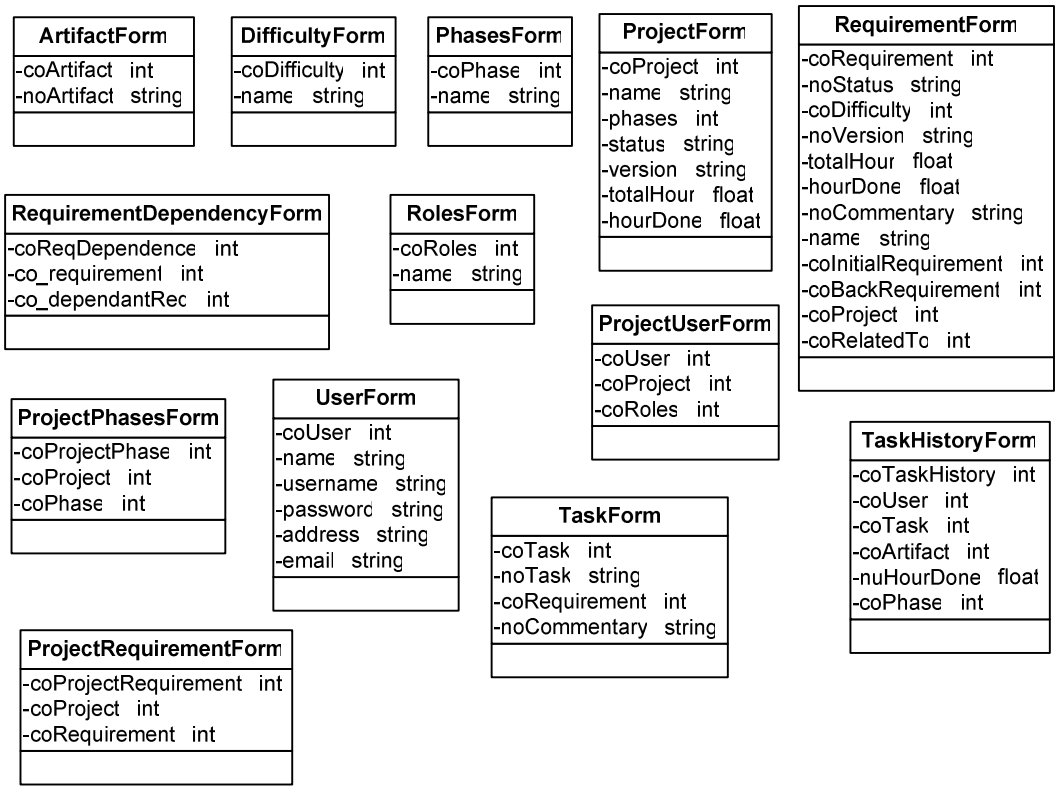


Figura 8: Classes do módulo Form

Nota-se pela figura acima que os atributos das classes coincidem com os atributos das tabelas do banco de dados. Isto é esperado visto que as classes representam as tabelas do banco e que as instâncias dessas classes representam tuplas dessas tabelas.

Vale ressaltar que as classes da figura acima estendem a classe ActionForm fornecido pelo Struts, não mostradas para não poluir a figura devido a grande quantidade de classes.

8.2.2.4 Módulo Rules

O módulo Rules contém as regras de persistência. Ele não persiste os dados propriamente dito, mas tem o conhecimento e fornece o que o módulo Broker precisa para, por exemplo, inserir algo no banco de dados.

Sem a existência deste módulo, ou no módulo Action ou no módulo Broker seriam inseridas as regras. Caso isso ocorresse, esses módulos teriam uma complexidade maior do que deveriam visto que desempenhariam um papel a mais.

Para cada tabela em que haja a necessidade de inserir ou recuperar algum dado no banco de dados devido a uma requisição do usuário, há uma classe Rules correspondente.

As classes que pertencem a este módulo são mostradas na figura abaixo.

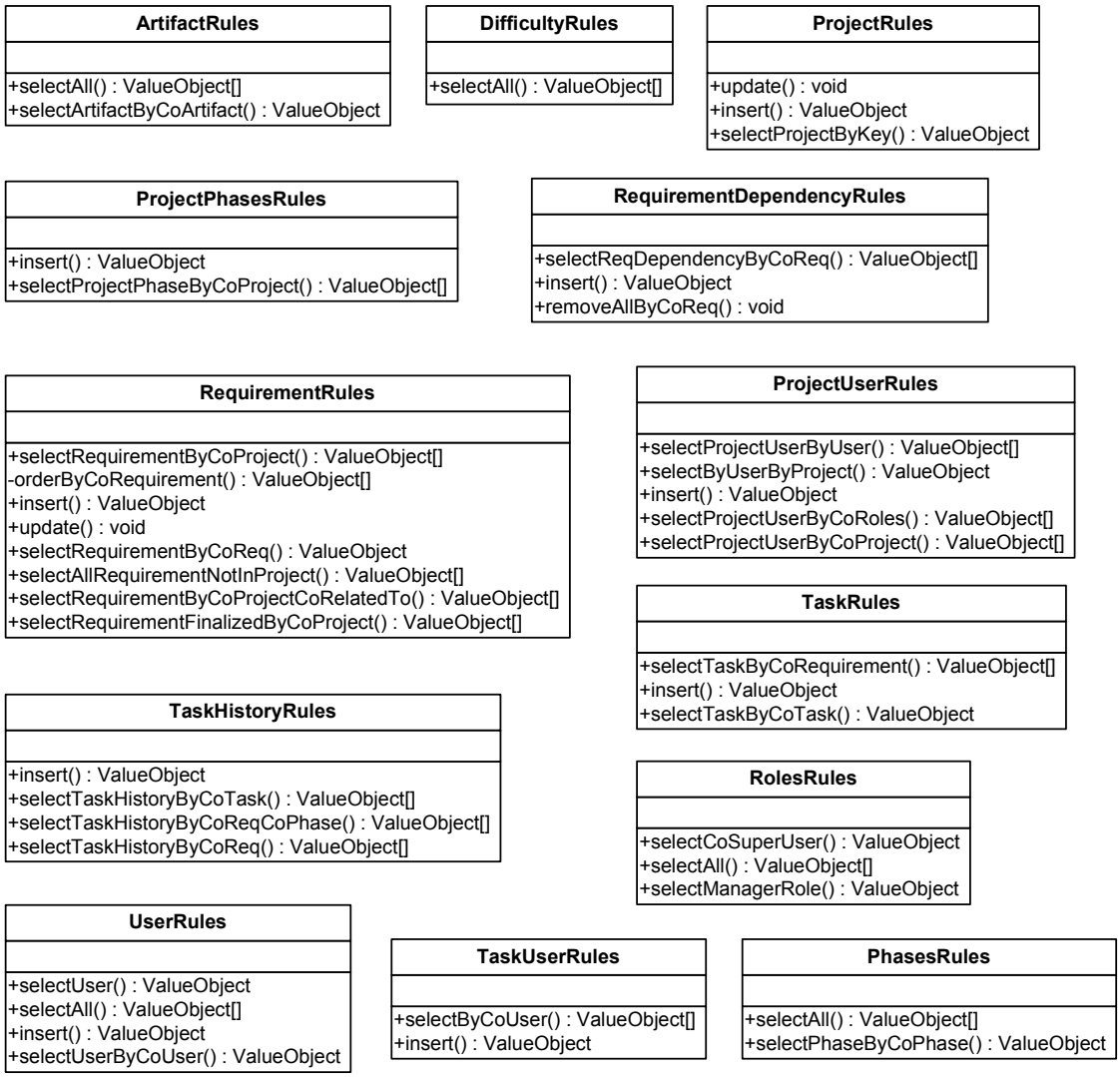


Figura 9: Classes do módulo Rules

8.2.2.5 Módulo Broker

O módulo Broker é responsável por se comunicar com o banco de dados e persistir um dado ou recuperar um dado já persistido.

Este módulo é composto de sub-módulos. Cada sub-módulo representa uma tabela no banco e é consistido de quatro classes. São elas:

- Broker: responsável por inserir ou recuperar um dado do banco de dados;
- Defaults: uma classe que possui somente atributos, não possuindo métodos. Tais atributos correspondem aos atributos de uma tabela do banco e são usados como chave quando o aplicativo faz o uso do padrão ValueObject;
- PK: representam a chave de uma tabela do banco de dados e é usado quando se faz uma pesquisa de um dado no banco por sua chave;

- ValueObjectHelper: uma classe auxiliar para um ValueObject;

8.2.2.6 JSP

O módulo JSP representa a camada de apresentação e pertencem a este módulo páginas JSP. Como dito anteriormente, cada página JSP tem um Action associado.

8.2.2.7 Classes utilitárias

As classes utilitárias foram concebidas para se evitar a redundância de código nos outros módulos. São duas as classes utilitárias: FormToValueObject, ValueObjectToForm.

Essas classes fazem a conversão de ActionForm para ValueObject e de ValueObject para ActionForm respectivamente.

A necessidade da conversão ActionForm-ValueObject se deve ao fato de que a camada de persistência aceita ou retorna ValueObject. O que importa para a camada de persistência são os dados em si e não uma instância, por exemplo de um requisito específico, sendo persistida ou recuperada do banco. Por isso o uso do padrão ValueObject, que nada mais é do que um agregador de dados.

Enquanto isso, a camada de apresentação faz somente o uso de ActionForm, ou seja, instâncias de alguma classe que pertença ao módulo Form. Isso ocorre porque o *framework* Strutus fornece *tags* que automatizam, por exemplo, a geração de uma tabela HTML e é necessário que os dados para o preenchimento dessa tabela sejam instâncias de ActionForm. Logo, a necessidade de conversão de ValueObject para ActionForm se deve ao fato de que esses dados estão no banco de dados e a camada de persistência retorna somente ValueObject.

As classes utilitárias são apresentadas na figura 10.

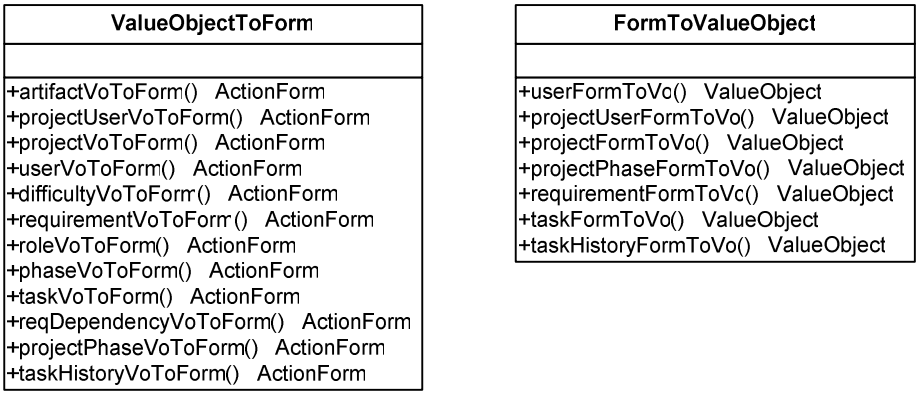


Figura 10: Classes utilitárias

8.3 Implementação

O aplicativo tem como interface inicial, uma tela de login(ver figura 11). A validação serve para o aplicativo fazer alguma restrição de funcionalidade, se necessário. Por exemplo, um *stakeholder* que não seja o gerente do projeto não poderá definir as dependências de um requisito.

Figura 11: Validação de usuário

Feita a validação e a escolha do projeto a ser visualizado, tem-se a descrição do projeto e seus requisitos. Para diferentes usuários há diferentes funcionalidades. Na figura 12 exibe-se quando um gerente de projeto entra no sistema e na figura 13 quando um outro participante do projeto que não seja um gerente de projeto entra no sistema.

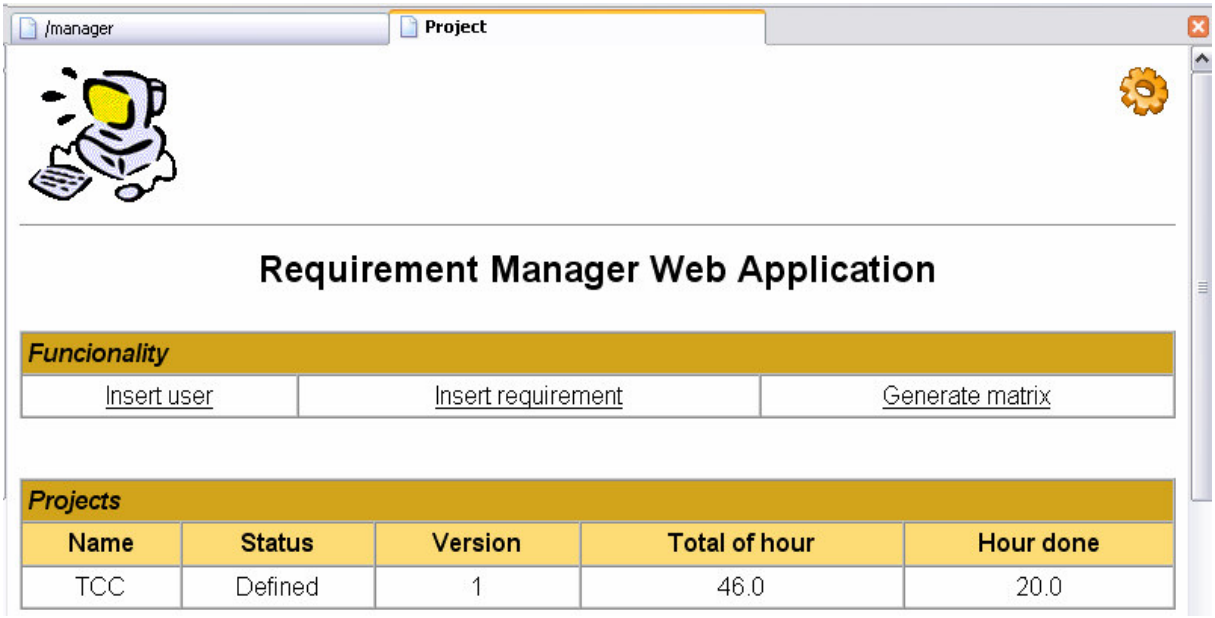


Figura 12: Descrição do projeto para gerente de projeto

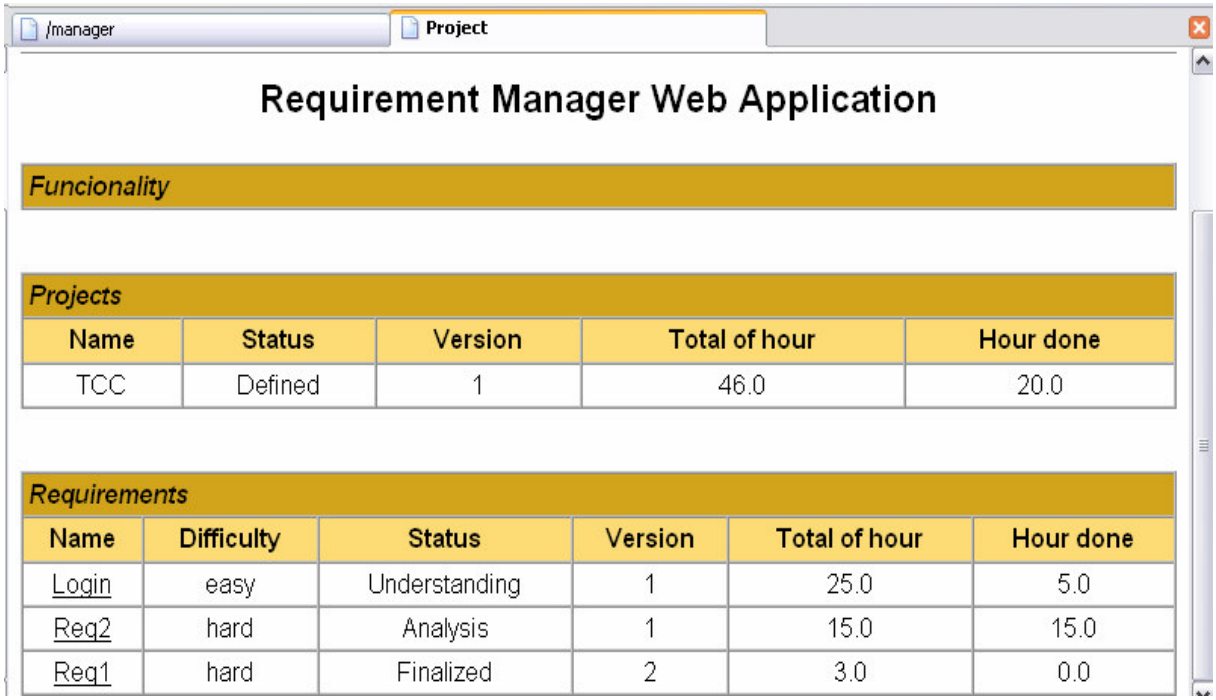


Figura 13: Descrição do projeto para outros participantes

Quando o gerente de projeto entra no sistema e escolhe o projeto em que ele deseja visualizar(ver figura 12), ele tem disponível quatro funcionalidades: inserir um usuário ao projeto, inserir requisitos ao sistema, ver a matriz de rastreabilidade dos requisitos ou ver a descrição de um requisito específico.

Se escolhida a opção de visualizar a descrição de um requisito específico, será mostrada uma tela semelhante à figura 14, se o usuário for um gerente do projeto.

