Universidade Federal de Pernambuco

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

2011.2

Revisão sistemática das características existentes em processos de desenvolvimento MDD (Model Driven Development) e o seu suporte para as áreas de processo de RD (Requirements Development) e TS (Technical Solution) do CMMI.

Trabalho de Graduação

Aluno	Bruno Florencio Pinheiro	{bfp@cin.ufpe.br}
Orientador	Alexandre M. Lins Vasconcelos	{amlv@cin.ufpe.br}

13 de Dezembro de 2011

“Existem coisas reservadas pra gente que fogem do nosso entendimento, mas que lá na frente vão fazer todo o sentido.”

Reynaldo Gianecchini

Agradecimentos

Essa página é a mais difícil de escrever em todo esse trabalho, pois, é através dela que eu vou resumir quatro anos e meio, ou até mais do que isso, de agradecimentos.

Durante todo esse tempo muitas coisas boas aconteceram em minha vida. Algumas outras ruins também. Mas, eu não seria capaz de conquistá-las e superá-las sem a ajuda de algumas pessoas. Tentarei não citar nomes para não ser injusto com ninguém.

Primeiro gostaria de agradecer aos meus pais e meu irmão que muitas vezes ficaram acordados até tarde sem saber que hora eu chegava ou me esperando para abrir a porta. Além deles e não menos importante do que eles têm meus avós, meus tios e tias, primos e primas. Enfim, a família. Obrigado por entender as noites que eu passei fora de casa, a ausência em algumas festas, a constante falta de tempo e a falta de atenção em alguns momentos.

Um parágrafo em especial tem que ser para agradecer a Natalia. Único nome que eu vou citar nesses agradecimento, pois, se no diploma tivesse espaço para um "vice-formando", o nome dela estaria lá no meu. Mesmo sem entender nada, ela escutava minhas explicações dos projetos, dava idéias, tentava ajudar de qualquer forma. Acho agora que ela já sabe até programar. Sem falar nas diversas vezes que cheguei atrasado ou cancelei alguma programação por causa do CIn ou do estágio. Obrigado por tudo.

Aos amigos de infância, tanto os da rua, colégio, e claro, a Irmandade, desculpa pela ausência em peladas, churrasco e festas. Aos amigos feitos no CIn, ou melhor, o G10, obrigado pela ajuda nos estudos e pela companhia nas viradas de noite.

O pessoal da Engenhotec e da Liferay está inserido nos amigos. Mas, eu gostaria de agradecer as empresas pela flexibilidade e ajuda sempre que necessitava de tempo para o CIn.

Saibam todos que eu sou muito grato por tudo que vocês fizeram por mim nesse tempo e que sem a ajuda de vocês eu não conseguiria concluir essa etapa tão importante na minha vida. Essa graduação é de todos vocês. Muito Obrigado!

Resumo

O presente trabalho de graduação tem com objetivo realizar uma Revisão Sistemática a fim de levantar os requisitos necessários para desenvolvimento de uma ferramenta Model Driven Development que suporte as áreas de processo Requirements Development e Technical Solution do CMMI.

Palavras-chave: Revisão sistemática; Model Driven Development; CMMI; Requirements Development; Technical Solution;

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

A preocupação em desenvolver software com maior qualidade e eficiência surgiu desde a década de 70. Porém, só a partir dos anos 90, que novos conceitos e abordagens conquistaram maturidade e visibilidade.

Tais conceitos trouxeram consigo uma carga maior de investimentos na melhoria de processos de desenvolvimento de software e adoção de modelos de qualidade de software. Entre eles, o CMMI [30] (Capability Maturity Model Integration) se destaca como um dos modelos mais reconhecidos em todo o meio.

Em paralelo aos conceitos de qualidade de software, surgiram ferramentas de geração automática de código. As quais nos últimos anos se tornaram tema presente em pesquisas na área de engenharia de Software.

O desenvolvimento dirigido a modelos, MDD (do inglês, Model Driven Development) aparece como um forte aliado nessa área. Porque a partir de um desenvolvimento formal, com o uso de modelos, é possível elevar a qualidade do software com técnicas formais de validação e verificação. E a partir de modelos com maiores níveis de abstração (domínio do problema) é possível chegar a modelos mais concretos (domínio da solução) até se obter, por fim, o código do sistema.

