Modelagem de Processos de Software Através do SPEM - Software Process Engineering Metamodel - Conceitos e Aplicação

Elias Canhadas Genvigir
INPE - Instituto Nacional de Pesquisas Espaciais
CEFET - PR - Centro Federal de Educação Tecnológica do Paraná
elias@lac.inpe.br

Luiz Fernando Borrego Filho
INPE - Instituto Nacional de Pesquisas Espaciais
FIT – Faculdade Impacta Tecnologia
luiz@lac.inpe.br

Nilson Sant'Anna
INPE - Instituto Nacional de Pesquisas Espaciais
nilson@lac.inpe.br

Resumo

A modelagem de processos de software é atualmente uma grande fonte de pesquisa, na qual a comunidade acadêmica e as empresas tem aplicado diversos esforços. O artigo tem como objetivo apresentar os principais conceitos sobre a modelagem de processos, sobre o SPEM e algumas técnicas utilizadas para modelar processos através desse metamodelo, visto que ainda existem poucas publicações descrevendo a prática sobre essa especificação.

Abstract

The modeling of software processes is now a great research source, in the which the academic community and the companies have been applying several efforts. The article has as objective presents the principal concepts on the modeling of processes, on SPEM and some techniques used to model processes through of that metamodelo, because few publications still exist describing the practice about that specification

1. Introdução

Um modelo de processo de software é uma representação das atividades do mundo real de um processo de produção de software. Um modelo de processo de software é desenvolvido, analisado, refinado, transformado e/ou representado dentro de um meta processo. Assim, qualquer modelo de processo de software tem que modelar adequadamente o processo do mundo real e tem que atender às exigências específicas de cada fase do meta processo.

Os processos precisam ser formalizados ou padronizados de forma a entendermos o seu funcionamento, isso possibilita o melhor treinamento, as propostas para melhorias dentre outros fatores. Um modelo de processo deve especificar os pré-requisitos e conseqüências de cada tarefa, bem como a sincronização entre essas tarefas [1].

2. Modelos de Processo

A modelagem de um processo de software deve ser conduzida de modo a possibilitar o entendimento e a padronização do processo, para tanto devemos observar os principais objetivos para se modelar o processo de software que são respectivamente [7] [10]:

- Possibilitar a comunicação e o entendimento efetivo do processo;
- Facilitar a reutilização do processo (padronização);
- Apoiar a evolução do processo;
- Facilitar o gerenciamento do processo.

Diversos fatores nos levam a padronização dos processos organizacionais, mas as principais razões que nos levam a efetuar realmente sua padronização, são respectivamente:

- permitir treinamento, gerenciamento, revisões e ferramentas de suporte;

- utilizando-se processos padronizados, cada experiência de projeto pode contribuir para a melhoria dos processos na organização;
- um processo padronizado fornece uma base estrutural para medição
- definições de processo levam tempo e esforço o que torna impraticável novas definições de processo para cada projeto [7].

Os elementos de um modelo de processo e das linguagens de modelagem de processo devem ser abordados pelas seguintes questões [3]:

1. O que tem de ser definido dentro de um modelo de processo? Quais são os elementos de um processo de software e como eles são inter-relacionados?
2. Quais são os requisitos para uma Linguagem de Modelagem de Processos e quais são os relacionamentos que cada fase do meta processo e os respectivos meta usuários ?

Os elementos e seus relacionamentos de modelos poderão ser classificados em elementos (constituintes) e relacionamentos do problema orientado. Um modelo de processo concreto consiste de instâncias desses tipos de relacionamentos e elementos. Um processo de software consiste de atividades de cooperação concorrentes, que abrange atividades de manutenção e desenvolvimento de software, bem como atividades de segurança de qualidade e gerenciamento de projeto.

Segundo Derniame e outros [3] os elementos de um modelo de processo podem ser divididos em primários, secundários e auxiliares.

Os elementos e os relacionamentos permitem uma modelagem rústica-granulada de um processo de software. Uma linguagem de modelagem de processo de software tem que fornecer recursos de linguagem para modelar esses elementos básicos e avançados bem como seus inter-relacionamentos. Existem muitas variantes sobre como uma Linguagem de Modelagem de Processos suporta a modelagem desses elementos em um modelo de processo. Cada variante poderá ser classificada de acordo com certas características. Como não é possível abranger todas as características dentro de uma linguagem de especificação, na mesma profundidade, deve se tomar na construção do modelo da linguagem a decisão do que está estritamente relacionado a fase do meta-processo e quem está lidando com o modelo do processo.