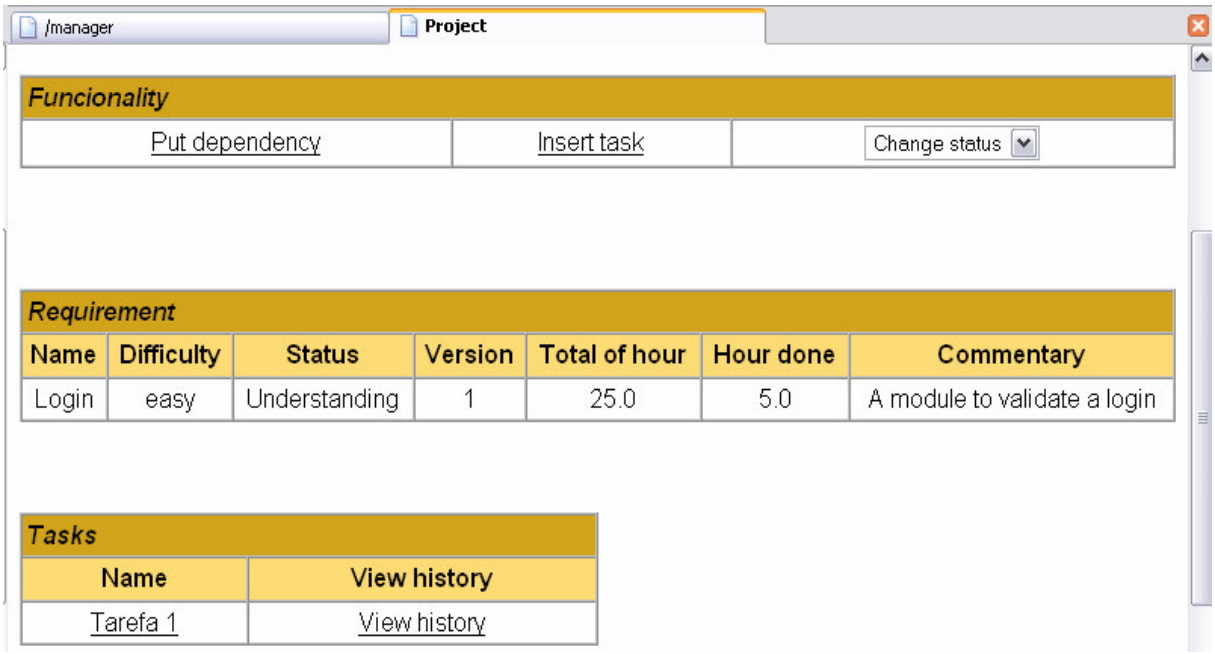


Figura 14: Descrição de um requisito

O gerente de projeto que escolha ver a descrição do requisito, terá as seguintes opções: colocar dependências no requisito, inserir tarefas para implementar o requisito, mudar o status de um requisito, cadastrar as horas de alguma tarefa que ele tenha feito e visualizar as horas cadastradas de todos os usuários. Vale ressaltar que um usuário que não seja o gerente de projeto terá somente a opção de cadastrar as horas de alguma tarefa que ele tenha feito.

Nesta seção foi mostrado somente as principais interfaces da ferramenta. Não foi apresentada todas as interfaces de todas as funcionalidades do aplicativo. Foi mostrado somente onde é possível fazer o uso dessas funcionalidades. Uma exibição mais detalhada das interfaces e das funcionalidades será feita na seção seguinte, onde será abordado um exemplo de uso do aplicativo.

8.4 Estudo de caso

Nesta seção será feita um estudo de caso do aplicativo para mostar o uso da ferramenta e suas funcionalidades.

Inicialmente, o adminstrador do sistema deve inserir um novo projeto. Para isso, ele deve definir um nome a ele, selecionar as fases e os gerentes do projeto, como mostra a figura abaixo.

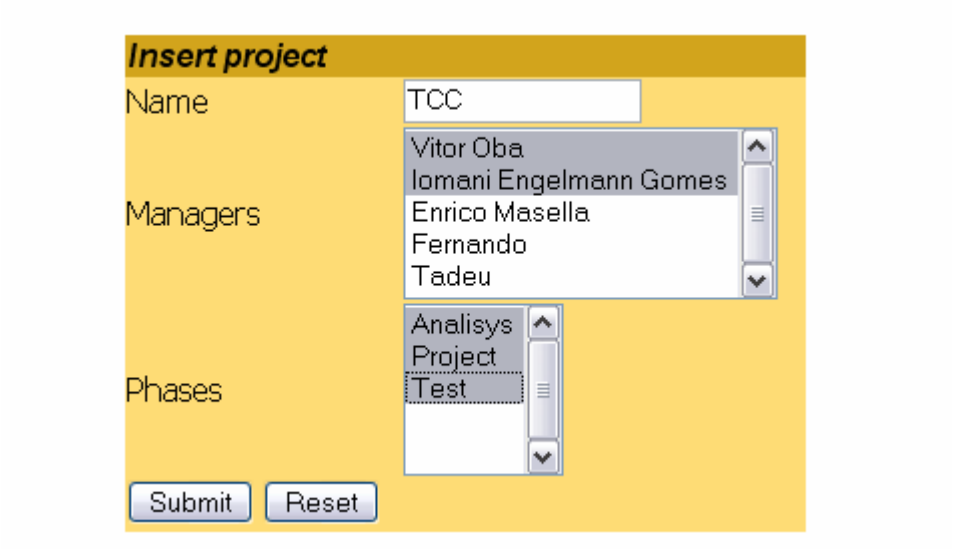


Figura 15: Inserindo um novo projeto

Inserido um projeto, o gerente de projeto pode fazer o uso do sistema fornecendo seu nome de usuário e sua senha. Feita a validação, será mostrado todos os projetos em que o usuário validado participa. Neste exemplo de uso foi inserido somente um projeto, logo será exibido somente este projeto(ver figura 16).

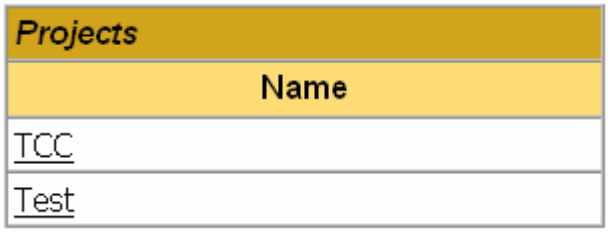
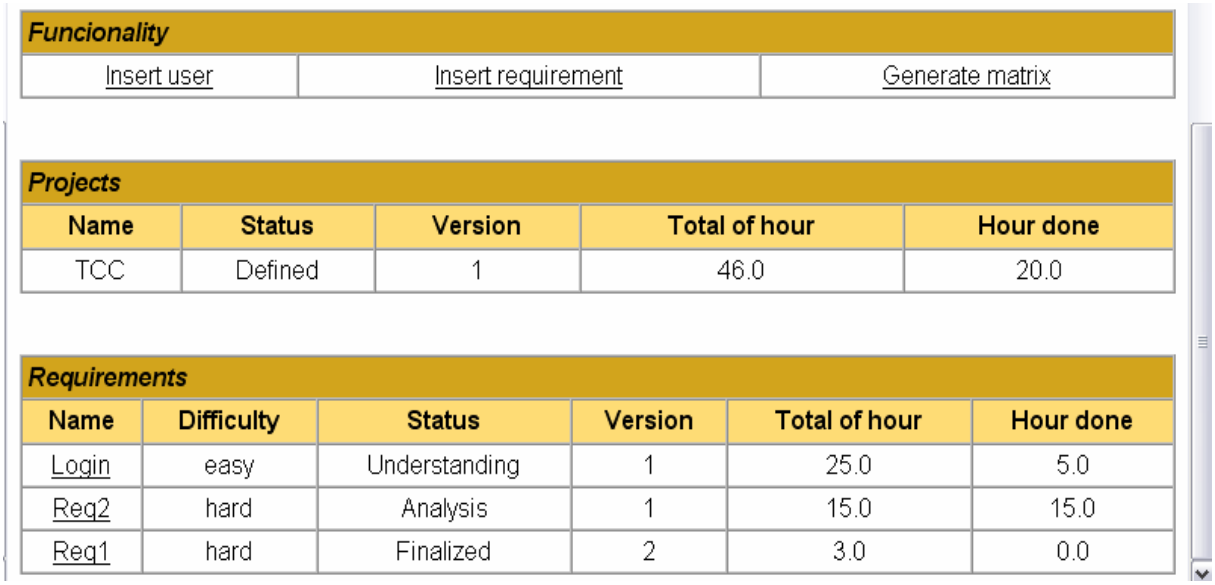


Figura 16: Projeto em que o usuário participa

Selecionado o projeto, o gerente de projeto terá uma descrição do projeto e três funcionalidades: inserir um usuário ao projeto, inserir um requisito ao projeto e gerar a matriz de rastreabilidade, como mostra a figura abaixo.



Funcionality

Insert user	Insert requirement	Generate matrix

Projects

Name	Status	Version	Total of hour	Hour done
TCC	Defined	1	46.0	20.0

Requirements

Name	Difficulty	Status	Version	Total of hour	Hour done
Login	easy	Understanding	1	25.0	5.0
Req2	hard	Analysis	1	15.0	15.0
Req1	hard	Finalized	2	3.0	0.0

Figura 17: Descrição do projeto

Se o gerente de projeto desejar inserir um usuário ao projeto, basta selecionar a opção inserir um usuário e escolher o usuário a ser inserido e o papel dele dentro do projeto(ver figura 18).

Figura 18: Inserir um novo usuário ao projeto

Selecionando a opção de inserir um requisito ao sistema, o gerente terá três alternativas: inserir um novo requisito puro e simples, inserir uma nova versão de um requisito que pertença ao projeto e que esteja finalizado ou importar um requisito que esteja finalizado e que não pertança ao projeto(ver figura 19).

Insert requirement

Insert a new requirement	Insert a new version of a requirement	Import a requirement

Figura 19: Três alternativas de inserção de requisito

Para inserir um novo requisito puro e simples, o gerente de projeto deve fornecer um nome, uma dificuldade, uma estimativa de horas para implementá-lo e se desejar um comentário ao requisito(ver figura 20).

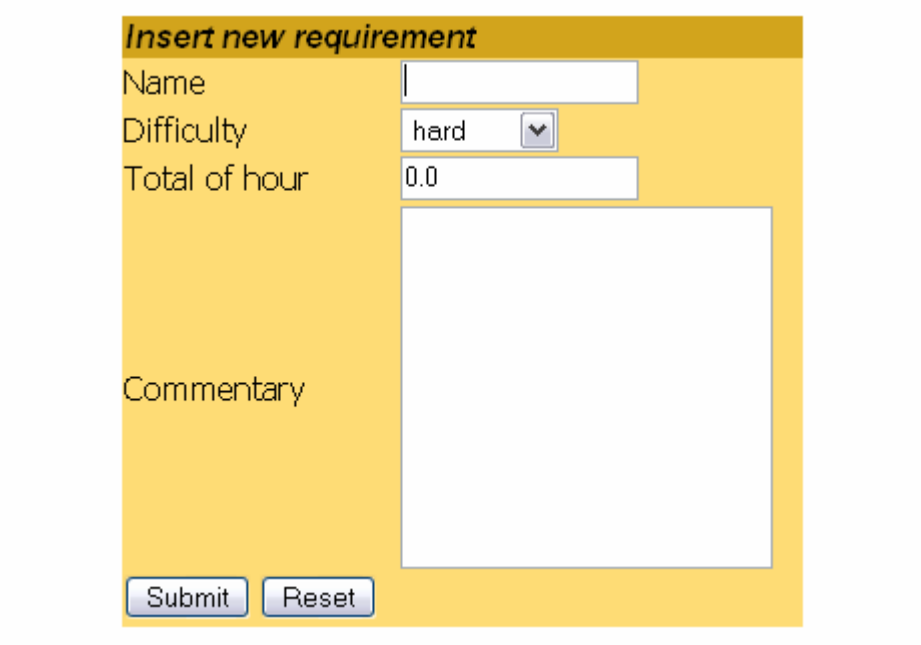


Figura 20: Inserindo um novo requisito puro e simples

Após feita a inserção do requisito, é mostrada a interface de descrição do projeto com a descrição do requisito inserido, ilustrada na figura 21.

Projects

Name	Status	Version	Total of hour	Hour done
TCC	Defined	1	46.0	20.0

Requirements

Name	Difficulty	Status	Version	Total of hour	Hour done
Login	easy	Understanding	1	25.0	5.0
Req2	hard	Analysis	1	15.0	15.0
Req1	hard	Finalized	2	3.0	0.0

Figura 21: Descrição do projeto com o requisito inserido

Optando por inserir uma nova versão de um requisito, o gerente de projeto deve selecionar qual o requisito em que deseja criar uma nova versão(ver figura 22). Vale ressaltar que é permitido somente criar novas versões de requisitos onde sua versão esteja fechada e que este requisito pertença ao projeto.

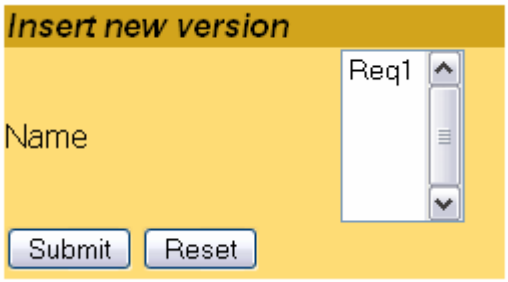


Figura 22: Interface para inserir uma nova versão de um requisisto

Há também a possibilidade de inserir um requisito ao projeto fazendo a importação deste. Se assim desejar, o gerente de projeto deve selecionar um requisito a ser importado(ver figura 23). O gerente de projeto pode importar somente um requisito em que sua versão esteja fechada e que não pertença ao projeto em que se deseja importar o requisito. Além disso, os requisitos de que ele depende são importados automaticamente.

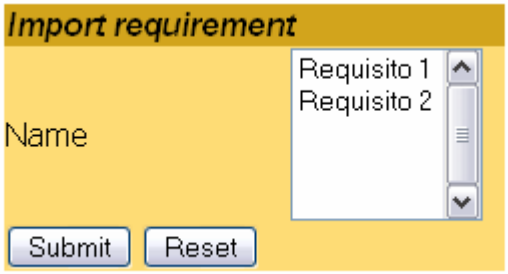


Figura 23: Interface para importar um requisito

Agora que se tem um requisito dentro do projeto, pode-se manipular esse requisito, ou seja, colocar as dependências, tarefas para implementá-lo ou mudar seu status. Para isso, basta selecionar o requisito desejado. Quando um requisito é selecionado, é mostrado suas informações e as tarefas criadas para implementá-lo. A figura abaixo mostra a interface quando o requisito é selecionado.

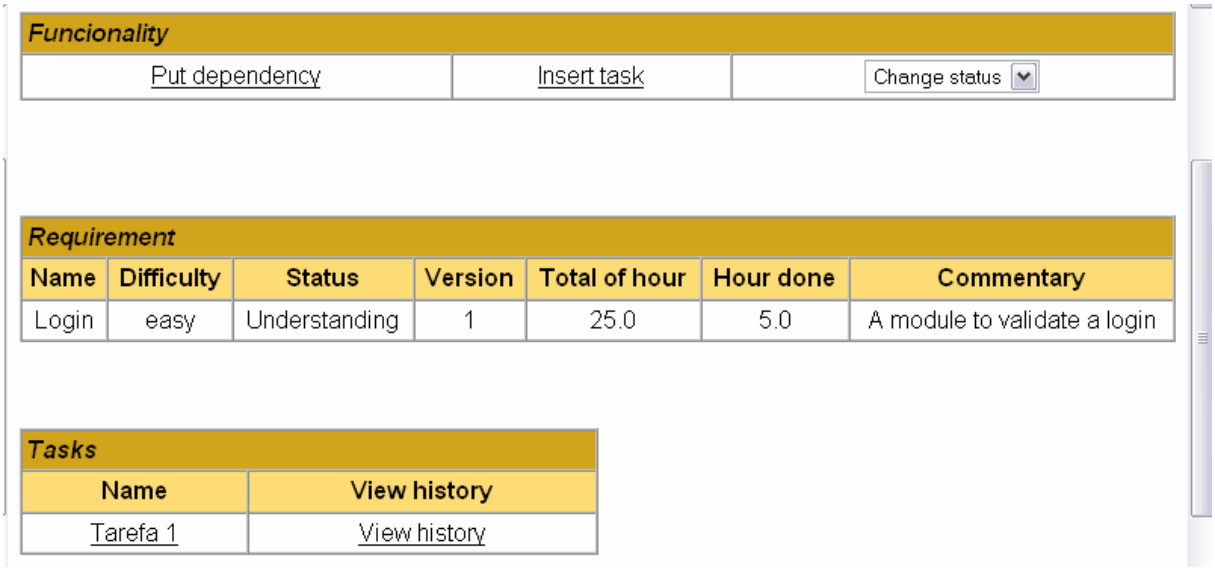


Figura 24: Informações sobre o requisito selecionado

Requisitos podem conter dependências. Satisfazer essa premissa, requer que o gerente de projeto selecione a funcionalidade de inserção de dependências e selecione os requisitos o qual um determinado requisito dependa(ver figura 25).

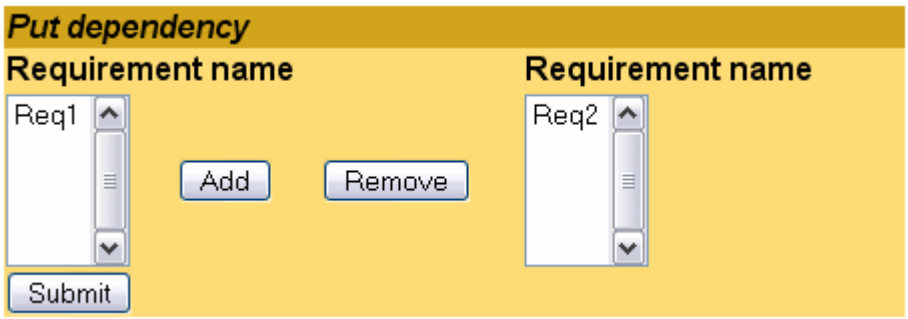


Figura 25: Interface para a inserção de um requisito

É necessário uma ou mais tarefas para implementar um requisito. O gerente de projeto adiciona uma tarefa ao requisito colocando uma descrição a ela(ver figura 26). A tarefa adicionada é automaticamente atribuída ao gerente de projeto e pode ser atrbuída a um outro *stakeholder* se o gerente de projeto assim desejar.

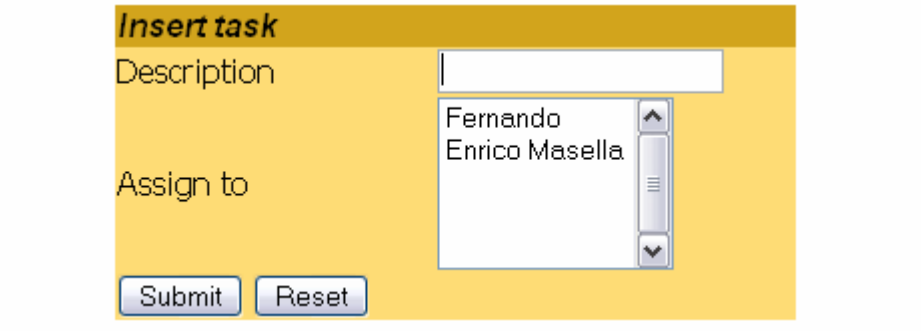


Figura 26: Interface para a inserção de tarefas

Após feita a inserção da tarefa, é mostrada a interface de descrição do requisito com a a tarefa inserida, ilustrada na figura 27.

