Proposta de Extensão de Modelo Conceitual de Gerência de Processo de Desenvolvimento de *Software* para Representar Aspectos de Qualidade de *Software*

Walkiria Cordenonzi, Cirano Iochpe

Universidade Federal do Rio Grande do Sul-

Instituto de Informática

Caixa Postal 15064 - Bairro Agronomia - Porto Alegre – RS -CEP:91501-970

{walkiria, iochpe}@inf.ufrgs.br

Universidade Regional Integrada do Alto Uruguai e das Missões - Campus Santo Ângelo

Walkiria@urisan.tche.br

RESUMO

O projeto TransCoop, sendo desenvolvido no CPGCC da UFRGS investiga o suporte ao desenvolvimento de *software*, utilizando técnicas de gerência de projeto, banco de dados e trabalho cooperativo. Objetivo básico é o suporte às interações humanas e o empréstimo, entre os participantes, dos objetos sendo desenvolvidos. No âmbito do TransCoop foi desenvolvido um modelo conceitual, denominado TransCoop-M, cujo objetivo é permitir a definição e o monitoramento do processo de desenvolvimento de *software*.

Este artigo relata uma proposta de extensão ao TransCoop-M para que se torne um modelo orientado à qualidade do processo de desenvolvimento, utilizando o CMM (*Capability Maturity Model*) e de produto de *software*, empregando a norma internacional ISO/IEC 9126.

Palavras-chave: Qualidade de Processo de *Software*, Qualidade de Produto de *Software*, Ambiente de Processo de Desenvolvimento de *Software*.

1. Introdução

Um ambiente de desenvolvimento de *software* possui certos aspectos peculiares que facilitam o trabalho das equipes que desenvolvem *software*. É definido e entendido como uma rede de atividades com a finalidade de construir um produto de *software*. As atividades são realizadas por agentes humanos (possivelmente organizados em equipes) com o uso de ferramentas. Além de permitir o armazenamento de informações sobre o processo e produto de *software* [1]. Contudo, este ambiente não possui uma preocupação direta com a qualidade de processo ou de produto de *software*.

Por outro lado, existem organizações desenvolvedoras de *software* com dificuldades em gerenciar o seu processo de negócio (*business process*), principalmente no que se refere a qualidade de processo ou de produto de *software*.

O TransCoop-M sendo um modelo de gerência de processo de desenvolvimento de *software*, até então não estava preparado para representar nenhum aspecto relativo à qualidade, seja esta de produto ou de processo de desenvolvimento.

Neste artigo é apresentado o modelo TransCoop-MQ o qual adiciona as facilidades para representação e manipulação do TransCoop-M a capacidade de representar aspectos de qualidade ao processo de desenvolvimento de *software*.

O principal objetivo do TransCoop-MQ é unir duas grandes áreas do conhecimento: ambientes de desenvolvimento de *software* (utilizados para definir e monitorar o processo) e qualidade de *software*.

Na área de qualidade de processo utiliza-se o modelo CMM, desenvolvido pelo *Software* Engineering Institute da Carnegie Mellon University (SEI-CMU) e, quando se trata de produto de *software* utiliza-se a norma internacional ISO/IEC 9126.

O TransCoop-MQ está concentrado no mapeamento das práticas chave, que descrevem as Características Comuns do CMM, e nas subcaracterísticas da ISO/IEC 9126, para novos elementos inseridos neste modelo. Desta forma, obtém-se um modelo que mantém o formalismo e as facilidades do TransCoop-M e adiciona a capacidade de endereçar aspectos de qualidade, utilizando o conceito de *patterns*.

O restante desse artigo está organizado de seguinte forma. Na Seção 2, o modelo TransCoop-M é descrito de forma sucinta. Na Seção 3, os modelos de qualidade, a norma ISO/IEC 9126 e o CMM são, brevemente, descritos. Na Seção 4 alguns mapeamentos são apresentados os quais originam o modelo TransCoop-MQ. O trabalho é concluído na Seção 5.

2 Modelo TransCoop-M

O TransCoop baseia-se em técnicas de gerência de projeto, trabalho cooperativo e banco de dados para controlar projetos auxiliados por computador e pode ser definido como um modelo de suporte à gerência do processo de desenvolvimento de *software,* apresentando linguagem e modelo de dados específicos. O estudo atual do modelo e da linguagem é descrito, detalhadamente, em [2].

O modelo conceitual da metabase de dados do TransCoop é composto de classes de objetos e seus relacionamentos, de uma série de grafos de transições de estados e de um conjunto de restrições de integridade. As classes e os relacionamentos capturam os elementos envolvidos no processo em um nível conceitual sem preocupação com considerações sobre sua representação em um sistema computacional.

As classes que representam a dinâmica do processo (projeto, atividade e tarefa) estão associadas a estados. Alterações de estado identificam novos níveis de completude do processo e dos produtos. Por último, o uso de restrições permite a definição de regras de negócio, normas de qualidade e metodologias de desenvolvimento de *software*.