3. Linguagens de Modelagem de Processo (PML's)

A modelagem de processos de software, iniciou através da utilização de técnicas de análise de sistemas, como a utilização de diagramas do paradigma estruturado (utilização de Diagramas de Fluxo de Dados, por exemplo) para representar um processo [7] [10], outros autores utilizam atualmente modelos orientados a objetos [6][9], por fim surgiram as chamadas (PML) linguagens de modelagem de processos, que agregam os diversos elementos utilizados nas formas de modelagem anteriores.

Uma PML expressa os processos de produção do software na forma de um modelo de processo, sendo uma descrição computacional de um processo externo.

O domínio do processo de software e o conceito de meta-processo configuram os requisitos básicos para uma PML: cada fase do meta-processo, por exemplo elicitação deve ser suportado por uma linguagem que permita expressar o modelo ao nível de abstração apropriado; cada elemento de processo, por exemplo, tarefas, papéis, e produtos, precisam ser descritos.

Observa-se em [3] que uma PML pode ser formal, semi-formal, ou informal. Uma PML formal é fornecida com uma sintaxe e semântica formalizada. As linguagens semi-formais normalmente têm uma notação gráfica com sintaxe formal, mas não uma semântica formal, por exemplo, não sendo executável. As linguagens naturais, como o inglês ou português, podem ser usadas nas PMLs informais.

Nos últimos 10 anos, muitas PMLs foram propostas, implementadas e testadas. Destaca-se algumas PMLs como a ProNet, [2] representante da primeira geração de PMLs, e as linguagens E3 "*Environment for Experimenting and Envolving Software Processes*" [8] e PROMENADE "*Process-oriented Modelling and Enactment of Software Developments*" [4] representantes de uma geração mais atual.

Em maio de 2000 a OMG - *Object Management Group* publicou a submissão inicial do UPM - *Unified Process Model* como uma proposta de unificação entre as diferentes metodologias para modelagem de processos [5]. A evolução do UPM se deu através da publicação em novembro de 2002 da versão 1.0 do SPEM - *Software Process Engineering Metamodel Specification* que é o metamodelo para definição de processos e seus componentes.

4. SPEM – *Software Process Engineering Metamodel*

O SPEM - Modelo de Processo Unificado é o metamodelo proposto pela da OMG para a descrição de um processo concreto de desenvolvimento de software ou uma família relacionada de processos de desenvolvimento de software.

Diferente da maioria das PML's o SPEM aparece como uma proposta de unificação entre as diferentes metodologias propostas para modelagem de processos. Da mesma forma que a modelagem orientada o objetos despertou uma série de propostas, o que acabou por

ocasionar a chamada guerra dos modelos, a modelagem de processos está passando pelo mesmo problema. Organizações, pesquisadores, universidades criam suas metodologias e nomeiam os elementos dos processos com diferentes termos causando grande dificuldade para o desenvolvimento de produtos e da pesquisa na área de desenvolvimento de software [5].

O SPEM utiliza a orientação a objeto para modelar uma família relacionada de processos de software e usa a UML como notação. A figura 1 mostra as quatro arquiteturas de modelagem definidas pelo OMG.

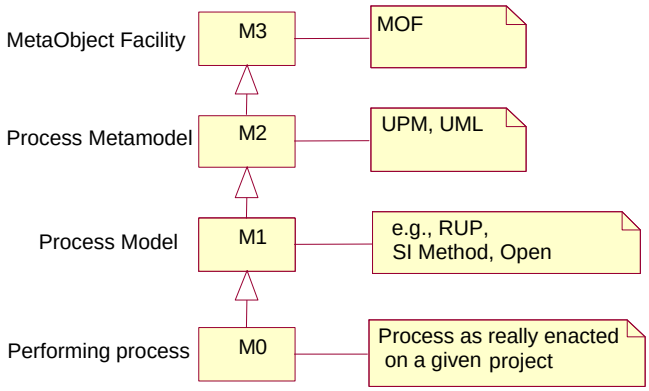

FIGURA 1.- níveis de modelagem definidos pela OMG - Fonte: [11] (OMG, 2002)

O SPEM encontra-se no nível M2 da arquitetura de quatro camadas da OMG e seu metamodelo é definido usando um subconjunto da UML de modo semelhante da UML com o MOF [12]. Este subconjunto da UML corresponde às facilidades implementadas e apoiadas pelo MOF.