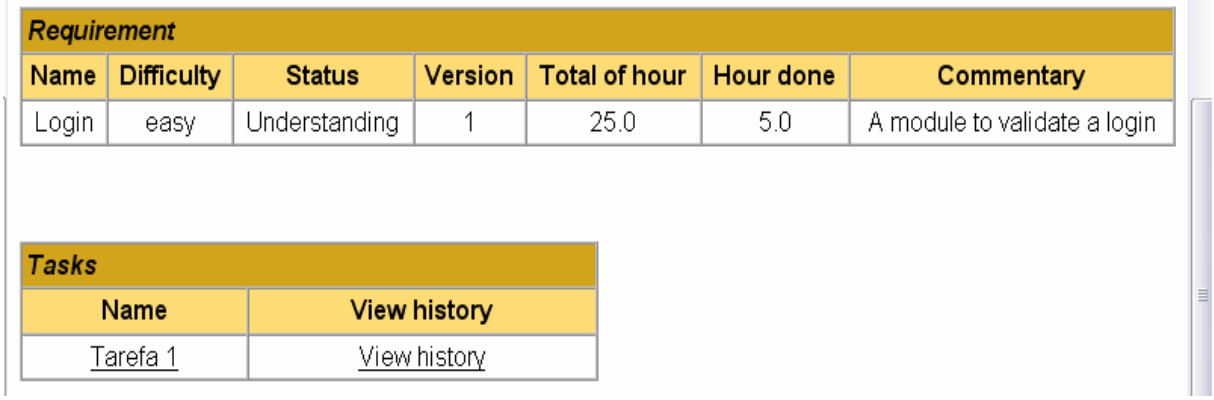


Figura 27: Descrição do requisito com a tarefa inserida

O usuário pode cadastrar as horas feitas numa atividade selecionando a tarefa em que a atividade se enquadra e informando o artefato feito, a quantidade de horas e a fase de desenvolvimento(ver figura 28).

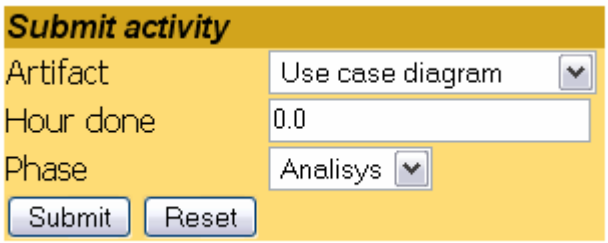


Figura 28: Interface para cadastro de horas de uma atividade

O gerente de projeto pode visualizar a qualquer momento todas as horas cadastradas das tarefas. Será fornecida uma tabela mostrando o usuário que cadastrou as horas, o artefato feito na atividade, a fase de desenvolvimento e as horas gastas(ver figura 29).

Task history

User name	Artifact	Phase	Hour done
Vitor Oba	Use case diagram	Analisys	5.0

Figura 29: Interface mostrando as horas cadastradas

Na interface onde são mostradas informações sobre um requisito selecionado, há também a possibilidade de mudar o status do requisito. Nesta interface existe um *combo box* mostrando as opções de status disponíveis a partir do status atual do requisito, sempre obedecendo a máquina de estados mostrada anteriormente.

Vale ressaltar que se o status atual do requisito for o status definido, o gerente de projeto não poderá inserir dependências do requisito. Se o status atual do requisito for o status finalizado, a única funcionalidade disponível será a visualização das horas cadastradas das tarefas.

Na tela de implementação onde é mostrada a descrição do projeto, existe a funcionalidade de geração da matriz de rastreabilidade. Selecionando essa funcionalidade, o gerente de projeto irá visualizar uma interface semelhante à figura 30.

Matrix of traceability

	Analisys	Project	Test
Login	2		
Req2	3	4	
Req1			

Description

ID	Task name	User name	Artifact	Hour done
2	Tarefa 1	Vitor Oba	Use case diagram	5.0
3	Test	Vitor Oba	State diagram	15.0
4	Test	Vitor Oba	Activity diagram	0.0

Figura 30: Matriz de rastreabilidade

Pela figura 30 nota-se que existem duas tabelas. A tabela superior representa a matriz de rastreabilidade, onde as colunas são as fases do projeto e as linhas são os requisitos do projeto. A

relação linha-coluna é preenchida com identificadores. Uma descrição mais detalhada desta relação é apresentada na tabela inferior.

Com este estudo de caso, foi mostrada a ferramenta proposta sendo usada em um projeto.

9 CONCLUSÃO

Um projeto que atenda às expectativas de um cliente deve ser executado com um série de passos, sendo um deles o gerenciamento de requisitos.

Ferramentas são utlizadas no gerenciamento. Porém as ferramentas atuais não automatizam aas tarefas relacionadas ao gerenciamento, provocando resistências ou até mesmo o abandono das tarefas, mesmo considerados de suma importância.

A ferramenta proposta neste trabalho auxilia no gerenciamento de requisitos de um projeto. Ela semi-automatiza as tarefas relativas ao gerenciamento, reduzindo assim o custo relacionado à burocracia, visto que nos aplicativos atuais é necessário o preenchimento extensivo de documentos. Com isso elemina o tempo gasto com o gerenciamento ou mesmo atualização dos documentos.

Logo, conclui-se que adoção de um modelo de processo como o CMMI e a utilização de ferramentas que automatizem as tarefas relacionados ao CMMI é de grande importância, especialmente em lugares onde o processo de desenvolvimento de software é caótico.

10 REFERÊNCIAS BIBLIOGRÁFICAS

[CSR 02] Carnegie Mellon University / Software Engineering Institute. Capability Maturity Model Integration (CMMISM) – Staged Representation, Version 1.1, 2002, 639p.

[CCR 02] Carnegie Mellon University / Software Engineering Institute. Continuous Maturity Model Integration (CMMISM) – Staged Representation, Version 1.1, 2002, 645p.

[TCC 03] Matias, Carlos A., Winck, Ricardo J. Suporte à rastreabilidade de informações ao longo das fases de desenvolvimento de software, 2003, 80p.

[CRA 02] Larman, Craig. Utilizando UML e padrões – Uma introdução à análise e ao projeto orientados a objetos, 2002, 492p.

[JMA 01] Java Magazine. Programação para internet, 2001, 54p.

11 ANEXOS

11.1 Código fonte

11.1.1 Módulo Form

```
package tcc.form;

import org.apache.struts.action.ActionForm;

public class ArtifactForm extends ActionForm{
	private int coArtifact;
	private String noArtifact;

	public int getCoArtifact() {
		return coArtifact;
	}
	public void setCoArtifact(int coArtifact) {
		this.coArtifact = coArtifact;
	}
	public String getNoArtifact() {
		return noArtifact;
	}
	public void setNoArtifact(String noArtifact) {
		this.noArtifact = noArtifact;
	}
}

package tcc.form;

import org.apache.struts.action.ActionForm;

public class DifficultyForm extends ActionForm {
	private int coDifficulty;
	private String name="";
```

```
	public int getCoDifficulty() {
		return coDifficulty;
	}

	public void setCoDifficulty(int codifficulty) {
		this.coDifficulty = codifficulty;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
}

package tcc.form;

import org.apache.struts.action.ActionForm;

public class PhasesForm extends ActionForm {
	private int coPhase;
	private String name;

	public PhasesForm(){}

	public int getCoPhase() {
		return coPhase;
	}

	public void setCoPhase(int coPhase) {
		this.coPhase = coPhase;
```

```
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
}

package tcc.form;

import org.apache.struts.action.ActionForm;

public class ProjectForm extends ActionForm {
	private int coProject;
	private String name;
	private int phases;
	private String status;
	private String version;
	private double totalHour;
	private double hourDone;

	private int coManagers[];
	private int coPhases[];

	public ProjectForm(){}

	public int getCoProject() {
		return coProject;
	}

	public void setCoProject(int coProject) {
```

```
        this.coProject = coProject;
}
public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}
public int[] getCoManagers() {
        return coManagers;
}

public void setCoManagers(int[] coManagers) {
        this.coManagers = coManagers;
}

public double getHourDone() {
        return hourDone;
}

public void setHourDone(double hourDone) {
        this.hourDone = hourDone;
}

public String getStatus() {
        return status;
}

public void setStatus(String status) {
        this.status = status;
}

public double getTotalHour() {
```

```
		return totalHour;
	}

	public void setTotalHour(double totalHour) {
		this.totalHour = totalHour;
	}

	public String getVersion() {
		return version;
	}

	public void setVersion(String version) {
		this.version = version;
	}

	public int getPhases() {
		return phases;
	}

	public void setPhases(int phases) {
		this.phases = phases;
	}

	public int[] getCoPhases() {
		return coPhases;
	}

	public void setCoPhases(int[] coPhases) {
		this.coPhases = coPhases;
	}
}

package tcc.form;
```

```
import org.apache.struts.action.ActionForm;

public class ProjectPhasesForm extends ActionForm {
	private int coProjectPhase;
	private int coProject;
	private int coPhase;

	public int getCoPhase() {
		return coPhase;
	}

	public void setCoPhase(int coPhase) {
		this.coPhase = coPhase;
	}

	public int getCoProject() {
		return coProject;
	}

	public void setCoProject(int coProject) {
		this.coProject = coProject;
	}

	public int getCoProjectPhase() {
		return coProjectPhase;
	}

	public void setCoProjectPhase(int coProjectPhase) {
		this.coProjectPhase = coProjectPhase;
	}
}

package tcc.form;
```

```
public class ProjectRequirement {
	private int coProjectRequirement;
	private int coProject;
	private int coRequirement;

	public int getCoProject() {
		return coProject;
	}

	public void setCoProject(int coProject) {
		this.coProject = coProject;
	}

	public int getCoProjectRequirement() {
		return coProjectRequirement;
	}

	public void setCoProjectRequirement(int coProjectRequirement) {
		this.coProjectRequirement = coProjectRequirement;
	}

	public int getCoRequirement() {
		return coRequirement;
	}

	public void setCoRequirement(int coRequirement) {
		this.coRequirement = coRequirement;
	}
}

package tcc.form;

import org.apache.struts.action.ActionForm;
```

```
public class ProjectUserForm extends ActionForm {
	private int coUser;
	private int coProject;
	private int coRoles;

	public ProjectUserForm(){}

	public int getCoProject() {
		return coProject;
	}

	public void setCoProject(int coProject) {
		this.coProject = coProject;
	}

	public int getCoRoles() {
		return coRoles;
	}

	public void setCoRoles(int coRoles) {
		this.coRoles = coRoles;
	}

	public int getCoUser() {
		return coUser;
	}

	public void setCoUser(int coUser) {
		this.coUser = coUser;
	}
}

package tcc.form;
```

```
import org.apache.struts.action.ActionForm;

public class RequirementDependencyForm extends ActionForm{
	private int coReqDependency;
	private int coRequirement;
	private int coDependantReq;

	private int coRequirements[]; //it stores all the requirements depandant

	public int getCoDependantReq() {
		return coDependantReq;
	}
	public void setCoDependantReq(int coDependantReq) {
		this.coDependantReq = coDependantReq;
	}
	public int getCoReqDependency() {
		return coReqDependency;
	}
	public void setCoReqDependency(int coReqDependency) {
		this.coReqDependency = coReqDependency;
	}
	public int getCoRequirement() {
		return coRequirement;
	}
	public void setCoRequirement(int coRequirement) {
		this.coRequirement = coRequirement;
	}
	public int[] getCoRequirements() {
		return coRequirements;
	}
	public void setCoRequirements(int[] coRequirements) {
		this.coRequirements = coRequirements;
	}
}
```

```
package tcc.form;

import org.apache.struts.action.ActionForm;

public class RequirementForm extends ActionForm {
	private int coRequirement;
	private String noStatus;
	private int coDifficulty;
	private String noVersion="";
	private double totalHour;
	private double hourDone;
	private String noCommentary="";
	private String name="";
	private int coFirstRequirement;
	private int coBackRequirement;
	private int coProject;
	private int coRelatedTo;

	private int coRequirements[];

	public RequirementForm() {

	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
	public int getCoDifficulty() {
		return coDifficulty;
```

```
}

public void setCoDifficulty(int coDificultty) {
	this.coDifficulty = coDificultty;
}

public int getCoRequirement() {
	return coRequirement;
}

public void setCoRequirement(int coRequirement) {
	this.coRequirement = coRequirement;
}

public String getNoStatus() {
	return noStatus;
}

public void setNoStatus(String coStatus) {
	this.noStatus = coStatus;
}

public double getHourDone() {
	return hourDone;
}

public void setHourDone(double hourDone) {
	this.hourDone = hourDone;
}

public String getNoCommentary() {
	return noCommentary;
}
```

```
public void setNoCommentary(String noCommentary) {
	this.noCommentary = noCommentary;
}

public String getNoVersion() {
	return noVersion;
}

public void setNoVersion(String noVersion) {
	this.noVersion = noVersion;
}

public double getTotalHour() {
	return totalHour;
}

public void setTotalHour(double totalHour) {
	this.totalHour = totalHour;
}

public int getCoBackRequirement() {
	return coBackRequirement;
}

public void setCoBackRequirement(int coBackRequirement) {
	this.coBackRequirement = coBackRequirement;
}

public int getCoFirstRequirement() {
	return coFirstRequirement;
}

public void setCoFirstRequirement(int coFirstRequirement) {
	this.coFirstRequirement = coFirstRequirement;
```

```
	}

	public int[] getCoRequirements() {
		return coRequirements;
	}

	public void setCoRequirements(int[] coRequirements) {
		this.coRequirements = coRequirements;
	}

	public int getCoProject() {
		return coProject;
	}

	public void setCoProject(int coProject) {
		this.coProject = coProject;
	}

	public int getCoRelatedTo() {
		return coRelatedTo;
	}

	public void setCoRelatedTo(int coRelatedTo) {
		this.coRelatedTo = coRelatedTo;
	}
}

package tcc.form;

import org.apache.struts.action.ActionForm;

public class RolesForm extends ActionForm {
	private int coRoles;
	private String name;
```

```
	public RolesForm() {

	}

	public int getCoRoles() {
		return coRoles;
	}

	public void setCoRoles(int coRoles) {
		this.coRoles = coRoles;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
}

package tcc.form;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class StatusForm extends ActionForm{
	private String noName;

	public String getNoName() {
```

```
		return noName;
	}
	public void setNoName(String noName) {
		this.noName = noName;
	}

	public ActionErrors validate(ActionMapping mapping, HttpServletRequest req) {

		if(noName.equals("Change status")) {

	req.getSession().getServletContext().getRequestDispatcher("/jsp/requirement/requirement.js
p");
		}
		return super.validate(mapping, req);
	}
}

package tcc.form;

import org.apache.struts.action.ActionForm;

public class TaskForm extends ActionForm {
	private int coTask;
	private String noTask;
	private int coRequirement;
	private String noCommentary;

	private int coUsers[];

	public int getCoRequirement() {
		return coRequirement;
	}
	public void setCoRequirement(int coRequirement) {
		this.coRequirement = coRequirement;
```

```
	}
	public int getCoTask() {
		return coTask;
	}
	public void setCoTask(int coTask) {
		this.coTask = coTask;
	}
	public String getNoCommentary() {
		return noCommentary;
	}
	public void setNoCommentary(String noCommentary) {
		this.noCommentary = noCommentary;
	}
	public String getNoTask() {
		return noTask;
	}
	public void setNoTask(String noTask) {
		this.noTask = noTask;
	}
	public int[] getCoUsers() {
		return coUsers;
	}
	public void setCoUsers(int[] coUsers) {
		this.coUsers = coUsers;
	}
}

package tcc.form;

import org.apache.struts.action.ActionForm;

public class TaskHistoryForm extends ActionForm{
	private int coTaskHistory;
	private int coUser;
```

```
private int coTask;
private int coArtifact;
private double nuHourDone;
private int coPhase;

public int getCoArtifact() {
	return coArtifact;
}
public void setCoArtifact(int coArtifact) {
	this.coArtifact = coArtifact;
}
public int getCoPhase() {
	return coPhase;
}
public void setCoPhase(int coPhase) {
	this.coPhase = coPhase;
}
public int getCoTask() {
	return coTask;
}
public void setCoTask(int coTask) {
	this.coTask = coTask;
}
public int getCoTaskHistory() {
	return coTaskHistory;
}
public void setCoTaskHistory(int coTaskHistory) {
	this.coTaskHistory = coTaskHistory;
}
public int getCoUser() {
	return coUser;
}
public void setCoUser(int coUser) {
	this.coUser = coUser;
```

```
	}
	public double getNuHourDone() {
		return nuHourDone;
	}
	public void setNuHourDone(double nuHourDone) {
		this.nuHourDone = nuHourDone;
	}
}

package tcc.form;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class UserForm extends ActionForm {
	private int coUser;
	private String name;
	private String username;
	private String password;
	private String address;
	private String email;

	private String retypepassword;
	private int coRoles;

	public UserForm() {
		super();
	}

	public void reset(ActionMapping arg0, HttpServletRequest arg1) {
		name = "";
		username = "";
```

```
        password = "";
        address = "";
        email = "";
}

public int getCoUser() {
        return coUser;
}

public int getCoRoles() {
        return coRoles;
}

public void setCoRoles(int coRole) {
        this.coRoles = coRole;
}

public String getRetypepassword() {
        return retypepassword;
}

public void setRetypepassword(String retypepassword) {
        this.retypepassword = retypepassword;
}
public void setCoUser(int coUser) {
        this.coUser = coUser;
}

public String getAddress() {
        return address;
}

public void setAddress(String address) {
        this.address = address;
```

```
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public String getUsername() {
		return username;
	}

	public void setUsername(String username) {
		this.username = username;
	}
}
```