A proposta de auxílio à gerência de projeto inclui a implementação de um banco de dados, com base no modelo, que permite organizar as informações sobre os processos e seus estados e sobre os componentes sendo desenvolvidos.

2.4 Esquema da Metabase de Dados do TransCoop-M

A Figura 1 apresenta um diagrama de classes em notação UML (*Unified Modeling Language*) [3], representando as classes e os seus principais relacionamentos no TransCoop-M. Todas as classes possuem associação com a classe **Documento**, o que faz com que qualquer classe possa ser ligada a um documento. Da mesma forma, toda classe está associada com a classe **Direito**, o que permite estabelecer direitos de acesso sobre objetos individualmente. Esses dois relacionamentos, na Figura 1, foram omitidos para simplificar a representação do modelo.

As classes da metabase de dados podem ser de tipos. Com relação ao intervalo de tempo em que são válidas. O primeiro tipo, **classes perenes**, são utilizadas para a descrição de

elementos da organização. O segundo tipo são as **classes de projeto**, cuja existência está associada ao escopo de um projeto.

O esquema da metabase de dados do TransCoop possui, como parte central, três classes: papel, projeto e componente. Cada uma destas classes representa um fator essencial para a descrição do processo de desenvolvimento de *software*. Estes fatores são, respectivamente: os recursos humanos, o processo e o componente de *software* sendo desenvolvido.

Por motivo de espaço somente serão descritas, na próxima seção, as classes e relacionamentos mais importantes para o entendimento do modelo, as quais serão referidas na Seção 4.