Pela proposta do SPEM um processo executado é aquele, na produção no mundo real ou como o processo é ordenado, sendo nivelado no nível M0. A definição do processo correspondente está nivelado em M1. Por exemplo, o Processo Unificado Rational 2000 ou o Método IBM SI estão definidos no nível M1. Uma específica customização do RUP e o próprio RUP, ambos processos genéricos, usados por um determinado projeto, estão nivelados no nível M1.

O SPEM pode ser usado para definir todos os tipos de processos, incluindo os que estejam focalizados no uso específico da UML. Instâncias para orientar subclasses para descrever práticas com a UML podem ser criadas com ferramentas para um processo UML específico.

A UML é definida por um metamodelo, que é definido como uma instância do metamodelo MOF. Um subconjunto da notação gráfica da UML é usado para descrever este metamodelo. O metamodel SPEM foi elaborado similarmente, e é formalmente definido como uma extensão de um subconjunto da UML chamado SPEM_*Foundation*.

4.1 Escopo do SPEM

O SPEM é um metamodelo para definir processos e seus componentes. Uma ferramenta baseada no SPEM seria uma ferramenta para autoria e customização de processo. A atual execução de um processo, planejar e executar um projeto usando um processo descrito com o SPEM, não está no escopo do modelo.

A proposta inicial da OMG limita-se a definir o conjunto mínimo de processo que modela os elementos necessário para descrever qualquer processo de desenvolvimento de software, sem adicionar modelos específicos ou condições para qualquer área específica ou disciplina, como gerenciamento de projeto ou análise.

4.2 Principais Elementos de Definição do Processo

Os elementos de definição do processo ajudam na demonstração de como o processo será executado. Descrevem ou restringem o comportamento geral do processo em execução, e são utilizados para auxiliar o planejamento, a execução e o monitoramento do processo. São divididos em três grandes grupos: Pacote da Estrutura do Processo, Pacote dos Componentes do Processo e Pacote do Ciclo de Vida do Processo.

Um *Process* pode ser visto como uma colaboração entre papéis para alcançar uma certa meta ou um objetivo. Para guiar sua execução, determina-se a ordem na qual as atividades devem ser, ou podem ser, executadas. Também existe a necessidade de definir o modelo do processo no tempo, utilizando-se para isso a estrutura *Lifecycle* em termos de fases e iterações. Um *Process* é um *ProcessComponent* que pode ser apresentado sozinho como um processo completo. Um *ProcessComponent* é elemento de conveniência de descrições do processo que é consistente internamente e pode ser reutilizado com outros *ProcessComponents* para montar um processo completo. Um *ProcessComponent* importa um grupo não arbitrário de elementos de definição de processo, modelados no SPEM pelo *ModelElements*. Um *Lifecycle* de processo é definido como uma seqüência de fases que alcançam uma meta específica, definindo o comportamento completo de um processo que será executado em um dado projeto ou programa.

Um *Workdefinition* é um tipo de *Operation* que descreve o trabalho executado no processo. Suas subclasses são *Activity*, *Phase*, *Iteration*, e *Lifecycle*. Um *WorkDefinition* pode ser composto de outros *WorkDefinitions*, estando diretamente relacionados aos *WorkProducts* que são usados através da classe *ActivityParameter*, especificando se são usados como entrada ou como saída dos elementos de trabalho. O trabalho descrito no *Workdefinition* utiliza *Workproducts*

de entrada, e cria ou atualiza os *Workproducts* de saída. A dependência entre *WorkDefinitions* se faz utilizando a dependência Precedes, podendo indicar dependências de início-início, fim-início ou fim-fim entre os trabalhos. Se a atividade B tiver uma dependência fim-início sobre a atividade A, então B poderá iniciar somente depois de A ter sido encerrada, tendo seqüência rígida e sem paralelismo. Se a atividade B tiver uma dependência fim-fim sobre a atividade B, então B poderá finalizar somente após A ter finalizada, o paralelismo é possível com sincronização no início. Se a atividade B tiver uma dependência início-início sobre a atividade A, então B poderá iniciar somente após A ter iniciado, o paralelismo é possível com sincronização no início.