11.1.2 Módulo Rules

```
package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.artifact.ArtifactBroker;
import tcc.persistence.artifact.ArtifactPK;

import com.techlab.infrastructure.ejb.ValueObject;

public class ArtifactRules {
	public ValueObject[] selectAll(){
		ArtifactBroker broker = new ArtifactBroker();

		try {
			return broker.findAll();
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public ValueObject selectArtifactByCoArtifact(int coArtifact){
		ArtifactPK pk = new ArtifactPK(coArtifact);

		ArtifactBroker broker = new ArtifactBroker();
		try {
			return broker.findByKey(pk);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
```

```
	}
}

package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.artifact.ArtifactBroker;
import tcc.persistence.artifact.ArtifactPK;

import com.techlab.infrastructure.ejb.ValueObject;

public class ArtifactRules {
	public ValueObject[] selectAll(){
		ArtifactBroker broker = new ArtifactBroker();

		try {
			return broker.findAll();
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public ValueObject selectArtifactByCoArtifact(int coArtifact){
		ArtifactPK pk = new ArtifactPK(coArtifact);

		ArtifactBroker broker = new ArtifactBroker();
		try {
			return broker.findByKey(pk);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
```

```
	}
}

package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.phases.PhasesBroker;
import tcc.persistence.phases.PhasesPK;

import com.techlab.infrastructure.ejb.ValueObject;

public class PhasesRules {
	public ValueObject[] selectAll(){
		PhasesBroker broker = new PhasesBroker();
		try {
			return broker.findAll();
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}
	public ValueObject selectPhaseByCoPhase(int coPhase) {
		PhasesPK pk = new PhasesPK(coPhase);

		PhasesBroker broker = new PhasesBroker();
		try {
			return broker.findByKey(pk);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}
}
```

```
package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.projectphases.ProjectPhasesBroker;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;

public class ProjectPhasesRules {
	public ValueObject insert(ValueObject vo) {
		ProjectPhasesBroker broker = new ProjectPhasesBroker();
		try {
			return broker.insert(vo);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public ValueObject[] selectProjectPhaseByCoProject(int coProject) {
		String sql = "SELECT * FROM projectphases WHERE co_project="+coProject;

		DataBaseHandler handler = new DataBaseHandler();
		try {
			return handler.select(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}
}
```

```
package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.project.ProjectBroker;
import tcc.persistence.projectuser.ProjectUserDefaults;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;

public class ProjectRules {
	public void update(ValueObject vo) throws SQLException {
		ProjectBroker broker = new ProjectBroker();

		broker.update(vo);
	}

	public ValueObject insert(ValueObject project) {
		ProjectBroker broker = new ProjectBroker();
		try {
			return broker.insert(project);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public ValueObject selectProjectByKey(ValueObject vo) {
		String sql = "SELECT * FROM project WHERE co_project=\'"+vo.get(ProjectUserDefaults.COPROJECT)+"\'";

		DataBaseHandler handler = new DataBaseHandler();
		try {
			return handler.select(sql)[0];
```

```
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return null;
        }
}

package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.projectuser.ProjectUserBroker;
import tcc.persistence.user.UserDefaults;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;

public class ProjectUserRules {
        public ValueObject[] selectProjectUserByUser(ValueObject vo) {
                String sql="SELECT * FROM projectuser "+
                                "WHERE
co_user=\'"+((Integer)vo.get(UserDefaults.COUSER)).intValue()+"\'";
                System.out.println("SQL:="+sql);
                DataBaseHandler handler = new DataBaseHandler();
                try {
                        ValueObject vos[] = handler.select(sql);
                        if(vos.length==0) {
                                return null;
                        } else {
                                return vos;
                        }
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
```

```
        }
        return null;
}

public ValueObject selectByUserByProject(int coProject, int coUser){
        String sql = "SELECT * FROM projectuser " +
                        "WHERE co_project=\'"+coProject+"\' AND " +
                        "co_roles=\'"+coUser+"\'";

        DataBaseHandler handler = new DataBaseHandler();
        try {
                return handler.select(sql)[0];
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        return null;
}

public ValueObject insert(ValueObject vo) {
        ProjectUserBroker broker = new ProjectUserBroker();
        try {
                return broker.insert(vo);
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        return null;
}

public ValueObject[] selectProjectUserByCoRoles(int coRole){
        String sql = "SELECT * FROM projectuser WHERE co_roles="+coRole;
        DataBaseHandler handler = new DataBaseHandler();
        try {
```

```
                return handler.select(sql);
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return null;
        }

        public ValueObject[] selectProjectUserByCoProject(int coProject) {
            String sql = "SELECT * FROM projectuser WHERE co_project="+coProject;

            DataBaseHandler handler= new DataBaseHandler();
            try {
                return handler.select(sql);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return null;
        }
}

package tcc.rules;

import java.rmi.RemoteException;
import java.sql.SQLException;

import tcc.persistence.requirementdependency.RequirementDependencyBroker;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;

public class RequirementDependencyRules {
        public ValueObject[] selectReqDependencyByCoReq(int coRequirement) {
```

```
		String sql = "SELECT * FROM requirementdependency WHERE co_requirement="+coRequirement;

		DataBaseHandler handler = new DataBaseHandler();
		try {
			return handler.select(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public void insert(ValueObject vo) {
		RequirementDependencyBroker broker = new RequirementDependencyBroker();
		try {
			broker.insert(vo);
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	public void removeAllByCoReq(int coRequirement)  {
		String sql = "DELETE FROM requirementdependency WHERE co_requirement="+coRequirement;

		DataBaseHandler handler = new DataBaseHandler();
		try {
			handler.delete(sql);
		} catch (RemoteException e) {
			e.printStackTrace();
		}
	}
}
```

```
package tcc.rules;

import java.sql.SQLException;
import java.util.ArrayList;

import tcc.form.ProjectForm;
import tcc.persistence.requirement.RequirementBroker;
import tcc.persistence.requirement.RequirementDefaults;
import tcc.persistence.requirement.RequirementPK;
import tcc.status.StatusStateMachine;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;

public class RequirementRules {
	public ValueObject[] selectRequirementByCoProject(ProjectForm vo) {
		String sql="SELECT * FROM requirement " +
				"WHERE co_project=\'"+vo.getCoProject()+"\'";
		DataBaseHandler handler = new DataBaseHandler();
		try {
			ValueObject vos[] = handler.select(sql);
			ArrayList list = new ArrayList();
			for(int i=0;i<vos.length;i++) {
				ValueObject v = vos[i];

	if(((Integer)v.get(RequirementDefaults.COREQUIREMENT)).intValue() ==

((Integer)v.get(RequirementDefaults.COINITIALREQUIREMENT)).intValue()) {
					String s = "SELECT * FROM requirement WHERE
co_initialrequirement=" +

((Integer)vos[i].get(RequirementDefaults.COINITIALREQUIREMENT)).intValue() +
					" AND co_requirement IN "+
```

```
                                                "(SELECT    MAX(co_requirement)    FROM
requirement WHERE "+
                                                "co_initialrequirement="+

((Integer)vos[i].get(RequirementDefaults.COINITIALREQUIREMENT)).intValue()+")";
                                        if(handler.select(s).length > 0) {
                                                list.add(handler.select(s)[0]);
                                        }
                                }
                        }
                        return orderByCoRequirement(list);
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                return  null;
        }

        private ValueObject[] orderByCoRequirement(ArrayList list) {
                ValueObject vos[] = new ValueObject[list.size()];
                int min = 0;
                int index = 0;

                for(int i=0;i<list.size();i++) {
                        vos[i] = (ValueObject)list.get(i);
                }

                for(int i=0;i<vos.length;i++) {
                        ValueObject v1 = (ValueObject) vos[i];
                        min = i;
                        for(int j=i+1;j<vos.length;j++) {
                                ValueObject v2 = (ValueObject)vos[j];

        if(((Integer)v2.get(RequirementDefaults.COREQUIREMENT)).intValue()<
```

```
((Integer)v1.get(RequirementDefaults.COREQUIREMENT)).intValue()) {
                                        min = j;
                                        v1 = v2;
                                }
                        }
                        ValueObject temp = vos[min];
                        vos[min] = vos[i];
                        vos[i] = temp;
                }

                return vos;
        }

        public ValueObject insert(ValueObject vo) throws SQLException{
                RequirementBroker broker = new RequirementBroker();

                return broker.insert(vo);
        }

        public void update(ValueObject vo) {
                RequirementBroker broker = new RequirementBroker();
                try {
                        broker.update(vo);
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }

        public ValueObject selectRequirementByCoReq(int coReq) {
                RequirementPK pk = new RequirementPK(coReq);

                RequirementBroker broker = new RequirementBroker();
```

```
		try {
			return broker.findByKey(pk);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public ValueObject[] selectAllRequirementNotInProject(int coProject) {
		/*String sql="SELECT * FROM requirement " +
		  "WHERE co_project<>\""+coProject+"\' AND co_relatedto=0";*/
		/*String sql = "SELECT * FROM requirement WHERE co_requirement NOT IN "+
				" (SELECT co_requirement FROM projectrequirement WHERE co_project="+coProject+ ") "+
				" AND co_requirement=co_initialrequirement";*/
		String sql = "SELECT * FROM (SELECT * FROM requirement WHERE co_project<>"+coProject+" AND co_relatedto=0)"+
				" AS req WHERE co_requirement NOT IN (SELECT co_relatedto FROM requirement WHERE co_project="+
				coProject + " AND co_relatedto<>0)";
		DataBaseHandler handler = new DataBaseHandler();
		try {
			ValueObject vos[] = handler.select(sql);
			ArrayList list = new ArrayList();
			for(int i=0;i<vos.length;i++) {
				ValueObject v = vos[i];

	if(((Integer)v.get(RequirementDefaults.COREQUIREMENT)).intValue() ==

((Integer)v.get(RequirementDefaults.COINITIALREQUIREMENT)).intValue()) {
				String s = "SELECT * FROM requirement WHERE co_initialrequirement=" +

((Integer)vos[i].get(RequirementDefaults.COINITIALREQUIREMENT)).intValue() +
```

```
                                                    " AND co_requirement IN "+
                                                    "(SELECT MAX(co_requirement) FROM
requirement WHERE "+
                                                    "co_initialrequirement="+

((Integer)vos[i].get(RequirementDefaults.COINITIALREQUIREMENT)).intValue()+")" +
                                                    "AND
no_status=\'"+StatusStateMachine.FINALIZED+"\'";
                                        if(handler.select(s).length > 0) {
                                                list.add(handler.select(s)[0]);
                                        }
                                }
                        }
                        return orderByCoRequirement(list);
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                return  null;
        }

        public ValueObject[] selectRequirementByCoProjectCoRelatedTo(int coProject, int
coRequirement) {
                String sql = "SELECT * FROM requirement WHERE co_project="+coProject+
                                        " AND co_relatedto="+coRequirement;

                DataBaseHandler handler = new DataBaseHandler();
                try {
                        return handler.select(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return null;
        }
```

```
public ValueObject[] selectRequirementFinalizedByCoProject(int coProject) {
        String sql="SELECT * FROM requirement " +
          "WHERE co_project=\'"+coProject+"\'";
        DataBaseHandler handler = new DataBaseHandler();
        try {
                ValueObject vos[] = handler.select(sql);
                ArrayList list = new ArrayList();
                for(int i=0;i<vos.length;i++) {
                        ValueObject v = vos[i];

public ValueObject[] selectRequirementFinalizedByCoProject(int coProject) {
        String sql="SELECT * FROM requirement " +
          "WHERE co_project=\'"+coProject+"\'";
        DataBaseHandler handler = new DataBaseHandler();
        try {
                ValueObject vos[] = handler.select(sql);
                ArrayList list = new ArrayList();
                for(int i=0;i<vos.length;i++) {
                        ValueObject v = vos[i];

if(((Integer)v.get(RequirementDefaults.COREQUIREMENT)).intValue() ==

((Integer)v.get(RequirementDefaults.COINITIALREQUIREMENT)).intValue()) {
                                String s = "SELECT * FROM requirement WHERE
co_initialrequirement=" +

((Integer)vos[i].get(RequirementDefaults.COINITIALREQUIREMENT)).intValue() +
                                        " AND co_requirement IN "+
                                        "(SELECT MAX(co_requirement) FROM
requirement WHERE "+
                                        "co_initialrequirement="+

((Integer)vos[i].get(RequirementDefaults.COINITIALREQUIREMENT)).intValue()+")" +
                                        "AND
no_status=\'"+StatusStateMachine.FINALIZED+"\'";
                                if(handler.select(s).length > 0) {
                                        list.add(handler.select(s)[0]);
                                }
                        }
                }
                return orderByCoRequirement(list);
        } catch (SQLException e) {
                // TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }
            return  null;
        }
}

package tcc.rules;

import java.sql.SQLException;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;

public class RolesRules {
        public ValueObject selectCoSuperUser() {
            String sql="SELECT * FROM roles "+
                        "WHERE no_name=\'SUPERUSER\'";

            DataBaseHandler handler = new DataBaseHandler();
            try {
                return handler.select(sql)[0];
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return null;
        }

        public ValueObject[] selectAll() {
            String sql = "SELECT * FROM roles WHERE no_name<>\'SUPERUSER\'";

            DataBaseHandler handler = new DataBaseHandler();
            try {
                return handler.select(sql);
```

```
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			return null;
		}

		public ValueObject selectManagerRole() {
			String sql="SELECT * FROM roles "+
					"WHERE no_name=\'Project Manager\'";

			DataBaseHandler handler = new DataBaseHandler();
			try {
				return handler.select(sql)[0];
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			return null;
		}
}

package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.taskhistory.TaskHistoryBroker;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;

public class TaskHistoryRules {
		public ValueObject insert(ValueObject vo) {
			TaskHistoryBroker broker = new TaskHistoryBroker();
```

```
		try {
			return broker.insert(vo);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public ValueObject[] selectTaskHistoryByCoTask(int coTask) {
		String sql = "SELECT * FROM taskhistory WHERE co_task="+coTask;

		DataBaseHandler handler = new DataBaseHandler();
		try {
			return handler.select(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public ValueObject[] selectTaskHistoryByCoReqCoPhase(int coRequirement, int coPhase)
{
		String sql = "SELECT * FROM taskhistory WHERE co_task IN" +
				" (SELECT co_task FROM task WHERE
co_requirement="+coRequirement+") AND" +
				" co_phase="+coPhase;
		DataBaseHandler handler = new DataBaseHandler();
		try {
			return handler.select(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}
```

```
	public ValueObject[] selectTaskHistoryByCoReq(int coRequirement) {
		String sql = "SELECT * FROM taskhistory WHERE co_task IN" +
				" (SELECT co_task FROM task WHERE co_requirement="+coRequirement+")";
		DataBaseHandler handler = new DataBaseHandler();
		try {
			return handler.select(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}
}

package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.task.TaskBroker;
import tcc.persistence.task.TaskPK;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;

public class TaskRules {
	public ValueObject[] selectTaskByCoRequirement(int coRequirement) {
		String sql = "SELECT * FROM task WHERE co_requirement="+coRequirement;

		DataBaseHandler handler = new DataBaseHandler();
		try {
			return handler.select(sql);
		} catch (SQLException e) {
```

```
                e.printStackTrace();
            }
            return null;
        }

        public ValueObject insert(ValueObject vo){
            TaskBroker broker = new TaskBroker();
            try {
                return broker.insert(vo);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return null;
        }

        public ValueObject selectTaskByCoTask(int coTask) {
            TaskBroker broker = new TaskBroker();
            try {
                return broker.findByKey(new TaskPK(coTask));
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return null;
        }
}

package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.taskuser.TaskUserBroker;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;
```

```
public class TaskUserRules {
	public ValueObject[] selectByCoUser(int coUser){
		String sql = "SELECT * FROM taskuser WHERE co_user="+coUser;

		DataBaseHandler handler = new DataBaseHandler();
		try {
			return handler.select(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public void insert(ValueObject vo){
		TaskUserBroker broker = new TaskUserBroker();
		try {
			broker.insert(vo);
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
}

package tcc.rules;

import java.sql.SQLException;

import tcc.persistence.user.UserBroker;
import tcc.persistence.user.UserDefaults;
import tcc.persistence.user.UserPK;

import com.techlab.infrastructure.ejb.ValueObject;
import com.techlab.util.persistence.DataBaseHandler;
```

```
public class UserRules {
	public ValueObject selectUser(ValueObject vo) {
		String sql = "SELECT * FROM [dbo].[user] "+
					"WHERE no_username=\'"+(String)
vo.get(UserDefaults.NOUSERNAME)+"\' "+
					"AND
no_password=\'"+vo.get(UserDefaults.NOPASSWORD)+"\'";

		DataBaseHandler handler = new DataBaseHandler();

		System.out.println("sql := "+sql);

		try {
			ValueObject vos[] = handler.select(sql);
			if(vos.length==0) {
				return null;
			} else {
				return vos[0];
			}
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		return vo;
	}

	public ValueObject[] selectAll() {
		UserBroker broker = new UserBroker();
		try {
			return broker.findAll();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
```

```
		}
		return null;
	}

	public ValueObject insert(ValueObject vo) {
		UserBroker broker = new UserBroker();
		try {
			return broker.insert(vo);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}

	public ValueObject selectUserByCoUser(int coUser){
		String sql = "SELECT * from [dbo].[user] WHERE co_user="+coUser;

		UserPK pk = new UserPK(coUser);
		UserBroker broker = new UserBroker();
		try {
			return broker.findByKey(pk);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return null;
	}
}
```