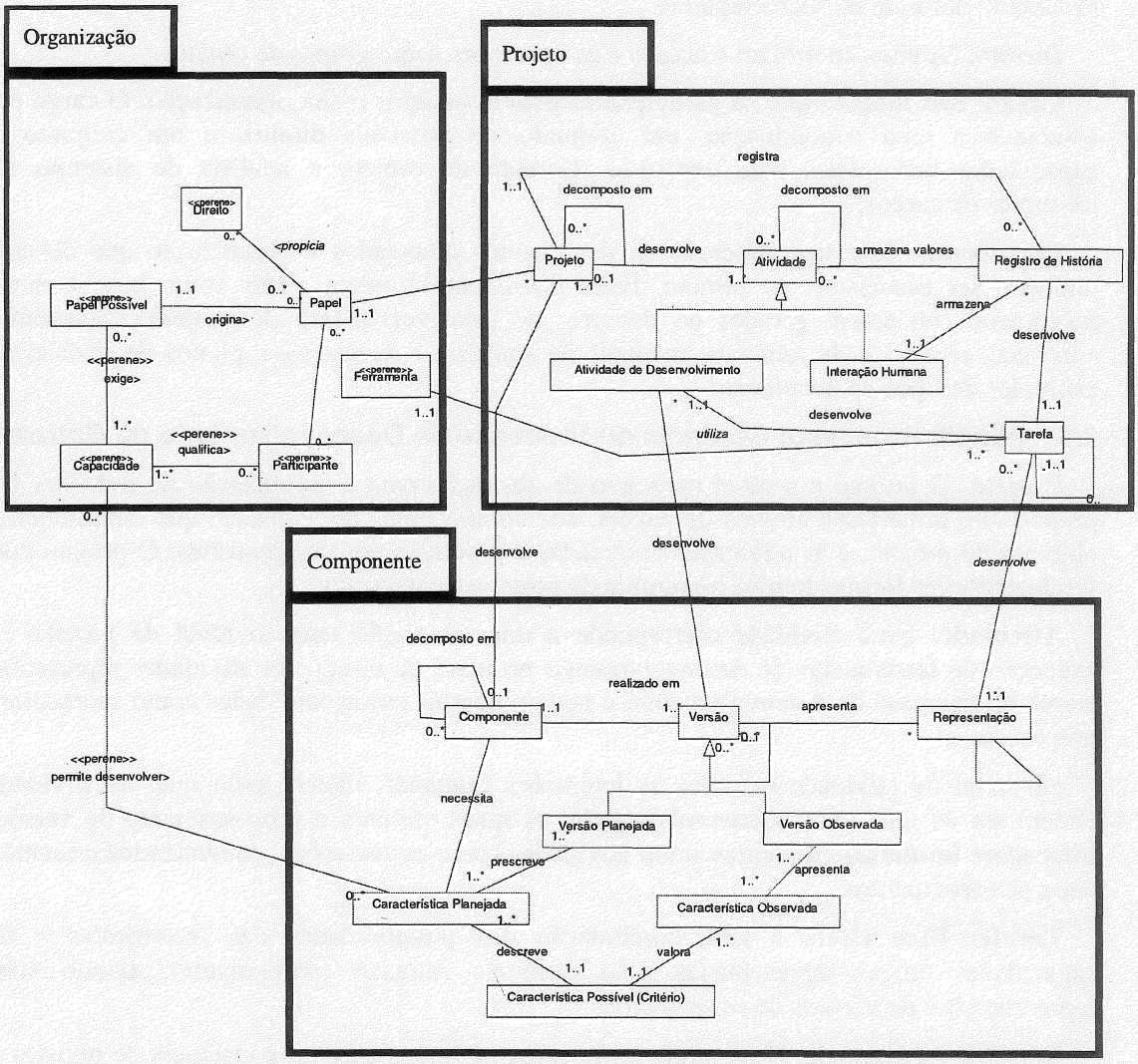

Figura 1 Esquema da metabase de dados do TransCoop-M

2.5 Pacote Organização

Característica Possível: O conceito de característica de componente de *software* é um dos pontos-chave do modelo. Através da definição de novas instâncias deste tipo, o modelo pode ser ampliado e adaptado para uma organização específica.

Ferramenta: Uma ferramenta representa um processo ou programa de computador capaz de efetuar uma transformação ou criar uma nova representação para uma versão de componente. Uma ferramenta executada sobre uma versão de componente implementa ou altera os valores de características planejadas para o componente (Pacote Produto, definido na Seção 2.7).

Participante: Um participante representa um indivíduo da organização disponível para desenvolver projetos.

Capacidade: As capacidades são definidas como habilidades, qualificações ou especializações de participantes e podem influir em sua indicação para um papel (Pacote Processo), definido na seção seguinte.

Direito: Direitos controlam o acesso e as operações sobre grupos de objetos.

Cargo: São funções que os participantes podem assumir numa organização. O cargo está associado a uma denominação, um conjunto de possíveis direitos e um conjunto de capacidades necessárias para exercê-lo. Gerente de projeto e analista de sistemas são exemplos de cargos.

Documento: Este tipo descreve os documentos associados à organização que deverão, também, ser associados ao projeto. Estes documentos podem servir como base a outros documentos ou serem gerados no decorrer do desenvolvimento do projeto. Orçamentos, contratos, relatórios de reuniões, manuais de *software* e *hardware* e planos de revisão são exemplos de tipos de documento.

2.6 Pacote Processo: Gerência do Processo de Desenvolvimento de *Software*

Projeto: O projeto é o nível mais alto de abstração para a organização do trabalho. Um projeto une projetistas, através de papéis, aos componentes de *software*, que determinam o objetivo do projeto, e às atividades associadas à produção dos componentes. O projeto pode ser decomposto formando uma hierarquia de projetos/subprojetos.

Atividade: Uma atividade corresponde a uma abstração entre o nível de projeto e a execução de ferramentas de desenvolvimento no nível de tarefa. As atividades representam etapas no processo de desenvolvimento e podem possuir outras atividades como sucessoras e predecessoras.

No nível de atividade ocorrem as interações humanas, motivo pelo qual as atividades podem ser de dois tipos: **desenvolvimento**, as quais efetuam o processamento de versões; **interações humanas** que representam atividades como negociações, comunicados e reuniões entre os participantes.

Tarefa: Uma tarefa é uma combinação das possibilidades das ferramentas e das capacidades únicas representadas pelo elemento humano (participante), agindo sobre representações de versões de componentes.

Registro de História: Uma tarefa ou interação humana gera uma instância de registro na história. Cada ocorrência da história corresponde a uma execução de ferramenta dentro de uma tarefa. As informações da história são um resumo dos valores dos relacionamentos da tarefa (e das interações humanas) no momento da execução da ferramenta.

2.7 Pacote Produto: Representação da Evolução do Componente de *Software*

Componente: Um componente de *software* modela um sistema, programa, tipo ou qualquer outra representação de *software* que seja significativa para o projeto. O componente

representa um objetivo a ser alcançado ao longo do projeto, sendo modelado através de suas características esperadas. Um componente pode ser decomposto em subcomponentes, formando uma hierarquia de componentes/subcomponentes.

Versão: Uma versão possui um participante criador e uma lista de características esperadas, as quais são um subconjunto daquelas previstas para o componente.

Característica Possível: O conceito de característica é um dos pontos chaves do modelo. Através da definição de novas instâncias desta classe, o modelo pode ser ampliado e adaptado para uma organização específica.

Característica Planejada: Características planejadas são associadas a **componentes** e **versões**. Esta classe fornece uma especificação de valores a uma característica.

2.8 Descrição do Processo

Um **projeto** recebe uma denominação e possui um participante responsável por sua conclusão e pelas decisões nele tomadas. Em um dado momento um projeto se encontra em um determinado **estado**. O estado auxilia a definir as condições internas de um projeto. Atualmente, existem seis possíveis estados que são: em definição, em execução, suspenso, interrompido e concluído [2].

As ações necessárias para a execução de um projeto são descritas através de **atividades** que são criadas para descrever quais os passos necessários para a execução do processo. Possuem cinco estados associados que são: em definição, habilitada, em execução, suspensa e concluída. As atividades que fazem uso de ferramentas são descritas em um nível maior de detalhe através das **tarefas**. As tarefas determinam uma série de dependências entre as ferramentas, que definem sua ordem de execução. As tarefas possuem cinco estados associados, que são: em definição, habilitada, em execução, concluída e cancelada.