O objetivo desse trabalho de graduação é definir os requisitos mínimos suportados por uma ferramenta MDD de suporte ao CMMI e quais requisitos precisariam ser implementados para satisfazer as áreas de TS (Technical Solution) e RD (Requirements Development) do CMMI. Tais áreas foram escolhidas por estarem diretamente relacionadas à produção de modelos do sistema e ao desenvolvimento de código que são os principais objetivos das abordagens MDD de desenvolvimento de software.

2. MDD

O Model-Driven Development (MDD) é uma abordagem que utiliza modelos como uma especificação de software e através de transformações dos modelos obtém-se o código fonte. Os modelos são criados antes de o código fonte ser escrito ou gerado. MDD visa acelerar o desenvolvimento de software e torná-lo mais eficiente, usando os modelos para visualização do código e domínio do problema. MDD também separa a tecnologia de implementação da lógica de negócios de um programa [7].

A idéia por trás do MDD é a possibilidade de criar modelos e transformá-los em um sistema real [6]. Sistemas são modelados em vários níveis de abstração e em várias perspectivas e os modelos criados são os artefatos primários de desenvolvimento do software. Estes modelos podem ser transformados em sistemas executáveis usando geradores de códigos ou através da execução dos modelos em tempo de execução. [8].

Model-Driven Architecture (MDA) do Object Management Group (OMG) é o exemplo mais conhecido de MDD. E, vem sendo citado como um sistema de modelagem de software [9]. O OMG é um consórcio de indústrias orientadas para o desenvolvimento padrões que permitem a implementação de MDD. No MDA, a especificação das funcionalidades do sistema é separada da especificação de sua implementação em uma determinada plataforma ou tecnologia. Nela, usa-se Unified Modeling Language (UML) [31] para visualizar o código [10]. Quando separa-se as regras de negocio da implementação, os desenvolvedores são capazes de se concentrar mais em resolver problemas do que nos detalhes da implementação de tal tecnologia [7].

No caso do MDD, muitas vantagens e desvantagens são reconhecidas, por exemplo, em [6], [11], e [9]. Mellor et al. [6] afirmam que os modelos podem ser usados para aumentar a produtividade. Como exemplo, eles mencionam que "é mais barato construir um modelo gráfico em UML do que escrever código em Java".

No entanto, eles mencionam também que existem argumentos que afirmam que os modelos oferecem mais obstáculos do que benefícios. Ainda de acordo com Mellor et al. [6], MDD oferece um potencial para a transformação automática de modelos em sistemas executáveis. Eles também comentam que o desafio das empresas que adotam MDD está em como encontrar desenvolvedores que "pensem no nível de abstração, acima de linguagens de programação e tecnologia".

3. CMMI

O CMMI - Capability Maturity Model Integration é um conjunto de modelos integrados de maturidade e capacidade para diversas disciplinas, tais como: engenharia de software e sistemas, aquisição e desenvolvimento integrado do produto.[17]

Desenvolvido pelo SEI (Software Engineering Institute), órgão da Universidade Carnegie Mellon, patrocinado pelo Departamento de Defesa dos Estados Unidos [3], teve sua primeira versão lançada em 2002.

Derivado principalmente dos modelos SW-CMM [32] e SE-CMM [33], o CMMI surgiu da percepção de que software básico e aplicações são desenvolvidos em contextos integrados. [18] Por isso, como seus antecessores, o CMMI não define como o processo deve ser implementado, mas prescreve suas características estruturais e semânticas em termos de objetivos e do grau de qualidade com o que o trabalho deve ser realizado.