11.1.3 Módulo Action

```
package tcc.action;

import java.util.List;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.form.RequirementDependencyForm;
import tcc.form.RequirementForm;

public class AddDependencyAction extends Action{
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		RequirementDependencyForm reqForm = (RequirementDependencyForm)form;

		List auxRequirement = (List) req.getSession().getAttribute("auxRequirement");
		List allDependency = (List) req.getSession().getAttribute("allDependency");
		if(reqForm.getCoRequirements()!=null) {
			for(int i=0;i<reqForm.getCoRequirements().length;i++) {
				for(int j=0;j<auxRequirement.size();j++) {
					RequirementForm rForm = (RequirementForm)auxRequirement.get(j);

	if(rForm.getCoRequirement()==reqForm.getCoRequirements()[i]){
							allDependency.add(auxRequirement.remove(j));
							break;
					}
				}
			}
		}

		System.out.println("sizeof allDependency := "+allDependency.size());
		System.out.println("sizeof auxRequirement := "+auxRequirement.size());
```

```
            return mapping.findForward("success");
        }
}

package tcc.action;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.action.utils.ValueObjectToForm;
import tcc.form.ProjectForm;
import tcc.form.ProjectUserForm;
import tcc.form.UserForm;
import tcc.persistence.roles.RolesDefaults;

import com.techlab.infrastructure.ejb.ValueObject;

public class AddTaskAction extends Action{
        public ActionForward execute(ActionMapping mapping, ActionForm form,
                        HttpServletRequest req, HttpServletResponse res) throws Exception {
                ProjectForm                                projectForm                                =
(ProjectForm)req.getSession().getAttribute("currentProject");
```

```
		List allUser = ValueObjectArrayToList.projectUserArrayToList(ActionHelper.selectProjectUserByCoProject(projectForm.getCoProject()));

		ValueObject vo = ActionHelper.selectManagerRole();
		int coManager = ((Integer)vo.get(RolesDefaults.COROLE)).intValue();

		ArrayList allManagers = new ArrayList();
		ArrayList allOther = new ArrayList();

		for(int i=0;i<allUser.size();i++) {
			ProjectUserForm pform = (ProjectUserForm)allUser.get(i);
			UserForm userForm = ValueObjectToForm.userVoToForm(ActionHelper.selectUserByCoUser(pform.getCoUser()));
			if(pform.getCoRoles()==coManager) {
				allManagers.add(userForm);
			} else {
				allOther.add(userForm);
			}
		}

		req.getSession().setAttribute("allOther",allOther);
		req.getSession().setAttribute("allManager",allManagers);

		return mapping.findForward("success");
	}
}
package tcc.action;

import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.FormToValueObject;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.action.utils.ValueObjectToForm;
import tcc.form.RequirementDependencyForm;
import tcc.form.RequirementForm;
import tcc.form.StatusForm;
import tcc.status.StatusStateMachine;

public class ChangeStatusAction extends Action{
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		StatusForm statusForm = (StatusForm)form;
		RequirementForm currentRequirement =
(RequirementForm)req.getSession().getAttribute("currentRequirement");

		if(!statusForm.getNoName().equals("Change status")) {
			String status = statusForm.getNoName();
			if(status.equals(StatusStateMachine.FINALIZED)) {
				List rdForm =
ValueObjectArrayToList.reqDependencyArrayToList(ActionHelper.selectReqDependencyByCoRe
q(currentRequirement.getCoRequirement()));
				boolean allFinalized = verifyAllFinalized(rdForm);
				if(allFinalized) {

	currentRequirement.setNoStatus(StatusStateMachine.FINALIZED);
```

```
ActionHelper.updateRequirement(FormToValueObject.requirementFormToVo(currentRequ
irement));

req.getSession().setAttribute("allStatus",StatusStateMachine.getInstance().nextState(StatusS
tateMachine.FINALIZED));
                                return mapping.findForward("success");
                        } else {
                                PrintWriter writer = res.getWriter();
                                writer.println("Dependancy requirement not finalized");
                                return mapping.findForward("success");
                        }
                } else {
                        currentRequirement.setNoStatus(status);

ActionHelper.updateRequirement(FormToValueObject.requirementFormToVo(currentRequ
irement));

                        ArrayList allStatus = new ArrayList();
                        StatusForm sForm = new StatusForm();
                        sForm.setNoName("Change status");
                        allStatus.add(sForm);

                        List temp = StatusStateMachine.getInstance().nextState(status);

                        for(int i=0;i<temp.size();i++) {

allStatus.add(ActionHelper.StringToStatusForm((String)temp.get(i)));
                        }

                        req.getSession().setAttribute("allStatus",allStatus);
                        return mapping.findForward("success");
                }
        }
```

```
                return mapping.findForward("success");
        }

        private boolean verifyAllFinalized(List rdForm) {
                for(int i=0;i<rdForm.size();i++) {
                        RequirementDependencyForm form =
(RequirementDependencyForm)rdForm.get(i);
                        RequirementForm reqForm =
ValueObjectToForm.requirementVoToForm(ActionHelper.selectRequirementByCoReq(form.getC
oDependantReq()));
                        if(!reqForm.getNoStatus().equals(StatusStateMachine.FINALIZED)) {
                                return false;
                        }
                }
                return true;
        }
}

package tcc.action;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import com.techlab.infrastructure.ejb.ValueObject;
```

```
import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.action.utils.ValueObjectToForm;
import tcc.form.ProjectForm;
import tcc.persistence.projectphases.ProjectPhasesDefaults;

public class GenerateMatrixAction extends Action{
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		ProjectForm currentProject = (ProjectForm)req.getSession().getAttribute("currentProject");
		ValueObject vos[] = ActionHelper.selectProjectPhaseByCoProject(currentProject.getCoProject());

		ArrayList allPhases = new ArrayList();
		for(int i=0;i<vos.length;i++){

	allPhases.add(ValueObjectToForm.phaseVoToForm(ActionHelper.selectPhasesByCoPhases(((Integer)vos[i].get(ProjectPhasesDefaults.COPHASE)).intValue())));
		}

		List allRequirements = ValueObjectArrayToList.requirementArrayToList(ActionHelper.selectAllRequirementByCoProject(currentProject.getCoProject()));

		req.getSession().setAttribute("matrix:allPhases",allPhases);
		req.getSession().setAttribute("matrix:allRequirement",allRequirements);

		return mapping.findForward("success");
	}
}

package tcc.action;
```

```
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.form.ProjectForm;

public class ImportRequirementAction extends Action {
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		ProjectForm currentProject = (ProjectForm)req.getSession().getAttribute("currentProject");
		int coProject = currentProject.getCoProject();

		List allRequirementNotInProject = ValueObjectArrayToList.requirementArrayToList(ActionHelper.selectAllRequirementNotInPtoject(coProject));
		for(int i=0;i<allRequirementNotInProject.size();i++) {
			System.out.println(allRequirementNotInProject.get(i));
		}

	req.getSession().setAttribute("allRequirementNotInProject",allRequirementNotInProject);

		return mapping.findForward("success");
	}
}
```

```
package tcc.action;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectToForm;
import tcc.form.RequirementDependencyForm;
import tcc.form.RequirementForm;

import com.techlab.infrastructure.ejb.ValueObject;

public class InsertDependencyAction extends Action{
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		RequirementForm currentRequirement = (RequirementForm) req.getSession().getAttribute("currentRequirement");
		ValueObject reqDependencies[] = ActionHelper.selectReqDependencyByCoReq(currentRequirement.getCoRequirement());

		ArrayList allDependency = new ArrayList();
		List allRequirement = (List)req.getSession().getAttribute("allRequirement");
		ArrayList auxList = new ArrayList();

		if(reqDependencies.length>0) {
			for(int i=0;i<reqDependencies.length;i++) {
```

```
                    RequirementDependencyForm dependencyForm =
ValueObjectToForm.reqDependencyVoToForm(reqDependencies[i]);
                    for(int j=0;j<allRequirement.size();j++) {
                        RequirementForm requirement =
(RequirementForm)allRequirement.get(j);

        if(dependencyForm.getCoDependantReq()==requirement.getCoRequirement()) {
                            allDependency.add(requirement);
                        }
                    }
                }
            }

            for(int i=0;i<allRequirement.size();i++) {
                RequirementForm requirement = (RequirementForm)allRequirement.get(i);

        if(requirement.getCoRequirement()!=currentRequirement.getCoRequirement()){
                    auxList.add(allRequirement.get(i));
                }
            }
            for(int i=0;i<auxList.size();i++){
                RequirementForm requirement = (RequirementForm)auxList.get(i);
                for(int j=0;j<allDependency.size();j++) {
                    RequirementForm reqForm =
(RequirementForm)allDependency.get(j);
                    if(requirement.getCoRequirement()==reqForm.getCoRequirement() ||

requirement.getCoRequirement()==currentRequirement.getCoRequirement()) {
                      auxList.remove(i);
                      i--;
                      break;
                    }
                }
            }
```

```
                req.getSession().setAttribute("auxRequirement",auxList);
                req.getSession().setAttribute("allDependency",allDependency);

                return mapping.findForward("success");
        }
}

package tcc.action;

import java.io.PrintWriter;
import java.sql.SQLException;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.FormToValueObject;
import tcc.form.ProjectForm;
import tcc.form.RequirementForm;
import tcc.persistence.requirement.RequirementDefaults;
import tcc.rules.ProjectRules;
import tcc.status.StatusStateMachine;

import com.techlab.infrastructure.ejb.ValueObject;

public class InsertNewRequirementAction extends Action {
        public ActionForward execute(ActionMapping mapping, ActionForm form,
```

```
            HttpServletRequest req, HttpServletResponse res) throws Exception {
        RequirementForm reqForm = (RequirementForm) form;

        ProjectForm currentProject = (ProjectForm)req.getSession().getAttribute("currentProject");
        reqForm.setNoVersion("1");
        reqForm.setCoProject(currentProject.getCoProject());
        reqForm.setNoStatus(StatusStateMachine.ANALYSIS);

        ValueObject vo = FormToValueObject.requirementFormToVo(reqForm);
        try{
            List list = (List) req.getSession().getAttribute("allRequirement");
            ActionHelper.insertRequirement(vo);

            int coRequirement = ((Integer)vo.get(RequirementDefaults.COREQUIREMENT)).intValue();
            vo.put(RequirementDefaults.COBACKREQUIREMENT,new Integer(coRequirement));
            vo.put(RequirementDefaults.COINITIALREQUIREMENT, new Integer(coRequirement));
            ActionHelper.updateRequirement(vo);

            reqForm.setCoRequirement(coRequirement);
            reqForm.setCoBackRequirement(coRequirement);
            reqForm.setCoFirstRequirement(coRequirement);

            list.add(list.size(),reqForm);

            ProjectForm pForm = (ProjectForm) req.getSession().getAttribute("currentProject");
            pForm.setTotalHour(pForm.getTotalHour()+reqForm.getTotalHour());
            updateProject(FormToValueObject.projectFormToVo(pForm));

        } catch(SQLException sqlex) {
```

```
                        PrintWriter print = res.getWriter();
                        print.println("Error to insert in a data base...");
                        print.println("Sorry.");
                        print.flush();
                        return mapping.findForward("failed");
                }
                return mapping.findForward("success");
        }

        private void updateProject(ValueObject vo) throws SQLException{
                ProjectRules rules = new ProjectRules();
                rules.update(vo);
        }
}

package tcc.action;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.form.ProjectForm;
import tcc.form.ProjectUserForm;
import tcc.form.RolesForm;
import tcc.form.UserForm;
```

```
import com.techlab.infrastructure.ejb.ValueObject;

public class InsertUserAtProjectAction extends Action {
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {

		ValueObject vos[] = ActionHelper.selectAllRoles();
		List rolesList = ValueObjectArrayToList.roleArrayToList(vos);
		RolesForm roleForm = getManagerRole(rolesList);
		removeManagerRole(rolesList);
		req.getSession().setAttribute("allRoles",rolesList);

		ValueObject allUsers[] = ActionHelper.selectAllUser();
		List list = ValueObjectArrayToList.userArrayToList(allUsers);
		UserForm currentUser = (UserForm)req.getSession().getAttribute("currentUser");

		ProjectForm project = (ProjectForm)req.getSession().getAttribute("currentProject");

		//ValueObject projectuser[] =
ActionHelper.selectProjectUserByCoRoles(roleForm.getCoRoles());
		List projectuser =
ValueObjectArrayToList.projectUserArrayToList(ActionHelper.selectProjectUserByCoProject(proj
ect.getCoProject()));

		removeUser(list,projectuser);
		removeRoot(list);
		req.getSession().setAttribute("allUsers",list);

		return mapping.findForward("addUser");
	}

	//remove others project's manager
	private void removeUser(List allUser, List projectuser) {
		for(int i=0;i<allUser.size();i++) {
```

```
                UserForm userForm = (UserForm)allUser.get(i);
                for(int j=0;j<projectuser.size();j++) {
                        ProjectUserForm pUser = (ProjectUserForm)projectuser.get(j);
                        if(userForm.getCoUser()==pUser.getCoUser()) {
                                allUser.remove(i);
                                i--;
                                break;
                        }
                }
        }
}

private RolesForm getManagerRole(List rolesList) {
        for(int i=0;i<rolesList.size();i++) {
                RolesForm form = (RolesForm)rolesList.get(i);
                if(form.getName().equals("Project Manager")) {
                        return form;
                }
        }
        return null;
}

private void removeManagerRole(List rolesList) {
        for(int i=0;i<rolesList.size();i++) {
                RolesForm form = (RolesForm)rolesList.get(i);
                if(form.getName().equals("Project Manager")) {
                        rolesList.remove(i);
                        break;
                }
        }
}

private void removeRoot(List list) {
        for(int i=0;i<list.size();i++) {
```

```
                        UserForm user = (UserForm) list.get(i);
                        if(user.getName().equals("root")) {
                                list.remove(i);
                                i--;
                        }
                }
        }
}

package tcc.action;

import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.FormToValueObject;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.action.utils.ValueObjectToForm;
import tcc.form.UserForm;
import tcc.persistence.projectuser.ProjectUserDefaults;
import tcc.persistence.roles.RolesDefaults;
import tcc.persistence.user.UserDefaults;

import com.techlab.infrastructure.ejb.ValueObject;
```

```
public class LoginAction extends Action {
	public ActionForward execute(ActionMapping mapping, ActionForm arg1,
			HttpServletRequest req, HttpServletResponse res) throws Exception {

		ValueObject vo = FormToValueObject.userFormToVo((UserForm)arg1);
		ValueObject temp = ActionHelper.selectUser(vo);

		PrintWriter p = res.getWriter();
		if(temp == null || temp.get(UserDefaults.COUSER) == null) {
			p.write("Invalid user or password");
			p.flush();
			return mapping.findForward("invalidPassword");
		} else {

	req.getSession().setAttribute("currentUser",ValueObjectToForm.userVoToForm(temp));


			ValueObject superUser = ActionHelper.selectCoSuperUser();

			ValueObject projectsUserByUser[] =
ActionHelper.selectProjectUserByUser(temp);
			List listOfProjectUser =
ValueObjectArrayToList.projectUserArrayToList(projectsUserByUser);

			if(projectsUserByUser!=null &&

((Integer)projectsUserByUser[0].get(ProjectUserDefaults.COROLES)).intValue() ==
			  ((Integer)superUser.get(RolesDefaults.COROLE)).intValue()) {

				return mapping.findForward("rootSuccess");
			} else {
				ArrayList userProjects = new ArrayList();
				for(int i=0;i<projectsUserByUser.length;i++) {
```

```
        userProjects.add(ValueObjectToForm.projectVoToForm(ActionHelper.selectProjectByCoPr
oject(projectsUserByUser[i])));
                                }
                                req.getSession().setAttribute("projectsUser",listOfProjectUser);
                                req.getSession().setAttribute("userProjects",userProjects);
                                return mapping.findForward("userSuccess");
                        }
                }
        }
}

package tcc.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class NewRequirementAction extends Action {

        public ActionForward execute(ActionMapping mapping, ActionForm form,
                        HttpServletRequest req, HttpServletResponse res) throws Exception {
                return mapping.findForward("success");
        }
}

package tcc.action;

import java.util.List;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.form.ProjectForm;

public class NewVersionRequirementAction extends Action {
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		ProjectForm currentProject =
(ProjectForm)req.getSession().getAttribute("currentProject");
		List list = ValueObjectArrayToList.requirementArrayToList(

	ActionHelper.selectRequirementFinalizedByCoProject(currentProject.getCoProject()));
		req.getSession().setAttribute("allRequirementFinalizedInProject",list);
		return mapping.findForward("success");
	}
}

package tcc.action;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
```

```
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.form.ProjectForm;
import tcc.form.ProjectUserForm;
import tcc.form.UserForm;
import tcc.persistence.roles.RolesDefaults;
import tcc.status.StatusStateMachine;

import com.techlab.infrastructure.ejb.ValueObject;

public class ProjectsAction extends Action {

	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {


		int coProject = Integer.parseInt(req.getParameter("coProject"));

		ArrayList userProjects = (ArrayList)req.getSession().getAttribute("userProjects");
		ArrayList projectUser = (ArrayList)req.getSession().getAttribute("projectsUser");

		String noStatus="";
		noStatus = setCurrentProject(req, coProject, userProjects, noStatus);


		if(noStatus.equals(StatusStateMachine.FINALIZED)) {
			return mapping.findForward("successFinalized");
		} else {
			int coUser =
((UserForm)req.getSession().getAttribute("currentUser")).getCoUser();
```

```
                int coRole = findCoRole(projectUser,coProject,coUser);
                req.getSession().setAttribute("currentRole", new Integer(coRole));

                ValueObject vo = ActionHelper.selectManagerRole();
                int coManager = ((Integer)vo.get(RolesDefaults.COROLE)).intValue();


                List listRequirement = ValueObjectArrayToList.requirementArrayToList(ActionHelper.selectAllRequirementByCoProject(coProject));
                req.getSession().setAttribute("allRequirement",listRequirement);

                ValueObject allDifficulty[] = ActionHelper.selectAllDifficulty();
                List listAllDifficulty = ValueObjectArrayToList.difficultyArrayToList(allDifficulty);
                req.getSession().setAttribute("allDifficulty",listAllDifficulty);

                //ValueObject allStatus[] = ActionHelper.selectAllStatus();
                //List listAllStatus = ValueObjectArrayToList.statusArrayToList(allStatus);
                //req.getSession().setAttribute("allStatus",listAllStatus);

                ProjectUserForm pUser = (ProjectUserForm)findCurrentProjectUser(req,coProject);

                req.getSession().setAttribute("coManager",new Integer(coManager));
                //if(pUser.getCoProject()==coProject && pUser.getCoRoles()==coManager && pUser.getCoUser()==coUser) {
                        return mapping.findForward("success");
                //} else {
            //          return mapping.findForward("successOpenedOther");
                //}
            }
        }
```

```
	private int findCoRole(ArrayList projectUser, int coProject, int coUser) {