Note-se que a partir destas informações pode-se construir um histórico de todas as execuções de ferramentas.

2.9 Descrição do Componente de *Software*

Um projeto pode definir um novo **componente**, pode utilizar um componente criado por outro projeto ou pode alterar um componente já existente.

Cada componente é implementado em uma série de atividades, em um nível inferior de abstração, através de **versões** associadas. A medida que as versões são concluídas, características planejadas do componente vão sendo implementadas. Quando todas as características planejadas estão presentes em uma de suas versões o componente pode ser dado como concluído.

2.10 Descrição da Organização

Participantes são instanciados fora do escopo de qualquer projeto e permanecem no banco de dados enquanto o indivíduo fizer parte da força de trabalho da organização, independentemente de associação a um projeto.

A utilização compartilhada de componentes, versões e representações pode estabelecer diferentes tipos de relações como cooperação ou negociação.

Para assumir uma função o participante deve apresentar um conjunto de capacidades. As capacidades podem ser utilizadas na seleção de participantes para a solução de um problema. De acordo com suas capacidades, participantes são associados a um ou mais projetos através de papéis. Todas as instâncias do modelo podem ser associadas a documentos da organização.

Os documentos estão espalhados pelo sistema de arquivos da organização e referem-se a tomadas de decisão, análise, projeto e outras informações relativas ao processo de desenvolvimento. O relacionamento com a classe **Documento** faz parte da herança comum a todas as classes.

3. Modelos de Qualidade: ISO/IEC 9126 e CMM

A norma ISO/IEC 9126 – *Information Technology- Software Quality Characteristics and Metrics* (Tecnologia da Informação – Características e Métricas de Qualidade de *Software*) consiste das seguintes partes [4]: 1- Modelo de Qualidade; 2- Métricas Externas;3 - Métricas Internas; 4- Métricas da Qualidade em Uso.

Somente a parte 1, desta norma, é utilizada neste trabalho pois esta descreve um modelo para qualidade de produto de *software*. A ISO/IEC 9126-1 privilegia a visão do usuário, seja este um usuário final, o pessoal da manutenção ou os operadores de *software*. Este modelo é representado de forma hierárquica. No primeiro nível estão definidas as características de qualidade de produto. No segundo nível, as subcaracterísticas. Estes níveis são bem definidos neste modelo de qualidade Figura 2.

A definição das características de qualidade apenas permite perceber um possível universo de requisitos que se enquadram no conceito das características. O desdobramento em subcaracterísticas serve para delimitar melhor o grande universo do escopo das características. Deve-se salientar que este desdobramento não é suficiente para especificar os requisitos de qualidade.

Para a aplicação deste modelo é necessário a definição de atributos (ou atributos de qualidade), que são dependentes de cada produto de *software* a ser desenvolvido. A definição destes atributos é o terceiro nível do modelo de qualidade, conforme pode ser visualizado na Figura 2.

Com relação ao CMM, este pode ser considerado como uma ferramenta que auxilia (a) a organização que desenvolve *software* a melhorar seus processos e (b) selecionar e gerenciar empresas externas contratadas para desenvolver algum produto de *software*. Em vista disso, a organização deve ter clareza sobre as necessidades específicas de seu negócio.

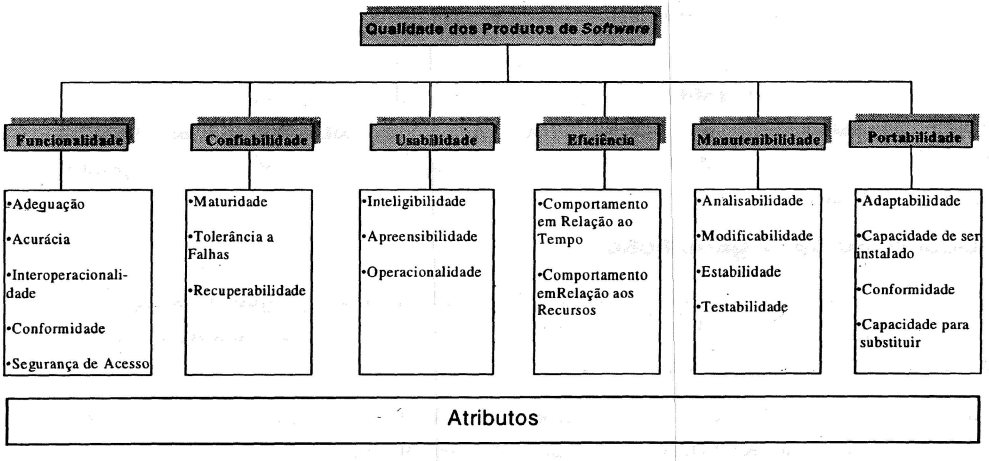

Figura 2 Estrutura do Modelo de Qualidade ISO/IEC 9126-1

O objetivo do CMM é descrever os aspectos de uma boa gerência, além de práticas de engenharia de *software* por meio de um *framework* de maturidade. Este *framework* é baseado

em 5 (cinco) níveis de maturidade com a finalidade de melhorar o processo de desenvolvimento de *software* continuamente, conforme mostrados na **Figura 3**.