Um *WorkProduct* ou artefato é qualquer elemento produzido, consumido ou modificado por um processo. Ele pode ser uma parte da informação, um documento, um modelo, código fonte, bem como outros elementos. Descreve uma classe de produtos do trabalho produzidos em um processo e uma categoria de produto do trabalho, como documento texto, modelos UML, arquivos executáveis, bibliotecas de código, e assim por diante. A faixa de tipos de produto de trabalho é dependente do processo que está sendo modelado. Um *WorkDefinition* tem um *ProcessPerformer* próprio, representando o papel principal que executa este *WorkDefinition* no processo. Um *ProcessRole* é responsável por um grupo de *WorkProducts.* Uma dependência Impacts atua a partir de um *WorkProduct* em um outro *WorkProduct* para indicar que a modificação de um poderá afetar o outro. O *ProcessPerformer* representa de forma abstrata o "processo inteiro" ou um de seus componentes, e é usado para *WorkDefinitions* próprias que não têm mais um proprietário especifico. O *ProcessPerformer* tem uma subclasse, *ProcessRole*. Um *ProcessRole* define responsabilidades sobre *WorkProducts* específicos, e sobre os papéis que executam e auxiliam em atividades específicas. Um *ProcessPerformer* é o executor de *WorkDefinitions* agregadas de alto nível que não podem ser associadas com *ProcessRoles* individuais. A cada *WorkDefinition* pode ser associada uma Precondition e uma Goal. As *Preconditions* e *Goals* são *Constraints*, onde a *Constraint* é expressa na forma de uma expressão booleana seguindo sintaxe semelhante àquela de uma condição de proteção na UML. A condição é expressa em termos dos estados dos *WorkProducts* que são os parâmetros da *WorkDefinition* ou de uma *WorkDefinition* inclusa.

Uma *Activity* é a subclasse principal da *WorkDefinition*. Ela descreve uma parte do trabalho executado por um *ProcessRole*: as tarefas, operações, e ações que são executadas por um papel ou com a qual o papel pode auxiliar. Uma atividade pode consistir de elementos atômicos chamados *Steps.* Uma *Activity* é pertencente a um *ProcessRole* que é o seu executor, podendo fazer uso dos *ProcessRoles* adicionais que são os assistentes na *Activity*. A decomposição dentro da atividade é feita usando *Steps*. Um *Step* é descrito no contexto da atividade, inclusa, no *ProcessRoles* e *WorkProducts* que o usa. Também nos casos em que *Activity* são realizadas por um indivíduo ou um pequeno grupo, poderá ser utilizado uma *ProcessRole*. Como no *WorkDefinition*, a dependência entre uma *Activity* para com outra é feita utilizando-se a dependência Precedes.

Após a identificação das atividades, é possível agrupá-las em Discipline. A Discipline é uma especialização particular do *Package* que divide as atividades dentro de um processo de acordo com um "tema" comum. A divisão das atividades (em seções ou compartimentos) neste modo implica que o *Guidance* associado e os *WorkProducts* de saída são categorizados semelhantemente sobre o tema. A inclusão de uma *Activity* em uma *Discipline* é representada pela dependência Categorizes, com a condição adicional que toda *Activity* é categorizada por exatamente uma *Discipline*. Uma dependência Categorizes atua em um *Package* com um elemento individual do processo em outro pacote, e fornece um meio de associar elementos do processo com múltiplas categorias. Da mesma forma da UML, um pacote é um *container* que pode tanto possuir como importar elementos de definição do processo. As *Activities* e *WorkDefinitions* são pertencentes respectivamente aos *ProcessRoles* e *ProcessPerformers*; *StateMachines* são pertencentes aos *WorkProducts* e possuem seus estados internos e transições; ActivityGraphs podem ser pertencentes aos *Packages*, *Classifiers*, ou *BehavioralFeatures*; outros *ModelElements* do SPEM podem ser pertencentes aos *Packages.*

Os *Packages* e a dependência Categorizes podem ser utilizados para implementar categorias gerais dos elementos de descrição do processo. Um *Package* é criado para representar cada categoria, e todos os elementos ligados por uma dependência Categorizes neste pacote para representar o membro da categoria. Um *Package* representa uma categoria quando ele for a origem de pelo menos uma dependência Categorizes. O nome da categoria é o nome do pacote. Múltiplas categorias de sobreposição poderão ser criadas para servir aos vários propósitos na engenharia do processo. Por exemplo, nove disciplinas são descritas no Processo Unificado Racional: Modelagem de Negócios, Gerenciamento de Requisito, Análise e Projeto, Implementação, Teste, Desenvolvimento, Gerenciamento de Projeto, Configuração e Gerenciamento de Alteração, e Ambiente.