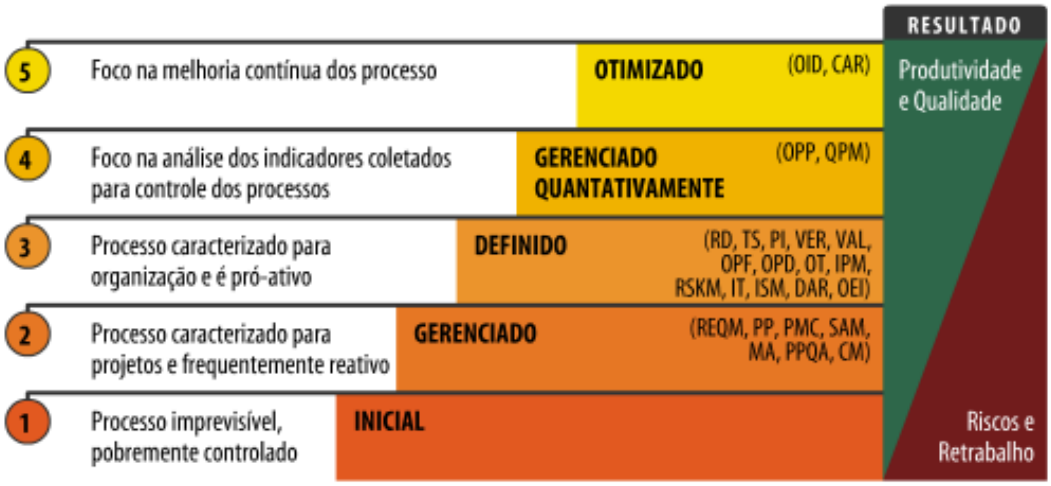

Figura 1 - Níveis do CMMI. Fonte: http://www.cits.br/cmmi.do

A avaliação do CMMI é realizada através de níveis de maturidade para o modelo estagiado e através das áreas de processo para o modelo contínuo. Os níveis de maturidade correspondem à capacidade da empresa em realizar projetos grandes e complexos. Estar em um nível de maturidade do CMMI

significa que a instituição implementa todas as áreas de processo do respectivo nível. Já no método contínuo, é avaliada a capacidade da empresa implementar uma determinada área de processo. Na figura 1, são demonstrados os níveis de maturidade do CMMI.

Cada nível de maturidade é composto de áreas de processos. O que faz com que o CMMI atenda as necessidades pontuais da organização, demonstrando uma maior eficiência. Ao todo, existem 25 áreas de processos no CMMI. Sendo elas divididas em quatro categorias: Gerenciamento de projetos, Gerenciamento de processos, Engenharia e Suporte.