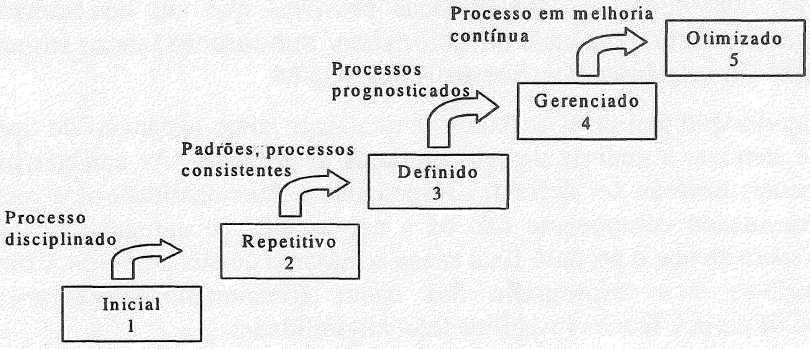

Figura 3 Estrutura do CMM

O CMM possui cinco Características Comuns [6]. Estas Características Comuns, por sua vez, contém práticas chave que descrevem as atividades a serem realizadas ou a infraestrutura da organização.

4. TransCoop-MQ

O TransCoop-M é um modelo de dados cuja principal função é oferecer suporte para a definição de elementos voltados ao processo de desenvolvimento de *software*. Porém, não está previsto, em sua definição, nenhum aspecto relativo a qualidade de produto ou de processo de desenvolvimento.

O modelo TransCoop-MQ é uma extensão do TransCoop-M que integra os modelos de qualidade CMM e ISO/IEC 9126 ao TransCoop-M. Esta integração é o resultado dos mapeamentos dos modelos de qualidade para o modelo de gerência de desenvolvimento de *software*.

As subcaracterísticas, da norma ISO/IEC 9126-1, e as práticas chave, referentes ao CMM, são mapeadas para uma das seguintes alternativas (conforme Figura 4): restrições, características possíveis ou novos tipos inseridos no TransCoop-M.

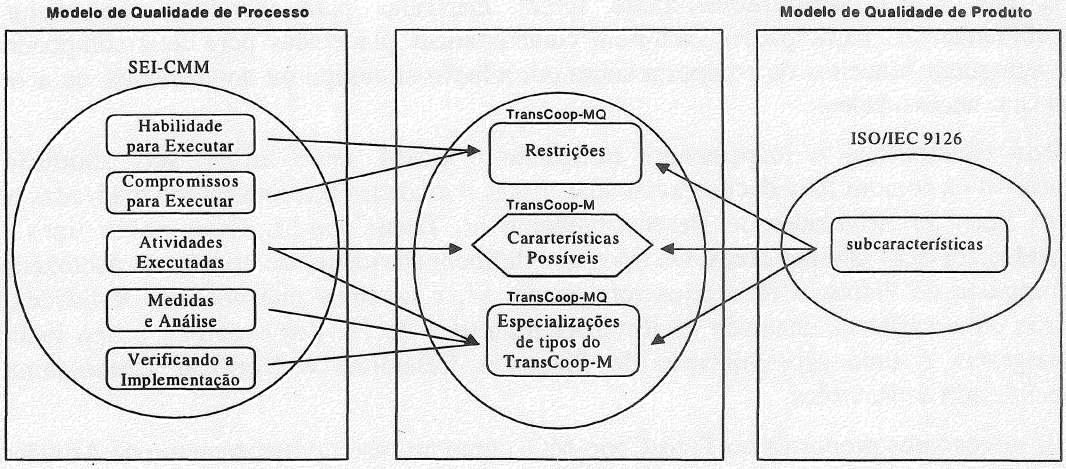

Figura 4 - Visão Geral do Mapeamento

Com relação ao mapeamento da norma para o TransCoop-M, e tendo que os atributos de qualidade são específicos de cada produto de *software*, no seu mapeamento para o modelo TransCoop-M, constituem as características possíveis que um determinado componente deverá agregar. Portanto, para cada característica/ subcaracterística, atributos deverão ser definidos como Característica Possível do TransCoop-M.

Cabe a organização definir os atributos de qualidade como instâncias do tipo Característica Possível que atendam a maioria de seus produtos de *software*. As características planejadas dos componentes deverão ser definidas respeitando a funcionalidade dos mesmos. Ou seja, para um determinado componente não há a necessidade de agregar todos os atributos de qualidade. Espera-se que o produto final reuna o máximo desses atributos. Como exemplos de atributos pode-se citar criptografia dos dados (funcionalidade/segurança de acesso), padronização de cores e ícones (usabilidade/inteligibilidade).