		for(int i=0;i<projectUser.size();i++) {
			ProjectUserForm form = (ProjectUserForm)projectUser.get(i);
			if(form.getCoProject()==coProject && form.getCoUser()==coUser) {
				return form.getCoRoles();
			}
		}
		return 0;
	}

	private ActionForm findCurrentProjectUser(HttpServletRequest req, int coProject) {
		List list = (List)req.getSession().getAttribute("projectsUser");

		for(int i=0;i<list.size();i++) {
			ProjectUserForm form = (ProjectUserForm)list.get(i);
			if(form.getCoProject()==coProject) {
				return form;
			}
		}
		return null;
	}

	private String setCurrentProject(HttpServletRequest req, int coProject, ArrayList
userProjects, String noStatus) {
		ProjectForm pForm = null;
		for(int i=0;i<userProjects.size();i++) {
			pForm = (ProjectForm) userProjects.get(i);
			if(pForm.getCoProject()==coProject) {
				req.getSession().setAttribute("currentProject",pForm);
				//ValueObject status =
ActionHelper.selectStatusByCoStatus(pForm.getStatus());
				//noStatus = (String)status.get(StatusDefaults.NONAME);
				req.getSession().setAttribute("projectStatus",pForm.getStatus());
```

```
                                break;
                        }
                }
                return noStatus;
        }
}

package tcc.action;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.form.RequirementDependencyForm;
import tcc.form.RequirementForm;

public class RemoveDependencyAction extends Action{
        public ActionForward execute(ActionMapping mapping, ActionForm form,
                        HttpServletRequest req, HttpServletResponse res) throws Exception {
                RequirementDependencyForm reqForm = (RequirementDependencyForm)form;


                List auxRequirement = (List) req.getSession().getAttribute("auxRequirement");
                List allDependency = (List) req.getSession().getAttribute("allDependency");

                System.out.println("sizeof allDependency := "+allDependency.size());
                System.out.println("sizeof auxRequirement := "+auxRequirement.size());
```

```
                if(reqForm.getCoRequirements()!=null) {
                        for(int i=0;i<reqForm.getCoRequirements().length;i++) {
                                for(int j=0;j<allDependency.size();j++) {
                                        RequirementForm rForm =
(RequirementForm)allDependency.get(j);

        if(rForm.getCoRequirement()==reqForm.getCoRequirements()[i]){
                                                auxRequirement.add(allDependency.remove(j));
                                                break;
                                        }
                                }
                        }
                }
                return mapping.findForward("success");
        }
}

package tcc.action;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectToForm;
import tcc.form.RequirementForm;
import tcc.form.StatusForm;
```

```
import tcc.form.TaskForm;
import tcc.form.UserForm;
import tcc.persistence.taskuser.TaskUserDefaults;
import tcc.status.StatusStateMachine;

import com.techlab.infrastructure.ejb.ValueObject;

public class RequirementSelectedAction extends Action {
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		int coRequirement = Integer.parseInt(req.getParameter("coRequirement"));
		List allRequirement = (List) req.getSession().getAttribute("allRequirement");

		RequirementForm formRequirement=null;
		for(int i=0;i<allRequirement.size();i++) {
			RequirementForm auxForm = (RequirementForm) allRequirement.get(i);
			if(auxForm.getCoRequirement()==coRequirement) {
				formRequirement = auxForm;
				//allRequirement.remove(i);
				break;
			}
		}
		req.getSession().setAttribute("currentRequirement",formRequirement);

		ArrayList allStatus = new ArrayList();

		StatusForm test1= new StatusForm();
		test1.setNoName("Change status");
		allStatus.add(test1);

		System.out.println("requirement status ?:= "+formRequirement.getNoStatus());

		List temp =
StatusStateMachine.getInstance().nextState(formRequirement.getNoStatus());
```

```
		for(int i=0;i<temp.size();i++) {
			allStatus.add(ActionHelper.StringToStatusForm((String)temp.get(i)));
		}
		StatusForm test= new StatusForm();

		req.getSession().setAttribute("allStatus",allStatus);

		ValueObject tasks[] = ActionHelper.selectTaskByCoRequirement(coRequirement);

		UserForm currentUser = (UserForm)req.getSession().getAttribute("currentUser");
		ValueObject taskuser[] = ActionHelper.selectTaskUserByCoUser(currentUser.getCoUser());

		ArrayList allTask = new ArrayList();

		for(int i=0;i<taskuser.length;i++) {
			for(int j=0;j<tasks.length;j++) {
				TaskForm taskForm = ValueObjectToForm.taskVoToForm(tasks[j]);

	if(taskForm.getCoTask()==((Integer)taskuser[i].get(TaskUserDefaults.COTASK)).intValue(
)) {
					allTask.add(taskForm);
				}
			}
		}

		req.getSession().setAttribute("allTask",allTask);

		return mapping.findForward("success");
	}
}

package tcc.action;
```

```
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;

public class RequirementViewHistoryAction extends Action{
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		int coRequirement = Integer.parseInt(req.getParameter("coRequirement"));
		List requirementHistory =
ValueObjectArrayToList.taskArrayToList(ActionHelper.selectTaskByCoRequirement(coRequirem
ent));
		req.getSession().setAttribute("requirementHistory",requirementHistory);

		return mapping.findForward("success");
	}
}

package tcc.action;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.form.ProjectForm;
import tcc.form.RequirementForm;
import tcc.persistence.requirement.RequirementDefaults;
import tcc.persistence.requirementdependency.RequirementDependencyDefaults;

import com.techlab.infrastructure.ejb.ValueObject;

public class SubmitImportRequirementAction extends Action {
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		RequirementForm requirementForm = (RequirementForm)form;
		ProjectForm currentProject =
(ProjectForm)req.getSession().getAttribute("currentProject");

		for(int i=0; i<requirementForm.getCoRequirements().length;i++) {

			ValueObject requirement =
ActionHelper.selectRequirementByCoReq(requirementForm.getCoRequirements()[i]);
			requirement.put(RequirementDefaults.COREQUIREMENT,null);
			requirement.put(RequirementDefaults.COPROJECT,new
Integer(currentProject.getCoProject()));
			requirement.put(RequirementDefaults.CORELATEDTO,new
Integer(requirementForm.getCoRequirements()[i]));

			ActionHelper.insertRequirement(requirement);
```

```
                int coRequirement =
((Integer)requirement.get(RequirementDefaults.COREQUIREMENT)).intValue();
                int initialRequirement =
((Integer)requirement.get(RequirementDefaults.COINITIALREQUIREMENT)).intValue();

                if(requirementForm.getCoRequirements()[i]==initialRequirement) {
                    requirement.put(RequirementDefaults.COINITIALREQUIREMENT,
new Integer(coRequirement));
                    requirement.put(RequirementDefaults.COBACKREQUIREMENT,
new Integer(coRequirement));
                }
                ActionHelper.updateRequirement(requirement);

                ArrayList list = new ArrayList();

        findAllRequirementDependency(list,requirementForm.getCoRequirements()[i]);
                for(int j=0;j<list.size();j++) {
                    ValueObject temp = (ValueObject)list.get(j);

                    ValueObject aux1 =
returnRequirementByCoRequirement(currentProject.getCoProject(),

((Integer)temp.get(RequirementDependencyDefaults.COREQUIREMENT)).intValue());
                    ValueObject aux2 =
returnRequirementByCoRequirement(currentProject.getCoProject(),

((Integer)temp.get(RequirementDependencyDefaults.CODEPENDANTREQ)).intValue());

                    ValueObject reqDependency = new ValueObject();

        reqDependency.put(RequirementDependencyDefaults.COREQUIREMENT,
                                                    new
Integer(((Integer)aux1.get(RequirementDefaults.COREQUIREMENT)).intValue()));
```

```
        reqDependency.put(RequirementDependencyDefaults.CODEPENDANTREQ,
                                                        new
Integer(((Integer)aux2.get(RequirementDefaults.COREQUIREMENT)).intValue()));
                        ActionHelper.insertReqDependency(reqDependency);
                }
        }
        List                    listRequirement                 =
ValueObjectArrayToList.requirementArrayToList(ActionHelper.selectAllRequirementByCoProject
(currentProject.getCoProject()));
        req.getSession().setAttribute("allRequirement",listRequirement);

        return mapping.findForward("success");
    }

    private ValueObject returnRequirementByCoRequirement(int coProject, int
coRequirement){
        ValueObject                 aux[]                   =
ActionHelper.selectRequirementByCoProjectCoRelatedTo(coProject,coRequirement);
        ValueObject __aux;
        if(aux.length==0) {
                __aux = ActionHelper.selectRequirementByCoReq(coRequirement);
                __aux.put(RequirementDefaults.COREQUIREMENT,null);
                __aux.put(RequirementDefaults.COPROJECT,new Integer(coProject));
                __aux.put(RequirementDefaults.CORELATEDTO,new
Integer(coRequirement));
                try {
                        ActionHelper.insertRequirement(__aux);

                        int                     coReq                   =
((Integer)__aux.get(RequirementDefaults.COREQUIREMENT)).intValue();
                        int                 initialRequirement          =
((Integer)__aux.get(RequirementDefaults.COINITIALREQUIREMENT)).intValue();
```

```
                                if(coRequirement==initialRequirement) {

        __aux.put(RequirementDefaults.COINITIALREQUIREMENT, new Integer(coReq));
                                        __aux.put(RequirementDefaults.COBACKREQUIREMENT,
new Integer(coReq));
                                }
                                ActionHelper.updateRequirement(__aux);
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                } else {
                        __aux = aux[0];
                }
                return __aux;
        }

        private void findAllRequirementDependency(ArrayList list, int coRequirement) {
                ValueObject vos[] = ActionHelper.selectReqDependencyByCoReq(coRequirement);

                for(int i=0;i<vos.length;i++) {
                        list.add(vos[i]);
                        System.out.println("requirement := "+vos[i]);

        findAllRequirementDependency(list,((Integer)vos[i].get(RequirementDependencyDefaults.
CODEPENDANTREQ)).intValue());
                }
        }
}

package tcc.action;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import com.techlab.infrastructure.ejb.ValueObject;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.form.ProjectForm;
import tcc.form.RequirementForm;
import tcc.persistence.requirement.RequirementDefaults;
import tcc.persistence.requirementdependency.RequirementDependencyDefaults;
import tcc.status.StatusStateMachine;

public class SubmitNewVersionRequirementAction extends Action {
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {

		RequirementForm requirementForm = (RequirementForm)form;
		ProjectForm currentProject =
(ProjectForm)req.getSession().getAttribute("currentProject");

		ValueObject requirement =
ActionHelper.selectRequirementByCoReq(requirementForm.getCoRequirements()[0]);
		ValueObject vos[] =
ActionHelper.selectReqDependencyByCoReq(requirementForm.getCoRequirements()[0]);

		requirement.put(RequirementDefaults.COREQUIREMENT,null);
		int version =
Integer.parseInt((String)requirement.get(RequirementDefaults.NOVERSION)) + 1;
		requirement.put(RequirementDefaults.NOVERSION, ""+version);
```

```
                requirement.put(RequirementDefaults.NOSTATUS,
StatusStateMachine.ANALYSIS);
                requirement.put(RequirementDefaults.NUHOURDONE, new Float(0));

                ActionHelper.insertRequirement(requirement);
                int                                    coRequirement                                    =
((Integer)requirement.get(RequirementDefaults.COREQUIREMENT)).intValue();
                int                                    initialRequirement                               =
((Integer)requirement.get(RequirementDefaults.COINITIALREQUIREMENT)).intValue();

                if(requirementForm.getCoRequirements()[0]==initialRequirement) {
                        //requirement.put(RequirementDefaults.COINITIALREQUIREMENT,    new
Integer(requirementForm.getCoRequirements()[0]));
                        requirement.put(RequirementDefaults.COBACKREQUIREMENT,         new
Integer(requirementForm.getCoRequirements()[0]));
                }
                ActionHelper.updateRequirement(requirement);

                for(int i=0;i<vos.length;i++) {
                        vos[i].put(RequirementDependencyDefaults.COREQDEPENDENCE,null);
                        vos[i].put(RequirementDependencyDefaults.COREQUIREMENT,new
Integer(coRequirement));
                        ActionHelper.insertReqDependency(vos[i]);
                }

                List                                   listRequirement                                  =
ValueObjectArrayToList.requirementArrayToList(ActionHelper.selectAllRequirementByCoProject
(currentProject.getCoProject()));
                req.getSession().setAttribute("allRequirement",listRequirement);

                return mapping.findForward("success");
        }
}
```

```
package tcc.action;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.FormToValueObject;
import tcc.action.utils.ValueObjectToForm;
import tcc.form.RequirementForm;
import tcc.form.TaskForm;
import tcc.form.UserForm;
import tcc.persistence.task.TaskDefaults;
import tcc.persistence.taskuser.TaskUserDefaults;

import com.techlab.infrastructure.ejb.ValueObject;

public class SubmitTaskAction extends Action{
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		TaskForm taskForm = (TaskForm)form;
		RequirementForm currentReq =
(RequirementForm)req.getSession().getAttribute("currentRequirement");
		taskForm.setCoRequirement(currentReq.getCoRequirement());

		ValueObject vo = FormToValueObject.taskFormToVo(taskForm);
		ActionHelper.insertTask(vo);
		int coTask = ((Integer)vo.get(TaskDefaults.COTASK)).intValue();
```

```
                List allManager = (List)req.getSession().getAttribute("allManager");
                List allOther = (List)req.getSession().getAttribute("allOther");
                for(int i=0;i<allManager.size();i++) {
                        UserForm user = (UserForm)allManager.get(i);
                        ValueObject v = new ValueObject();

                        v.put(TaskUserDefaults.COTASK, new Integer(coTask));
                        v.put(TaskUserDefaults.COUSER, new Integer(user.getCoUser()));

                        ActionHelper.insertTaskUser(v);
                }

                if(taskForm.getCoUsers()!=null) {
                        for(int i=0;i<taskForm.getCoUsers().length;i++) {
                                ValueObject v = new ValueObject();

                                v.put(TaskUserDefaults.COTASK, new Integer(coTask));
                                v.put(TaskUserDefaults.COUSER,                                  new
Integer(taskForm.getCoUsers()[i]));

                                ActionHelper.insertTaskUser(v);
                        }
                }

                List allTask = (List)req.getSession().getAttribute("allTask");
                allTask.add(ValueObjectToForm.taskVoToForm(vo));

                return mapping.findForward("success");
        }
}

package tcc.action;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.FormToValueObject;
import tcc.form.ProjectForm;
import tcc.form.RequirementForm;
import tcc.form.TaskForm;
import tcc.form.TaskHistoryForm;
import tcc.form.UserForm;

public class SubmitTaskHistoryAction extends Action{
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		TaskHistoryForm taskHistory = (TaskHistoryForm)form;

		UserForm currentUser = (UserForm)req.getSession().getAttribute("currentUser");
		TaskForm currentTask = (TaskForm)req.getSession().getAttribute("currentTask");
		taskHistory.setCoUser(currentUser.getCoUser());
		taskHistory.setCoTask(currentTask.getCoTask());


	ActionHelper.insertTaskHistory(FormToValueObject.taskHistoryFormToVo(taskHistory));

		ProjectForm currentProject =
(ProjectForm)req.getSession().getAttribute("currentProject");
		RequirementForm currentReq=
(RequirementForm)req.getSession().getAttribute("currentRequirement");
```

```
		currentReq.setHourDone(currentReq.getHourDone()+taskHistory.getNuHourDone());

		currentProject.setHourDone(currentProject.getHourDone()+taskHistory.getNuHourDone());

			if(currentReq.getHourDone()>currentReq.getTotalHour()) {
				currentReq.setTotalHour(currentReq.getHourDone());

		currentProject.setTotalHour(currentProject.getTotalHour()+taskHistory.getNuHourDone());
			}

			if(currentProject.getHourDone()>currentProject.getTotalHour()) {
				currentProject.setTotalHour(currentProject.getHourDone());
			}


		ActionHelper.updateProject(FormToValueObject.projectFormToVo(currentProject));

		ActionHelper.updateRequirement(FormToValueObject.requirementFormToVo(currentReq)
);

			return mapping.findForward("success");
		}
}

package tcc.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
```

```
import tcc.action.helper.ActionHelper;
import tcc.action.utils.FormToValueObject;
import tcc.form.ProjectForm;
import tcc.form.ProjectUserForm;

import com.techlab.infrastructure.ejb.ValueObject;

public class SubmitUserAtProjectAction extends Action {
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {

		ProjectForm projectForm =
(ProjectForm)req.getSession().getAttribute("currentProject");

		ProjectUserForm puForm = (ProjectUserForm)form;
		puForm.setCoProject(projectForm.getCoProject());

		ValueObject vo = FormToValueObject.projectUserFormToVo(puForm);
		ActionHelper.insertProjectUser(vo);

		return mapping.findForward("success");
	}
}

package tcc.action;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
```

```
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;
import tcc.action.utils.ValueObjectToForm;
import tcc.form.ProjectForm;
import tcc.form.ProjectPhasesForm;
import tcc.form.TaskForm;

public class TaskSelectedAction extends Action{
	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest req, HttpServletResponse res) throws Exception {
		List allArtifacts = ValueObjectArrayToList.artifactsArrayToList(ActionHelper.selectAllArtifacts());

		ProjectForm projectForm = (ProjectForm)req.getSession().getAttribute("currentProject");
		List projectPhases = ValueObjectArrayToList.projectPhasesArrayToList(ActionHelper.selectProjectPhaseByCoProject(projectForm.getCoProject()));

		ArrayList allPhases = new ArrayList();
		for(int i=0;i<projectPhases.size();i++) {
			ProjectPhasesForm ppForm = (ProjectPhasesForm)projectPhases.get(i);

	allPhases.add(ValueObjectToForm.phaseVoToForm(ActionHelper.selectPhasesByCoPhases(ppForm.getCoPhase())));
		}

		int coTask = Integer.parseInt((String)req.getParameter("coTask"));
		List allTask = (List)req.getSession().getAttribute("allTask");
		for(int i=0;i<allTask.size();i++) {
```

```
                        TaskForm task = (TaskForm)allTask.get(i);
                        if(task.getCoTask()==coTask) {
                                req.getSession().setAttribute("currentTask",task);
                        }
                }

                req.getSession().setAttribute("allArtifact",allArtifacts);
                req.getSession().setAttribute("allPhase",allPhases);

                return mapping.findForward("success");
        }
}

package tcc.action;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import tcc.action.helper.ActionHelper;
import tcc.action.utils.ValueObjectArrayToList;

public class TaskViewHistoryAction extends Action {
        public ActionForward execute(ActionMapping mapping, ActionForm form,
                        HttpServletRequest req, HttpServletResponse res) throws Exception {
                int coTask = Integer.parseInt(req.getParameter("coTask"));
```

```
            List taskHistory = ValueObjectArrayToList.taskHistoryArrayToList(ActionHelper.selectTaskHistoryByCoTask(coTask));
            List allUser = ValueObjectArrayToList.userArrayToList(ActionHelper.selectAllUser());
            List allArtifact = ValueObjectArrayToList.artifactsArrayToList(ActionHelper.selectAllArtifacts());
            List allPhase = ValueObjectArrayToList.phaseArrayToList(ActionHelper.selectAllPhases());

            req.getSession().setAttribute("allTaskHistory",taskHistory);
            req.getSession().setAttribute("allUser",allUser);
            req.getSession().setAttribute("allArtifact",allArtifact);
            req.getSession().setAttribute("allPhase",allPhase);

            return mapping.findForward("success");
        }
}
```