Alguns dos elementos do SPEM podem ser representados graficamente através de ícones outros existem como metainformação e não possuem representação gráfica.

Tabela 1 - Representação gráfica através de ícones dos principais elementos do SPEM

Estereótipo	Representação
WorkProduct	
WorkDefinition	
Guidance	
Activity	
ProcessRole	
ProcessPackage	
Process	
Document	
UMLModel	

6 Um Processo Modelado

O processo escolhido para representar uma modelagem através do SPEM foi o de Engenharia de Requisitos. Este processo é composto por quatro *Disciplines*, Elicitação, Análise e Negociação, Validação e Documentação, representadas pelo ícone de *ProcessPackage* como na figura 2.

Figura 2 - Representação da *Discipline* Elicitação. Um Processo pode ser composto por várias *Disciplines*.

São utilizados diagramas da UML e propostos pelo SPEM para modelar tanto os elementos estáticos como os dinâmicos do processo, como o de Pacote, de Caso de Uso, de Classe e de Atividade e adaptados com os respectivos estereótipos.

A *Discipline* Elicitação é composta por quatro *WorkDefinition*: Compreender o Domínio da Aplicação, Compreender o Problema a Ser Resolvido, Compreender o Contexto do Negócio e Compreender as Necessidades dos Usuários .

Na figura 3 observa-se em um nível de visão amplo a modelagem do *WorkDefinition* Compreender o Domínio da Aplicação através do diagrama de Pacotes. Todos as outras *Disciplines* e *WorkDefinition* do Processo seguem o mesmo padrão de diagramas e representações.

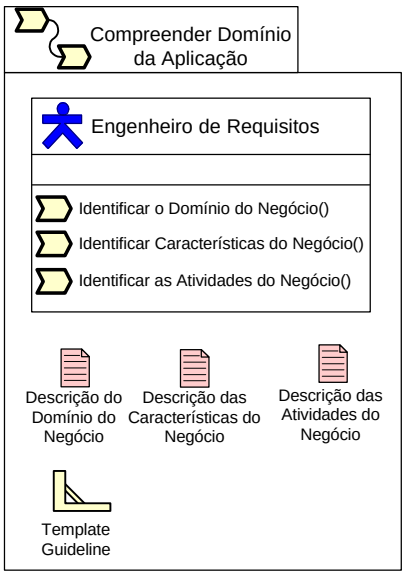

Figura 3 - Diagrama de Pacotes do WorkDefinition - Compreender Domínio da Aplicação. O Diagrama apresenta as Activities, os WorkProducts gerados pela WorkDefinition e o ProcessRole responsável.

O diagrama de Caso de Uso, figura 4, permite visualizar as Activities que cada *ProcessRole* executa ou participa na *Discipline* ou no *Workdefintion*.

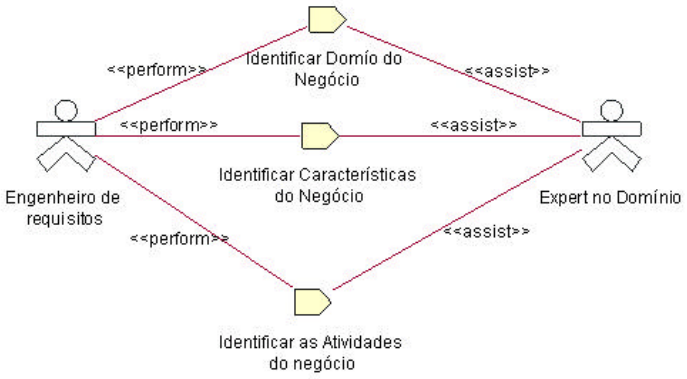

Figura 4 - Diagrama de Caso de Uso do *WorkDefinition* - Compreender Domínio da Aplicação.

O Diagrama de Classes, figura 5, permite a visualização dos *WorkProduct* gerados pelo *WorkdDefinition* ou *Discipline*, dos elementos utilizados

para a produção dos artefatos e quais os *ProcessRole* que são responsáveis por cada um dos artefatos gerados.