Nas seções seguintes a representação das as classes Atividade e Componente no TransCoop-MQ.

4.1 Atividade

As subcaracterísticas **Comportamento em Relação ao Tempo e Comportamento em Relação aos Recursos**, da característica **Eficiência**, deriva-se a atividade de desenvolvimento do tipo Tempo de Resposta e da segunda, Comportamento em Relação aos Recursos, a atividade do tipo Alocação de Recursos (conforme Figura 5).

Tal mapeamento dá-se pelo fato de que para verificar tempo de resposta e comportamento em relação aos recursos deve-se utilizar algum tipo de ferramenta, a qual deverá estar disponível na organização. Segundo a definição do TransCoop-M, as ferramentas estão associadas às tarefas, que por sua vez, estão associadas às atividades.

Nas tarefas associadas à atividade Tempo de Resposta poderão ser utilizadas ferramentas que avaliam o desempenho, por exemplo. Ainda, deve-se levar em consideração o *hardware* utilizado e a natureza do *software*. Na atividade Alocação de Recursos poderão ser desenvolvidos algoritmos de otimização e nas suas tarefas associadas, fazer uso de ferramentas que verifiquem o desempenho do sistema em relação ao volume de transações, velocidade, espaço de armazenamento, tempo de CPU, entre outros.

O fato destas subcaracterísticas serem mapeadas para atividade não exclui a possibilidade dos participantes incluírem características planejadas para seus componentes que agreguem conceitos de comportamento em relação ao tempo ou aos recursos, de acordo com suas necessidades.

Para exemplificar o mapeamento utilizando o CMM, neste artigo, será abordado a característica comum Atividades Executadas. Essas Atividades Executadas são mapeadas para novos tipos de Atividades de Desenvolvimento no TransCoop-M. Esses novos tipos são divididos em duas grandes áreas. Os tipos que definem atividades de criação de documentos, denominada de Planos e Relatórios totalizando 17, e os tipos que definem atividades de revisão e/ou auditoria, chamada de Revisões num total de 28. Por exemplo, o tipo Estimar Cronograma é uma especialização de Planos e Relatórios e Verificar Cronograma é especializado de Revisões.

Os novos tipos propostos no TransCoop-MQ, como no caso do mapeamento de Atividades Executadas (do CMM) e das subcaracterísticas (da norma), num primeiro momento, são especializações do tipo Atividade de Desenvolvimento, do TransCoop-M. Estas especializações acarretaram muitos níveis de hierarquia no modelo. Essas representações

possuem muitos tipos especializados e alguns deles não apresentam nenhum atributo ou método que os diferencie dos demais tipos.

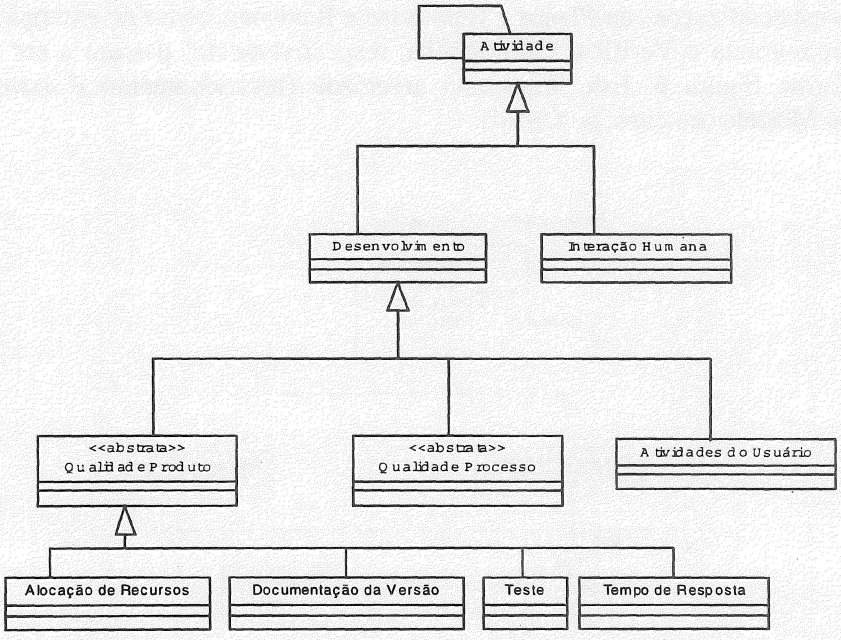

Figura 5 Extensão do Tipo Atividade de Desenvolvimento

Em um determinado modelo, quando muitas hierarquias ocorrem, torna-o muito complexo. A complexidade deste pode ser gerenciada por introduzir um nível de conhecimento – *knowledge level*. Para reduzir tal complexidade, o modelo é dividido em dois níveis: o nível operacional e o do conhecimento [7].