11.1.4 Clases utilitárias

```
package tcc.action.helper;

import java.sql.SQLException;

import tcc.form.ProjectForm;
import tcc.form.StatusForm;
import tcc.rules.ArtifactRules;
import tcc.rules.DifficultyRules;
import tcc.rules.PhasesRules;
import tcc.rules.ProjectPhasesRules;
import tcc.rules.ProjectRules;
import tcc.rules.ProjectUserRules;
import tcc.rules.RequirementDependencyRules;
```

```
import tcc.rules.RequirementRules;
import tcc.rules.RolesRules;
import tcc.rules.TaskHistoryRules;
import tcc.rules.TaskRules;
import tcc.rules.TaskUserRules;
import tcc.rules.UserRules;

import com.techlab.infrastructure.ejb.ValueObject;

public class ActionHelper {
	public static ValueObject[] selectAllArtifacts() {
		ArtifactRules rules = new ArtifactRules();
		return rules.selectAll();
	}

	public static ValueObject selectProjectByCoProject(ValueObject vo) {
		ProjectRules rules = new ProjectRules();
		return rules.selectProjectByKey(vo);
	}

	public static ValueObject selectCoSuperUser() {
		RolesRules rules = new RolesRules();
		return rules.selectCoSuperUser();
	}

	public static ValueObject[] selectProjectUserByUser(ValueObject vo) {
		ProjectUserRules rules = new ProjectUserRules();
		return rules.selectProjectUserByUser(vo);
	}

	public static ValueObject selectUser(ValueObject vo) {
		UserRules user = new UserRules();
		return user.selectUser(vo);
	}
```

```
/*public static ValueObject[] selectAllStatus() {
        StatusRules rules = new StatusRules();
        return rules.selectAll();
}*/

public static ValueObject[] selectAllDifficulty(){
        DifficultyRules rules = new DifficultyRules();
        return rules.selectAll();
}

public static ValueObject[] selectAllRequirementByCoProject(int coProject) {
        RequirementRules rules = new RequirementRules();

        ProjectForm form = new ProjectForm();
        form.setCoProject(coProject);

        return rules.selectRequirementByCoProject(form);
}

public static ValueObject selectManagerRole() {
        RolesRules rules = new RolesRules();
        return rules.selectManagerRole();
}

/*public static ValueObject selectStatusByCoStatus(int status) {
        StatusRules rules = new StatusRules();
        return rules.selectStatusByCoStatus(status);
}*/

public static ValueObject insertProjectUser(ValueObject vo) {
        ProjectUserRules rules = new ProjectUserRules();
        return rules.insert(vo);
}
```

```
public static ValueObject[] selectAllRoles() {
        RolesRules rules = new RolesRules();
        return rules.selectAll();
}

public static ValueObject[] selectAllUser() {
        UserRules rules = new UserRules();
        return rules.selectAll();
}

public static ValueObject[] selectAllPhases() {
        PhasesRules rules = new PhasesRules();
        return rules.selectAll();
}

/*public static ValueObject selectStatusOpen() {
        StatusRules rules = new StatusRules();
        return rules.selectStatusOpen();
}*/

public static ValueObject insertProject(ValueObject project) {
        ProjectRules rules = new ProjectRules();
        return rules.insert(project);
}

public static ValueObject insertUser(ValueObject vo) {
        UserRules rules = new UserRules();
        return rules.insert(vo);
}

public static ValueObject insertProjectPhase(ValueObject vo) {
        ProjectPhasesRules rules = new ProjectPhasesRules();
        return rules.insert(vo);
```

```
}

public static ValueObject[] selectProjectUserByCoRoles(int coRoles) {
        ProjectUserRules rules = new ProjectUserRules();
        return rules.selectProjectUserByCoRoles(coRoles);
}

public static ValueObject[] selectTaskByCoRequirement(int coRequirement) {
        TaskRules rules = new TaskRules();
        return rules.selectTaskByCoRequirement(coRequirement);
}

public static ValueObject[] selectReqDependencyByCoReq(int coRequirement){
        RequirementDependencyRules rules = new RequirementDependencyRules();
        return rules.selectReqDependencyByCoReq(coRequirement);
}

public static void removeReqDependencyByCoReq(int coRequirement){
        RequirementDependencyRules rules = new RequirementDependencyRules();
        rules.removeAllByCoReq(coRequirement);
}

public static void insertReqDependency(ValueObject vo) {
        RequirementDependencyRules rules = new RequirementDependencyRules();
        rules.insert(vo);
}

public static ValueObject[] selectTaskUserByCoUser(int coUser) {
        TaskUserRules rules = new TaskUserRules();
        return rules.selectByCoUser(coUser);
}

public static ValueObject[] selectProjectUserByCoProject(int coProject){
        ProjectUserRules rules = new ProjectUserRules();
```

```
        return rules.selectProjectUserByCoProject(coProject);
}

public static ValueObject selectUserByCoUser(int coUser){
        UserRules rules = new UserRules();
        return rules.selectUserByCoUser(coUser);
}

public static ValueObject insertTask(ValueObject vo){
        TaskRules rules = new TaskRules();
        return rules.insert(vo);
}

public static void insertTaskUser(ValueObject vo) {
        TaskUserRules rules = new TaskUserRules();
        rules.insert(vo);
}

public static ValueObject[] selectProjectPhaseByCoProject(int coProject) {
        ProjectPhasesRules rules = new ProjectPhasesRules();
        return rules.selectProjectPhaseByCoProject(coProject);
}

public static ValueObject selectArtifactByCoArtifact(int coArtifact){
        ArtifactRules rules = new ArtifactRules();
        return rules.selectArtifactByCoArtifact(coArtifact);
}

public static ValueObject selectPhasesByCoPhases(int coPhases) {
        PhasesRules rules = new PhasesRules();
        return rules.selectPhaseByCoPhase(coPhases);
}

public static ValueObject insertTaskHistory(ValueObject vo) {
```

```
        TaskHistoryRules rules = new TaskHistoryRules();
        return rules.insert(vo);
}

public static void updateRequirement(ValueObject vo) {
        RequirementRules rules = new RequirementRules();
        rules.update(vo);
}

public static void updateProject(ValueObject vo) {
        ProjectRules rules = new ProjectRules();
        try {
                rules.update(vo);
        } catch (SQLException e) {
                e.printStackTrace();
        }
}

public static StatusForm StringToStatusForm(String name) {
        StatusForm form = new StatusForm();
        form.setNoName(name);
        return form;
}

public static ValueObject selectRequirementByCoReq(int coReq) {
        RequirementRules rules = new RequirementRules();
        return rules.selectRequirementByCoReq(coReq);
}

public static ValueObject[] selectTaskHistoryByCoTask(int coTask) {
        TaskHistoryRules rules = new TaskHistoryRules();
        return rules.selectTaskHistoryByCoTask(coTask);
}
```

```
	public static ValueObject[] selectAllRequirementNotInPtoject(int coProject) {
		RequirementRules rules = new RequirementRules();
		return rules.selectAllRequirementNotInProject(coProject);
	}

	public static ValueObject insertRequirement(ValueObject vo) throws SQLException {
		RequirementRules rules = new RequirementRules();
		return rules.insert(vo);
	}

	public static ValueObject[] selectRequirementByCoProjectCoRelatedTo(int coProject, int coRequirement) {
		RequirementRules rules = new RequirementRules();
		return rules.selectRequirementByCoProjectCoRelatedTo(coProject,coRequirement);
	}

	public static ValueObject[] selectRequirementFinalizedByCoProject(int coProject) {
		RequirementRules rules = new RequirementRules();
		return rules.selectRequirementFinalizedByCoProject(coProject);
	}

	public static ValueObject[] selectTaskHistoryByCoReqCoPhase(int coRequirement, int coPhase) {
		TaskHistoryRules rules = new TaskHistoryRules();
		return rules.selectTaskHistoryByCoReqCoPhase(coRequirement,coPhase);
	}

	public static ValueObject[] selectTaskHistoryByCoReq(int coRequirement) {
		TaskHistoryRules rules = new TaskHistoryRules();
		return rules.selectTaskHistoryByCoReq(coRequirement);
	}

	public static ValueObject selectTaskByCoTask(int coTask) {
		TaskRules rules = new TaskRules();
```

```
                return rules.selectTaskByCoTask(coTask);
        }
}

package tcc.action.utils;

import org.apache.struts.action.ActionForm;

import tcc.form.ProjectForm;
import tcc.form.ProjectPhasesForm;
import tcc.form.ProjectUserForm;
import tcc.form.RequirementForm;
import tcc.form.TaskForm;
import tcc.form.TaskHistoryForm;
import tcc.form.UserForm;
import tcc.persistence.project.ProjectDefaults;
import tcc.persistence.projectphases.ProjectPhasesDefaults;
import tcc.persistence.projectuser.ProjectUserDefaults;
import tcc.persistence.requirement.RequirementDefaults;
import tcc.persistence.task.TaskDefaults;
import tcc.persistence.taskhistory.TaskHistoryDefaults;
import tcc.persistence.user.UserDefaults;

import com.techlab.infrastructure.ejb.ValueObject;

public class FormToValueObject {
        public static ValueObject userFormToVo(ActionForm form) {
                UserForm user = (UserForm) form;
                ValueObject vo = new ValueObject();

                vo.put(UserDefaults.COUSER,new Integer(user.getCoUser()));
                vo.put(UserDefaults.NOADDRESS,user.getAddress());
                vo.put(UserDefaults.NOEMAIL,user.getEmail());
                vo.put(UserDefaults.NONAME,user.getName());
```

```
        vo.put(UserDefaults.NOPASSWORD,user.getPassword());
        vo.put(UserDefaults.NOUSERNAME,user.getUsername());

        return vo;
}

public static ValueObject projectUserFormToVo(ActionForm form) {
        ProjectUserForm puForm = (ProjectUserForm)form;
        ValueObject vo = new ValueObject();

        vo.put(ProjectUserDefaults.COPROJECT,new Integer(puForm.getCoProject()));
        vo.put(ProjectUserDefaults.COROLES,new Integer(puForm.getCoRoles()));
        vo.put(ProjectUserDefaults.COUSER,new Integer(puForm.getCoUser()));

        return vo;
}

public static ValueObject projectFormToVo(ActionForm form) {
        ProjectForm project = (ProjectForm)form;
        ValueObject vo = new ValueObject();

        vo.put(ProjectDefaults.NONAME,project.getName());
        vo.put(ProjectDefaults.NOVERSION,project.getVersion());
        vo.put(ProjectDefaults.NUTOTALHOUR,new Float(project.getTotalHour()));
        vo.put(ProjectDefaults.NUHOURDONE,new Float(project.getHourDone()));
        vo.put(ProjectDefaults.NUPHASES,new Integer(project.getPhases()));
        vo.put(ProjectDefaults.NOSTATUS, project.getStatus());
        vo.put(ProjectDefaults.COPROJECT, new Integer(project.getCoProject()));

        return vo;
}

public static ValueObject projectPhaseFormToVo(ActionForm form){
        ProjectPhasesForm pPhases = (ProjectPhasesForm)form;
```

```
		ValueObject vo = new ValueObject();

		vo.put(ProjectPhasesDefaults.COPHASE, new Integer(pPhases.getCoPhase()));
		vo.put(ProjectPhasesDefaults.COPROJECT, new Integer(pPhases.getCoProject()));
		vo.put(ProjectPhasesDefaults.COPROJECTPHASES, new
Integer(pPhases.getCoProjectPhase()));

		return vo;
	}

	public static ValueObject requirementFormToVo(ActionForm form) {
		RequirementForm reqForm = (RequirementForm)form;
		ValueObject vo = new ValueObject();

		vo.put(RequirementDefaults.CODIFICULTTY,new
Integer(reqForm.getCoDifficulty()));
		vo.put(RequirementDefaults.COREQUIREMENT,new
Integer(reqForm.getCoRequirement()));
		vo.put(RequirementDefaults.NOSTATUS,reqForm.getNoStatus());
		vo.put(RequirementDefaults.NOCOMMENTARY,reqForm.getNoCommentary());
		vo.put(RequirementDefaults.NONAME,reqForm.getName());
		vo.put(RequirementDefaults.NOVERSION,reqForm.getNoVersion());
		vo.put(RequirementDefaults.NUHOURDONE,new Float(reqForm.getHourDone()));
		vo.put(RequirementDefaults.NUTOTALHOUR,new
Float(reqForm.getTotalHour()));
		vo.put(RequirementDefaults.COINITIALREQUIREMENT, new
Integer(reqForm.getCoFirstRequirement()));
		vo.put(RequirementDefaults.COBACKREQUIREMENT, new
Integer(reqForm.getCoBackRequirement()));
		vo.put(RequirementDefaults.COPROJECT, new Integer(reqForm.getCoProject()));
		vo.put(RequirementDefaults.CORELATEDTO, new
Integer(reqForm.getCoRelatedTo()));

		return vo;
```

```
	}

	public static ValueObject taskFormToVo(TaskForm form) {
		ValueObject vo = new ValueObject();
		vo.put(TaskDefaults.COREQUIREMENT,new Integer(form.getCoRequirement()));
		vo.put(TaskDefaults.NOTASK,form.getNoTask());

		return vo;
	}

	public static ValueObject taskHistoryFormToVo(TaskHistoryForm form){
		ValueObject vo = new ValueObject();

		vo.put(TaskHistoryDefaults.COARTIFACT, new Integer(form.getCoArtifact()));
		vo.put(TaskHistoryDefaults.COPHASE, new Integer(form.getCoPhase()));
		vo.put(TaskHistoryDefaults.COTASK, new Integer(form.getCoTask()));
		vo.put(TaskHistoryDefaults.COTASKHISTORY, new
Integer(form.getCoTaskHistory()));
		vo.put(TaskHistoryDefaults.COUSER, new Integer(form.getCoUser()));
		vo.put(TaskHistoryDefaults.NUHOURDONE, new Float(form.getNuHourDone()));

		return vo;
	}
}

package tcc.action.utils;

import java.util.ArrayList;
import java.util.List;

import com.techlab.infrastructure.ejb.ValueObject;

public class ValueObjectArrayToList {
	public static List artifactsArrayToList(ValueObject[] artifacts) {
```

```
		ArrayList list = new ArrayList();

		for(int i=0;i<artifacts.length;i++) {
			list.add(ValueObjectToForm.artifactVoToForm(artifacts[i]));
		}

		return list;
	}

	public static List projectUserArrayToList(ValueObject[] projectsUserByUser) {
		ArrayList list = new ArrayList();

		for(int i=0;i<projectsUserByUser.length;i++) {

	list.add(ValueObjectToForm.projectUserVoToForm(projectsUserByUser[i]));
		}

		return list;
	}

	/*public static List statusArrayToList(ValueObject[] allStatus) {
		ArrayList list = new ArrayList();

		for(int i=0;i<allStatus.length;i++) {
			list.add(ValueObjectToForm.statusVoToForm(allStatus[i]));
		}

		return list;
	}*/

	public static List difficultyArrayToList(ValueObject[] allDifficulty) {
		ArrayList list = new ArrayList();

		for(int i=0;i<allDifficulty.length;i++) {
```

```
                list.add(ValueObjectToForm.difficultyVoToForm(allDifficulty[i]));
        }

        return list;
}

public static List requirementArrayToList(ValueObject[] requirements) {
        ArrayList list = new ArrayList();

        if(requirements!=null) {
                for(int i=0;i<requirements.length;i++) {

list.add(ValueObjectToForm.requirementVoToForm(requirements[i]));
                }
        }

        return list;
}

public static List roleArrayToList(ValueObject[] vos) {
        ArrayList list = new ArrayList();

        for(int i=0;i<vos.length;i++) {
                list.add(ValueObjectToForm.roleVoToForm(vos[i]));
        }

        return list;
}

public static List userArrayToList(ValueObject[] allUsers) {
        ArrayList list = new ArrayList();

        for(int i=0;i<allUsers.length;i++) {
                list.add(ValueObjectToForm.userVoToForm(allUsers[i]));
```

```
        }

        return list;
}

public static List phaseArrayToList(ValueObject[] allPhases) {
        ArrayList list = new ArrayList();

        for(int i=0;i<allPhases.length;i++) {
                list.add(ValueObjectToForm.phaseVoToForm(allPhases[i]));
        }

        return list;
}

public static List taskArrayToList(ValueObject[] allTasks) {
        ArrayList list = new ArrayList();

        for(int i=0;i<allTasks.length;i++) {
                list.add(ValueObjectToForm.taskVoToForm(allTasks[i]));
        }

        return list;
}

public static ArrayList reqDependencyArrayToList(ValueObject[] allDependencies) {
        ArrayList list = new ArrayList();

        for(int i=0;i<allDependencies.length;i++) {

list.add(ValueObjectToForm.reqDependencyVoToForm(allDependencies[i]));
        }

        return list;
```

```
	}

	public static ArrayList projectPhasesArrayToList(ValueObject[] projectPhases){
		ArrayList list = new ArrayList();

		for(int i=0;i<projectPhases.length;i++) {
			list.add(ValueObjectToForm.projectPhaseVoToForm(projectPhases[i]));
		}

		return list;
	}

	public static ArrayList taskHistoryArrayToList(ValueObject[] taskHistories) {
		ArrayList list = new ArrayList();

		for(int i=0;i<taskHistories.length;i++) {
			list.add(ValueObjectToForm.taskHistoryVoToForm(taskHistories[i]));
		}

		return list;
	}
}

package tcc.action.utils;

import tcc.form.ArtifactForm;
import tcc.form.DifficultyForm;
import tcc.form.PhasesForm;
import tcc.form.ProjectForm;
import tcc.form.ProjectPhasesForm;
import tcc.form.ProjectUserForm;
import tcc.form.RequirementDependencyForm;
import tcc.form.RequirementForm;
import tcc.form.RolesForm;
```

```
import tcc.form.TaskForm;
import tcc.form.TaskHistoryForm;
import tcc.form.UserForm;
import tcc.persistence.artifact.ArtifactDefaults;
import tcc.persistence.difficulty.DifficultyDefaults;
import tcc.persistence.phases.PhasesDefaults;
import tcc.persistence.project.ProjectDefaults;
import tcc.persistence.projectphases.ProjectPhasesDefaults;
import tcc.persistence.projectuser.ProjectUserDefaults;
import tcc.persistence.requirement.RequirementDefaults;
import tcc.persistence.requirementdependency.RequirementDependencyDefaults;
import tcc.persistence.roles.RolesDefaults;
import tcc.persistence.task.TaskDefaults;
import tcc.persistence.taskhistory.TaskHistoryDefaults;
import tcc.persistence.user.UserDefaults;

import com.techlab.infrastructure.ejb.ValueObject;

public class ValueObjectToForm {
	public static ArtifactForm artifactVoToForm(ValueObject vo) {
		ArtifactForm form = new ArtifactForm();

		form.setCoArtifact(((Integer)vo.get(ArtifactDefaults.COARTIFACT)).intValue());
		form.setNoArtifact((String)vo.get(ArtifactDefaults.NOARTIFACT));

		return form;
	}

	public static ProjectUserForm projectUserVoToForm(ValueObject vo) {
		ProjectUserForm form = new ProjectUserForm();

		form.setCoProject(((Integer)vo.get(ProjectUserDefaults.COPROJECT)).intValue());
		form.setCoRoles(((Integer)vo.get(ProjectUserDefaults.COROLES)).intValue());
		form.setCoUser(((Integer)vo.get(ProjectUserDefaults.COUSER)).intValue());
```

```
        return form;
}

public static ProjectForm projectVoToForm(ValueObject vo) {
        ProjectForm form = new ProjectForm();

        if(((Float)vo.get(ProjectDefaults.NUHOURDONE)) != null) {

form.setHourDone(((Float)vo.get(ProjectDefaults.NUHOURDONE)).doubleValue());
        } else {
                form.setHourDone(0);
        }
        form.setCoProject(((Integer)vo.get(ProjectDefaults.COPROJECT)).intValue());
        form.setName((String)vo.get(ProjectDefaults.NONAME));
        form.setStatus((String)vo.get(ProjectDefaults.NOSTATUS));

        //System.err.println((vo.get(ProjectDefaults.NUTOTALHOUR)).getClass());


form.setTotalHour(((Float)vo.get(ProjectDefaults.NUTOTALHOUR)).doubleValue());
        form.setVersion((String)vo.get(ProjectDefaults.NOVERSION));

        return form;
}

public static UserForm userVoToForm(ValueObject vo) {
        UserForm user = new UserForm();

        user.setCoUser(((Integer)vo.get(UserDefaults.COUSER)).intValue());
        user.setAddress((String)vo.get(UserDefaults.NOADDRESS));
        user.setEmail((String)vo.get(UserDefaults.NOEMAIL));
        user.setName((String)vo.get(UserDefaults.NONAME));
        user.setPassword((String)vo.get(UserDefaults.NOPASSWORD));
```

```
            user.setUsername((String)vo.get(UserDefaults.NOUSERNAME));

            return user;
        }

        /*public static StatusForm statusVoToForm(ValueObject vo) {
            StatusForm form = new StatusForm();

            form.setCoStatus(((Integer)vo.get(StatusDefaults.COSTATUS)).intValue());
            form.setName((String)vo.get(StatusDefaults.NONAME));

            return form;
        }*/

        public static DifficultyForm difficultyVoToForm(ValueObject vo) {
            DifficultyForm form = new DifficultyForm();


        form.setCoDifficulty(((Integer)vo.get(DifficultyDefaults.CODIFFICULTY)).intValue());
            form.setName((String)vo.get(DifficultyDefaults.NODIFFICULTY));

            return form;
        }

        public static RequirementForm requirementVoToForm(ValueObject vo) {
            RequirementForm form = new RequirementForm();


        form.setCoDifficulty(((Integer)vo.get(RequirementDefaults.CODIFICULTTY)).intValue());


        form.setCoRequirement(((Integer)vo.get(RequirementDefaults.COREQUIREMENT)).intVa
lue());
            form.setNoStatus((String)vo.get(RequirementDefaults.NOSTATUS));
```

```
        form.setHourDone(((Float)vo.get(RequirementDefaults.NUHOURDONE)).doubleValue());
                form.setName((String)vo.get(RequirementDefaults.NONAME));

        form.setNoCommentary((String)vo.get(RequirementDefaults.NOCOMMENTARY));
                form.setNoVersion((String)vo.get(RequirementDefaults.NOVERSION));

        form.setTotalHour(((Float)vo.get(RequirementDefaults.NUTOTALHOUR)).doubleValue())
;

        form.setCoBackRequirement(((Integer)vo.get(RequirementDefaults.COBACKREQUIREM
ENT)).intValue());

        form.setCoFirstRequirement(((Integer)vo.get(RequirementDefaults.COINITIALREQUIRE
MENT)).intValue());

        form.setCoProject(((Integer)vo.get(RequirementDefaults.COPROJECT)).intValue());

        form.setCoRelatedTo(((Integer)vo.get(RequirementDefaults.CORELATEDTO)).intValue())
;

                return form;
        }

        public static RolesForm roleVoToForm(ValueObject vo) {
                RolesForm form = new RolesForm();

                form.setCoRoles(((Integer)vo.get(RolesDefaults.COROLE)).intValue());
                form.setName((String)vo.get(RolesDefaults.NONAME));

                return form;
        }

        public static PhasesForm phaseVoToForm(ValueObject vo) {
```

```
        PhasesForm form = new PhasesForm();

        form.setCoPhase(((Integer)vo.get(PhasesDefaults.COPHASE)).intValue());
        form.setName((String)vo.get(PhasesDefaults.NONAME));

        return form;
    }

    public static TaskForm taskVoToForm(ValueObject vo) {
        TaskForm form = new TaskForm();


    form.setCoRequirement(((Integer)vo.get(TaskDefaults.COREQUIREMENT)).intValue());
        form.setNoTask((String)vo.get(TaskDefaults.NOTASK));
        form.setCoTask(((Integer)vo.get(TaskDefaults.COTASK)).intValue());
        form.setNoCommentary((String)vo.get(TaskDefaults.NOCOMMENTARY));

        return form;
    }

    public static RequirementDependencyForm reqDependencyVoToForm(ValueObject vo) {
        RequirementDependencyForm form = new RequirementDependencyForm();


    form.setCoDependantReq(((Integer)vo.get(RequirementDependencyDefaults.CODEPEND
ANTREQ)).intValue());

    form.setCoReqDependency(((Integer)vo.get(RequirementDependencyDefaults.COREQDEP
ENDENCE)).intValue());

    form.setCoRequirement(((Integer)vo.get(RequirementDependencyDefaults.COREQUIREM
ENT)).intValue());

        return form;
```

```
	}

	public static ProjectPhasesForm projectPhaseVoToForm(ValueObject vo) {
		ProjectPhasesForm form = new ProjectPhasesForm();

		form.setCoPhase(((Integer)vo.get(ProjectPhasesDefaults.COPHASE)).intValue());

	form.setCoProject(((Integer)vo.get(ProjectPhasesDefaults.COPROJECT)).intValue());

	form.setCoProjectPhase(((Integer)vo.get(ProjectPhasesDefaults.COPROJECTPHASES)).int
Value());

		return form;
	}

	public static TaskHistoryForm taskHistoryVoToForm(ValueObject vo){
		TaskHistoryForm form = new TaskHistoryForm();


	form.setCoArtifact(((Integer)vo.get(TaskHistoryDefaults.COARTIFACT)).intValue());
		form.setCoPhase(((Integer)vo.get(TaskHistoryDefaults.COPHASE)).intValue());
		form.setCoTask(((Integer)vo.get(TaskHistoryDefaults.COTASK)).intValue());

	form.setCoTaskHistory(((Integer)vo.get(TaskHistoryDefaults.COTASKHISTORY)).intVal
ue());
		form.setCoUser(((Integer)vo.get(TaskHistoryDefaults.COUSER)).intValue());

	form.setNuHourDone(((Float)vo.get(TaskHistoryDefaults.NUHOURDONE)).doubleValue(
));

		return form;
	}
}
```