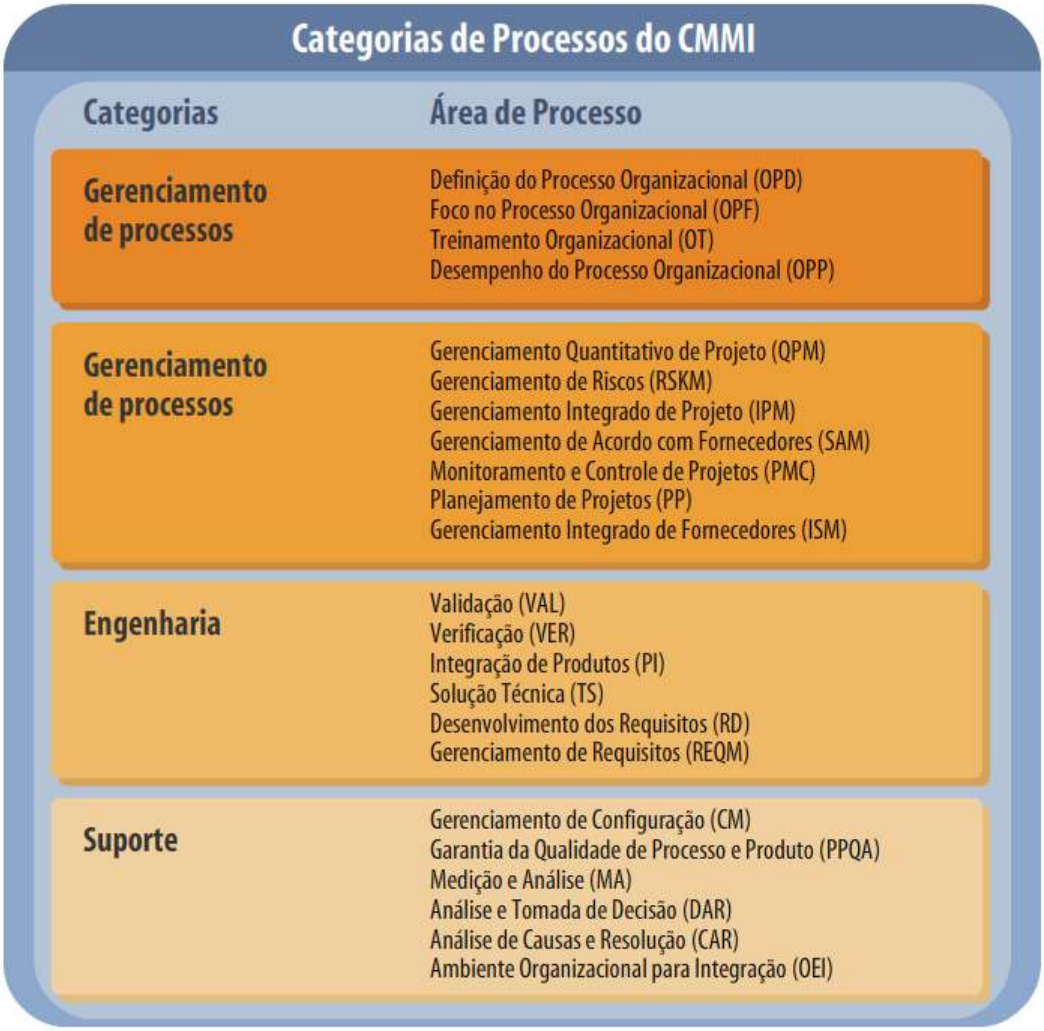

Figura 2 - Áreas de Processos do CMMI. Fonte: http://www.cits.br/cmmi.do

Nesse trabalho, abordaremos as áreas de Solução Técnica (TS) e Desenvolvimento de Requisitos (RD) por estarem relacionadas diretamente com a produção de modelos do sistema e ao desenvolvimento de código. Como podemos observar na figura 1, ambas as áreas de processo se encontram no nível 3 de maturidade do CMMI.

O RD descreve três tipos de requisitos: requisitos do cliente, requisitos do produto, e os requisitos de componentes do produto. Tomados em conjunto, estes requisitos atendem às necessidades relevantes as partes interessadas, incluindo as necessidades pertinentes às diferentes fases do ciclo de vida do produto (por exemplo, os critérios de testes de aceitação) e atributos do produto (por exemplo, a capacidade de resposta, segurança, confiabilidade de manutenção). Seu objetivo é produzir e analisar esses requisitos. [34]

A área de processo TS é aplicável em qualquer nível da arquitetura do produto e para cada produto, componente do produto e para cada processo de ciclo de vida do produto. Em todas as áreas de processo, onde os termos "produto" e "componente do produto" são usados, seus significados abrangem também serviços, sistemas, e seus componentes. Essa área visa desenhar, desenvolver e implementar as soluções para atender os requisitos estabelecidos. [34]

4. Revisão Sistemática

Uma revisão sistemática consiste em avaliar e interpretar todas as pesquisas disponíveis e relevantes sobre um tópico, assunto, fenômeno ou questão de pesquisa em particular. Com o objetivo de apresentar uma avaliação justa baseada em um protocolo confiável, rigoroso e baseado em uma metodologia de pesquisa.

Diferentemente de uma revisão literária, uma revisão sistemática é capaz de ser reproduzida e continuada por outros pesquisadores ou entusiastas do tema. Tal benefício é resultado da metodologia de pesquisa utilizada, a qual, através da forma sistemática como é conduzida, permite o desenvolvimento de um protocolo.

O protocolo é o meio principal de uma Revisão Sistemática, pois, é através dele que outros pesquisadores poderão julgar se a maneira como foi proposta a condução da pesquisa se adéqua a suas necessidades.

As primeiras revisões sistemáticas apareceram no início do século XX e tinham como meio de pesquisa a área de saúde. A partir da década de 30, outras áreas como estatística e física também aderiram a essa metodologia de pesquisa [1].

Acompanhado de uma melhoria na metodologia utilizada, surgiram algumas revisões no campo de engenharia de software. No entanto, esses estudos ainda são considerados como primários, pois, esse ramo trabalha com tecnologias que não possuem evidências suficientes para serem mensuradas.

Algumas iniciativas com o objetivo de mostrar como a Engenharia de Software poderia se beneficiar da revisão sistemática foram apresentadas. Uma delas é o modelo proposto por Kitchenham [12] para revisões sistemáticas, no qual é sugerido que uma revisão sistemática deve envolver três etapas essenciais, explicadas abaixo:

– Planejamento da revisão: a necessidade da revisão é identificada, as questões e o protocolo são definidos.