O modelo, no nível operacional, armazena os eventos do dia a dia do domínio. No nível do conhecimento este armazena as regras genéricas que governam a estrutura. As instâncias do nível de conhecimento governam a configuração das instâncias no nível operacional.

O nível operacional e do conhecimento são uma característica comum dos modelos, embora a diferença entre os níveis não é claramente apresentada. [7] faz a divisão explicitamente com o intuito de esclarecer a modelagem.

O TransCoop-MQ apresentou-se de uma forma bastaste complexa e com muitas hierarquias e/ou especializações. Por outro lado, a proposta do uso de *patterns* surge para reduzir justamente o problema encontrado no modelo.

Na Figura 6 é apresentado o tipo Atividade utilizando o conceito de *patterns*. O nível operacional permanece inalterado em relação ao modelo TransCoop-M, ou seja o tipo Atividade é especializado em Atividade de Desenvolvimento e Interação Humana. É no nível do conhecimento que as alterações ocorrem para que se torne possível inserir qualidade de produto e de processo de *software*.

As especializações no tipo Atividade de Desenvolvimento, provenientes do mapeamento da norma ISO/IEC 9126 para o TransCoop-M, passam a ser instâncias de **Tipo**. O **Tipo** é associado a um **Modelo**. No exemplo das subcaracterísticas Comportamento em Relação ao

Tempo e Comportamento em Relação aos Recursos, ambas mapeadas para tipos de Atividade de Desenvolvimento, são nesta nova representação instâncias de **Tipo**, associado ao **Modelo** ISO/IEC 9126.

Todas as especializações de Planos e Relatórios e Revisões, como no exemplo apresentado (Estimar Cronograma e Verificar Cronograma, respectivamente) passam a ser instâncias de **Tipo**, conforme Figura 6. Este **Tipo** está associado (relacionamento *é associado*) a um determinado **Modelo** (no caso, ao CMM).

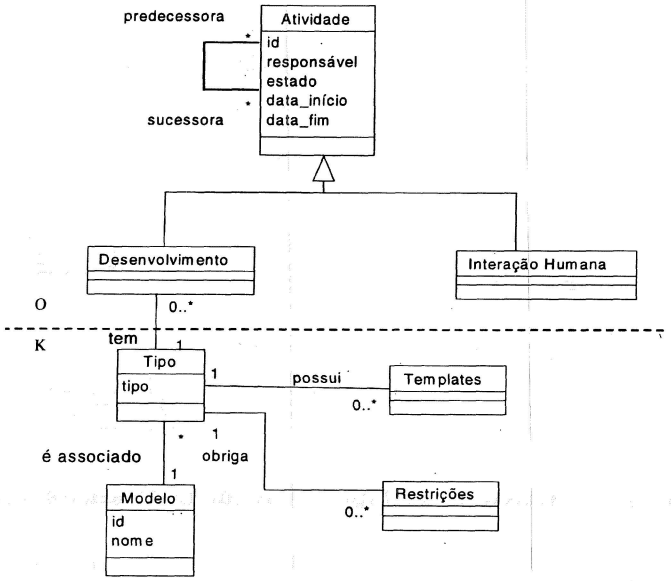

Figura 6 - Tipo Atividade com o uso de Patterns

Os *Templates*, em Atividades de Desenvolvimento, possuem informações relacionadas a definição e ao funcionamento do TransCoop-MQ, bem como, a qual **Tipo** de documentos deverão ser construídos na execução da determinada atividade.

Alguns tipos de atividades, previstas no TransCoop-MQ, possuem restrições, armazenadas em Restrições. Uma atividade do tipo Revisão Cronograma deve ser sucessora de uma atividade do tipo Estimar Cronograma é um exemplo de restrição de ordem entre atividades.

4.2. Componente

O tipo Componente somente é especializado quando realizou-se o mapeamento da ISO/IEC 9126-1 para o modelo TransCoop-M.

O modelo TransCoop-MQ, para esse tipo, é apresentado na Figura 7.

Componente, no nível operacional, não apresenta modificações em relação ao modelo TransCoop-M.

No nível de conhecimento, o tipo Componente assemelha-se a Atividades de Desenvolvimento, Projeto e Documento do TransCoop-MQ. As especializações no Modelo Intermediário são Interação com o Usuário, Sem Interação com o Usuário, Instalação do Produto e Manuais. Todas essas representadas em **Tipo**, no TransCoop-MQ. Esses tipos estão associados ao **Modelo** ISO/IEC 9126-1.

Quando emprega-se o TransCoop-MQ somente com o Modelo CMM habilitado, estes tipos não são válidos, pois referem-se ao Modelo ISO/IEC 9126-1. Existe a possibilidade do participante definir tipos para o CMM, por exemplo, produto de trabalho e produtos de trabalho de *software*, conforme este classifica o resultado de suas atividades e tarefas.