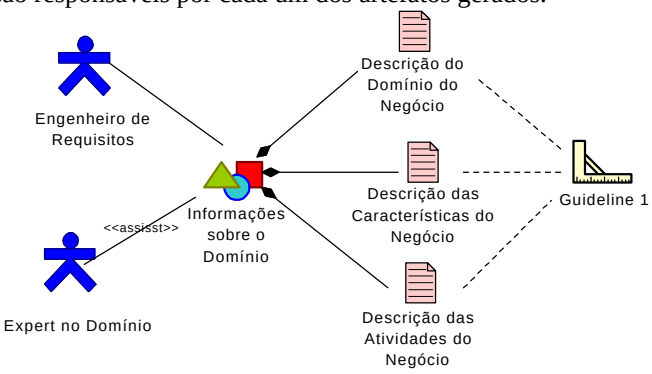

Figura 5 - Diagrama de Classe do WorkDefinition - Compreender Domínio da Aplicação

O Diagrama de Atividades, figura 6, permite visualizar a seqüência, a ordem e quais *ProcessRole* executam cada uma das *Activities* do *WorkDefinition* ou *Discipline*.

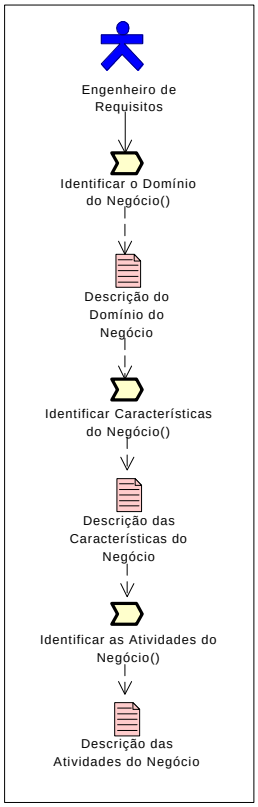

Figura 6 - Diagrama de Atividades do *WorkDefinition* Compreender Domínio da Aplicação. O *Expert* do Domínio não aparece porque ele só assiste as atividades e sua participação é opcional

8. Referências.

[1]. Ben-Shaul, I. Z.; Gai, K. E. A Paradigm for Decentralized Process Modeling and its Realization in the Oz Environment. IEEE Software, 1994, 179-188

[2]. Christie Alan M. Process-Centered Development Environments: An Exploration of Issues. SEI Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, (CMU/SEI-93-TR-4), 1993

[3]. Derniame, J.C., Kaba, B. A., Burkina Faso; Software Process: Principles, Methodology, Technology; 1999 XIII, Softcover

[4]. Franch X.; Ribó, J. M. Using UML for Modelling the Static Part of a Software Process, UML99 - Beyond the Standard Second International Conference, Fort Collins CO, USA, October 28-30, 1999

[5]. Genvigir, E., C.; Sant'anna, N.; Borrego, L., F., F.; Cereja, M., G., J., Casillo, B. H.; Modelando Processos de Software através do UPM - Modelo de Processo Unificado; XIV Congresso Internacional de Tecnologia de Software Qualidade de Software(14.:2003:Curitiba) Anais/XIV Congresso Internacional de Tecnologia de Software, 14. Curitiba, 24 a 27 de junho de 2003 - Curitiba:CITS, 2003.

[6]. Georgakopoulos D.; Hornick M. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. Distributed and Parallel Databases, Boston, 3, 119-153, 1995

[7]. Humphrey, W., S.; Kellner, M. Software Process Modeling: Principles of Entity Process Models. SEI Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, (CMU/SEI-89-TR-2), 1989.

[8]. Jaccheri M. L.; Baldi M.; Divitini M. Evaluating the requirements of software process modeling languages and systems, Process Support for Distributed Team-based Software Development (PDTSD99), Orlando, Florida, pages 570-578, August, 1999

[9]. Jacobson I; Booch G.; Rumbaugh J. The Unified Software Development Process. Addison-Wesley Publishing Company, Massachusetts, 1999.

[10]. Kellner, M. I.; Hansen G.; A. Software Process Modeling. SEI Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, (CMU/SEI-88-TR-9), 1988.

[11].Object Management Group, Software Process Engineering Metamodel Specification (SPEM), Formal Submission, OMG document number formal/02-11-14, November 2002.

[12].Object Management Group, MetaObject Facility Specification (MOF) Formal Submission, Version 1.4, April 2002, OMG Headquarters 250 First Avenue, Suite 201 Needham, MA 02494, USA