11.2 Artigo

Aplicativo para o gerenciamento de requisitos

Vitor Oba, Ricardo Pereira e Silva, Iomani Engelmann Gomes

Ciências da Computação
INE – Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC), Brasil, 88040-900
Fone (0XX48)333-9999, Fax (0XX48)333-9999
vitoroba@inf.ufsc.br, ricardo@inf.ufsc.br, iomani.engelmann@pixeon.com.br

Resumo

Por muitos anos, o foco de um bom produto era o próprio produto, e muitas informações da confecção do mesmo eram perdidas, e o processo que o desenvolvia não era avaliado.Atualmente, o produto com qualidade é consequência de um projeto bem feito e inspecionado.
Um projeto bem definido deve ter um série de passos e um deles é o gerenciamento de requisitos. Com o gerenciamento de requisitos é possível gerenciar os requisitos de um projeto assim como identificar inconsistências entre os requisitos.
Este projeto voltou-se ao desenvolvimento de uma ferramenta para auxiliar as práticas de gerenciamento de requisitos baseada no modelo CMMI(Capability Maturity Model Integration), se restringindo ao nível 2 deste modelo. Uma ferramenta operacional e totalmente extensível para projetos futuros adicionarem integrações com ferramentaas de padrão no mercado.

Palavras-chave: Gerenciamento de requisitos, CMMI.

Abstract

For many years, the focus of a good product was the product itself, and a lot of information to implement it was lost, and the process that develops it wasn't evaluated. Nowadays, the product with quality is the consequence of a well done and inspected project.
A well done project must have a set of steps and one of them is the requirement management. With the requirement management is possible to manage the requirement of a project as well as identify inconsistency between them.
This project focus in the development of a tool to help in the practice of requirement management based in the CMMI (Capability Maturity Model Integration) model, restricting in the level 2 of this model. An operational tool and totally extensible for future projects add integrations with standard tools in the market.

Key-words: Requirement management, CMMI.

1. Motivação

Hoje em dia, os aplicativos que fazem o gerenciamento de requisitos exigem o preenchimento extensivo de *templates* para fazer o gerenciamento dos mesmos. Este entrave, torna esta tarefa dispendiosa em tempo e por muitas vezes monótona. Gera uma burocratização que pode acarretar com o abandono do uso deste gerenciamento.

Além disso, dados estatísticos mostram que até 70% do esforço do projeto são gastos com revisões e validação das necessidades iniciais do cliente, principalmente por requisitos não atendidos ou mal executados.

Adicionado a este contexto, as boas práticas de engenharia de software nas empresas brasileiras, de modo geral, não são bem executadas ou definidas. Muitas destas questões, pelo alto custo das ferramentas e também pela pouca flexibilidade que as mesmas apresentam. Com isto, ter um trabalho aberto para execução deste importante passo dentro de um porcesso de software, torna-se o principal motivador na execução deste trabalho.

2. CMMI

O CMMI é um modelo de processo que foi proposto fazendo a integração dos seguintes modelos[CSR2]:

- Capability Maturity Model for Software (SW-CMM) v2.0 draft C;
- Eletronic Industries Alliance Interim Standard (EIA/IS) 731;
- Integrated Product Development Capability Maturity Model (IPD-CMM) v0.98;

O propósito dos modelos do CMMI é propor uma orientação que permita melhorias de um processo de uma organização. O papel dela é fornecer mecanismos para que uma organização gerencie o desenvolvimento, a aquisição e manutenção dos serviços, construindo processos estáveis, capazes e maduros.

2.1 Representações

O CMMI tem duas representações: uma em estágios e outra contínua.

A representação em estágios é dividida em níveis de maturidade. São cinco níveis de maturidade: inicial, gerenciado, definido, quantitativamente gerenciado e otimizado. Cada nível de maturidade é constituído de áreas de processo, onde estes possuem objetivos específicos e genéricos. Os objetivos específicos possuem práticas específicas associadas a eles e os objetivos genéricos possuem características comuns que organizam as práticas genéricas. A figura abaixo mostra a estrutura da representação em estágios.

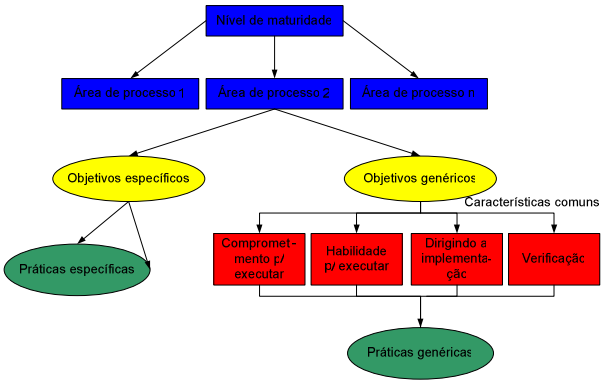


Figura 1: Representação em estágios

A representação contínua é dividida em níveis de capacidade. São seis níveis de capacidade: incompleto, executado, gerenciado, definido, quantitativamente gerenciado e otimizado. A representação contínua é constituída de áreas de processo, onde estes possuem objetivos específicos e genéricos. Os objetivos específicos possuem práticas específicas associadas a eles e os objetivos genéricos possuem práticas genéricas associadas a eles. A figura abaixo mostra a estrutura da representação contínua.

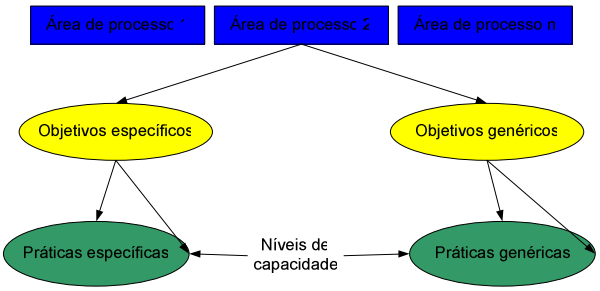


Figura 2: Representação contínua

A seguir será feita uma breve explanação do gerenciamento de requisitos do CMMI nível 2, onde está definido o escopo do aplicativo.

2.2 Gerenciamento de requisitos

O propósito do gerencimanto de requisitos é gerenciar os requisitos de um projeto e garantir que tais requisitos atendam às expectativas iniciais do

cliente. Esta área de processo possui um objetivo específico e duas genéricas, são eles:

- **Objetivo específico 1:** Gerenciar requisitos;
- **Objetivo genérico 2:** Institucionalizer um processo gerenciado;
- **Objetivo genérico 2:** Institucionalizar um processo definido;

As práticas do objetivo especíco 1 são as práticas relacionadas a este projeto. A seguir são citadas tais práticas:

- Obter um entendimento dos requisitos;
- Obter o comprometimento com os requisitos;
- Gerenciar mudanças nos requisitos;
- Manter a rastreabilidade bi-direcional dos requisitos;
- Identificar inconsistências entre produtos de trabalho e os requisitos;

Os objetivos específicos aqui apresentados definem o escopo da ferramenta. Além disso, a ferramenta proposta semi-automatiza as práticas relacionadas a estes objetivos. Por exemplo, o usuário não precisa preencher longos documentos para o aplicativo fornecer a matriz de rastreabilidade.

Na seção a seguir, descreve-se o aplicativo que cobre os objetivos específicos aqui apresentados.

3 Ferramenta

Como dito anteriormente, o aplicativo está restrito ao gerenciamento de requisitos do CMMI nível 2. Mais especicamente, o aplicativo se restringe aos objetivos específicos do gerenciamento de requisitos do CMMI nível 2.

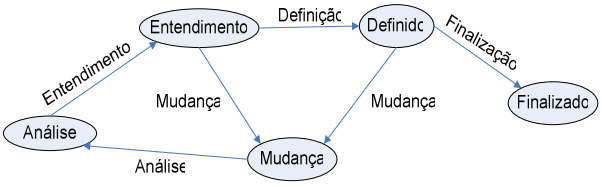
O aplicativo foi desenvolvido voltado para a *web* para permitir que todos os *stakeholders* (indivíduos que de alguma forma podem influenciar um projeto) de um projeto tenham o acesso ao mesmo e na hora que assim o desejar.

O administador do sistema tem a possibilidade de definir uma hierarquia de usuários, onde pode-se definir o que o usuário pode fazer no sistema, podendo o usuário ter diferentes papéis em diferentes projeto.

O aplicativo tem um módulo que permite o cadastramento de projetos, já que requisitos pertencem a um projeto. Todo projeto tem uma versão e sua versão só pode ser fechada caso todos os seus requisitos estejam fechados.

Cada projeto é composto de um ou mais requisitos. Para a inclusão de um requisito em num projeto, o gerente de projeto tem três alternativas: importar um requisito de algum outro projeto, criar um nova versão de um requisito já existente dentro do projeto e que sua versão esteja finalizada ou ser um novo requisito puro e simples.

O aplicativo faz o controle do status do requisito, ou seja, dado um status inicial, o aplicativo exibe as possibilidades seguintes. Para tal, o aplicativo segue as máquinas de estados mostrada na figura abaixo:



Tarefas são atrbuídas aos requisitos e tais tarefas são atividades inseridas pelo gerente de projeto para implementar o requisito. Uma tarefa é atrubuída automaticamente ao gerente de projeto, podendo ele atribuí-la a mais alguém que participe do projeto.

Cada vez que alguém alterar um requisito ou fazer alguma atividade para implementá-lo, o usuário deve informar qual fase do desenvolvimento do projeto pertence a atividade que ele deseja cadastrar, quantas horas foram gastas e selecionar qual artefato foi utilizado.

A ferramenta permite também a geração da matriz de reastreabilidade dos requisitos mapeando bidirecionalmente os requisitos do projeto. Para a composição da matriz, a ferramenta leva em consideração as últimas versões dos requisitos que estão dentro do projeto. Esta matriz irá responder às seguintes perguntas[TCC3]:

- Quais os impactos de uma mudança de um determinado requisito[TCC3]?
- Onde um requisito foi mapeado e/ou implementado[TCC3]?
- Todos os requisitos foram cobertos e/ou implementados[TCC3]?
- O requisito é mesmo necessário[TCC3]?
- Quais os testes utilizados na verificação de um requisito[TCC3]?

Todas as informações relativas ao uso da ferramenta são armazenadas em um banco de dados.

3 Conclusão

Um projeto que atenda às expectativas de um cliente deve ser executado com uma série de passos, sendo um deles o gerenciamento de requisitos.

Ferramentas são utilizadas no gerenciamento. Porém as ferramentas atuais não automatizam as tarefas relacionadas ao gerenciamento, provocando resistências ou até mesmo o abandono das tarefas, mesmo consideradas de suma importância.

A ferramenta proposta neste trabalho auxilia no gerenciamento de requisitos de um projeto. Ela semi-automatiza as tarefas relativas ao gerenciamento, reduzindo assim o custo relacionado à burocracia, visto que nos aplicativos atuais é necessário o preenchimento extensivo de documentos. Com isso elimina o tempo gasto com o gerenciamento ou mesmo atualização dos documentos.

Logo, conclui-se que a adoção de um modelo de processo como o CMMI e a utilização de ferramentas que automatizem as tarefas relacionados ao CMMI é de grande importância, especialmente em lugares onde o processo de desenvolvimento de software é caótico.

3 Referências

[CSR 02] Carnegie Mellon University / Software Engineering Institute. Capability Maturity Model Integration (CMMISM) – Staged Representation, Version 1.1, 2002, 639p.

[CCR 02] Carnegie Mellon University / Software Engineering Institute. Continuous Maturity Model Integration (CMMISM) – Staged Representation, Version 1.1, 2002, 645p.

[TCC 03] Matias, Carlos A., Winck, Ricardo J. Suporte à rastreabilidade de informações ao longo das fases de desenvolvimento de software, 2003, 80p.

[CRA 02] Larman, Craig. Utilizando UML e padrões – Uma introdução à análise e ao projeto orientados a objetos, 2002, 492p.

[JMA 01] Java Magazine. Programação para internet, 2001, 54p.