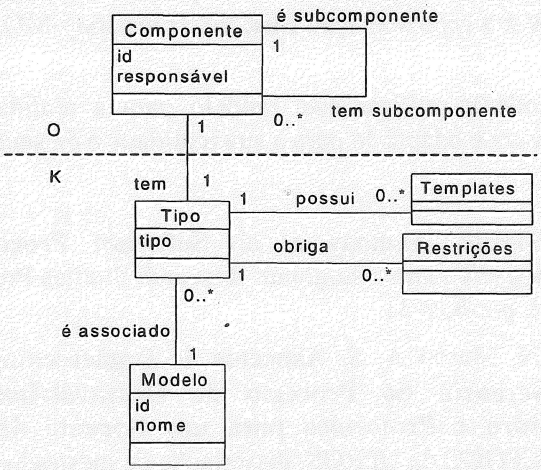

Figura 7 – Tipo Componente no TransCoop-MQ

Restrições sobre Tipos deverão ser definidas no elemento Restrições. Por exemplo, o tipo Manuais não pode ser subcomponente do tipo Instalação do Produto e vice-versa.

A semântica dos *templates* de Componente são semelhantes a Documentos. Componentes podem ser qualquer representação de software válida para os participantes e com *templates* a maior vantagem provém do reuso dos mesmos.

A seguir, um exemplo de definição de componente utilizando o TransCoop-MQ. Em **Modelo** existe a instância "ISO". Em **Tipo** as instâncias "Interação com o Usuário", "Sem Interação com o Usuário", "Manuais" e "Instalação do Produto". Não existe, nesse caso, nenhuma instância em ***template***. Uma instância de **Restrições**: "o tipo Instalação do Produto não pode ser subcomponente do tipo Manuais".

Na definição de um componente de software deve-se informar a qual **Tipo** esse pertence. Nesse exemplo, será definido um componente cujo objetivo é realizar a instalação do produto final. Portanto o **Tipo** desse componente será "Instalação do Produto", associado ao **Modelo** "ISO". Caso esse tipo de componente possua algum ***Template***, esse poderá ser utilizado. Se o Tipo "Instalação do Produto" obrigar **Restrições**, essas deverão ser respeitadas.

5. Conclusão

Esse artigo apresentou a proposta de extensão do modelo TransCoop-M denominada TransCoop-MQ. Este modelo agrega aspectos de qualidade de processo de desenvolvimento de *software* através do mapeamento do modelo CMM e da norma ISO/IEC 9126.

Em conseqüência do mapeamento originaram-se restrições, características possíveis e novos tipos. Devido a quantidade destes novos tipos gerados e algumas similaridades encontradas durante este processo, decidiu-se representar o TransCoop-MQ usando o conceito de *patterns*.

A maior contribuição deste trabalho reside no fato de agrupar duas grandes áreas do conhecimento: ambientes de desenvolvimento de *software* e qualidade de *software*. Além de propor um modelo genérico – TransCoop-MQ, baseado em *patterns*, no sentido de poder agregar quaisquer outros padrões sejam estes de qualidade ou não, sem que ocorra alterações na sua representação. Devido a questão de espaço não foi possível apresentar todos os mapeamentos realizados e a representação final do TransCoop-MQ, pode ser encontrado em [8].

Trabalhos futuros poderão utilizar este modelo para a realidade de uma organização desenvolvedora de *software* e adaptá-lo para a sua realidade e de seus produtos.

6. Bibliografia

[1] FINKELSTEIN, Anthony et al. **Software Process Modelling and Technology**. Taunton, England: Research Studies Press Ltda. 1994. 362p. chap.1, 2. p.1-8, 9-31.

[2] MANGAN, Marco A. S. **Aspectos de Implementação de um Modelo para Gerência do Processo de Desenvolvimento de *Software*: Arquitetura e Protocolos para um Gerente de *Designflow***. Porto Alegre: CPGCC da UFRGS, dissertação de mestrado. Março 1998. 113p.

[3] UNIFIED MODELING LANGUAGE. Disponível por WWW em http://www.rational.com/uml. (02 jul. 98)

[4] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **Information Technology - *Software* Product Evaluation- Quality Characteristics and Guidelines for their use**, ISO/IEC 91261/IEC 9126. [S.l.], 1991.

[5] KOSCIANSKI, André et al. **Avaliação de Qualidade de Produto de *Software* segundo o Modelo das Normas da ISO/IEC JTC1/SC7**. [S.l.:s.n], 1998.

[6] PAULK, Mark et al. **The Capability Maturity Model: Guidelines for Improving the *Software* Process**. Massachusetts: Addison-Wesley, 1995. 441p.

[7] FOWLER, Martin. **Analysis Patterns: Reusable Object Models**. California: Addison-Wesley, 1997. 357p.

[8] CORDENONZI, Walkiria. **Mapeamento de Padrões Internacionais de Qualidade de Produto e de Processo de *Software* para um Modelo Conceitual de Gerência do Processo de Desenvolvimento de *Software*.** Porto Alegre: CPGCC da UFRGS, dissertação de mestrado, 1999.