– Condução da revisão: os critérios para seleção de pesquisas é definido e alguns estudos iniciais são selecionados e seus dados extraídos. Tal processo é refinado até atingir um nível de maturidade aceitável pelo pesquisador.

– Divulgação da revisão: um relatório contendo o resultado da revisão é disponibilizado.

As atividades referentes ao planejamento e condução desta revisão seguirão o modelo proposto por Kitchenham [12] e serão descritos nos capítulos a seguir.

5. Planejamento da Revisão

O planejamento é visto como a atividade inicial do processo de revisão sistemática, onde são definidas uma ou mais questões de pesquisa e a metodologia a ser empregada para conduzir tal revisão, bem como as fontes e estratégias de busca.

Com o objetivo de levantar quais requisitos seriam necessários para uma ferramenta MDD de suporte a determinadas áreas do CMMI, formulamos a seguinte pergunta: "Quais requisitos apresentados em ferramentas MDD se aplicam as áreas de TS e RD do CMMI?". A partir de tal pergunta buscou-se extrair de diversas publicações científicas, características em comum e relevantes para a revisão.

A estratégia para a busca de estudos primários foi definida com base nas fontes de busca, idiomas dos estudos e palavras-chave relacionadas. As fontes correspondem aos locais de onde provêm as publicações, escritas em determinado(s) idioma(s) e encontrados por meio de palavras-chave bem definidas.

As fontes principais para obtenção dessas publicações foram os Periódicos da CAPES[1] e o Google Academic[2]. Ambos buscadores indexam um grande número de revistas científicas e provém acesso rápido às mesmas.

A consulta foi realizada usando palavras-chaves em inglês e português. Sendo elas: MDD, Model Driven Development, CMMI, TS, Technical Solution, RD, Requirements Development, Ferramentas MDD, MDD Tools. Grande parte das publicações retornadas estava escrita em inglês.

Além da consulta direta nesses portais, outras publicações referenciadas nos estudos retornados, também receberam a atenção devida e algumas foram adicionadas ao resultado final.

Especialistas na área também foram consultados e sugeriram alguns estudos para pesquisa.

[1] http://www.periodicos.capes.gov.br/
[2] http://scholar.google.com.br/

5.1. Critérios para seleção de um estudo

Foram incluídos estudos relacionados a ferramentas de suporte ao MDD nos quais se listavam as características de cada ferramenta.

Ferramentas que não se enquadravam nas áreas de processo do CMMI abordadas nesse estudo (TS e RD) foram excluídas. Também não foram relacionados estudos duplicados e referenciados em outros estudos que não puderam ser acessados.

5.2. Processo de Seleção Inicial

O processo de seleção inicial foi realizado através da consulta às fontes indicadas anteriormente. Os resumos dos estudos obtidos nessa etapa foram lidos com o objetivo de identificar quais publicações eram relevantes para a revisão.

Estudos que não possuem resumo serão lidos de maneira rápida dando atenção principalmente aos tópicos abordados.

5.3. Processo de Seleção Final

Nessa etapa, os estudos selecionados na fase anterior foram avaliados através da sua leitura na íntegra com base nos critérios de seleção com o objetivo de extrair dados relevantes para uma análise final.

6. Execução da Revisão

A condução da revisão se caracteriza pela identificação, seleção e avaliação de estudos primários de acordo com critérios de inclusão e exclusão estabelecidos durante a definição do protocolo de revisão.

Essa revisão foi realizada durante os meses de Setembro e Outubro de 2011. Ao todo foram obtidas 25 fontes de trabalhos. As quais foram refinadas ao passarem por a fase de seleção inicial e tiveram suas características extraídas na fase final.

Através da combinação das palavras-chaves citadas anteriormente no capítulo 4, obteve-se a seguinte *string* de busca:

(MDD Tools OR Ferramentas MDD)

AND

(

CMMI OR

(TS OR Technical Solution) OR

(RD OR Requirements Development)

)

Tal consulta foi aplicada às fontes selecionadas, sofrendo modificações, quando necessário, para adaptar-se ao padrão do meio de pesquisa. Dos 25 trabalhos recuperados, tanto pela consulta quanto pela indicação de

especialistas, apenas 10 passaram pela seleção inicial. O gráfico, ilustrado na figura 1, mostra que apenas 29% dos documentos seguiram para a fase de seleção final.

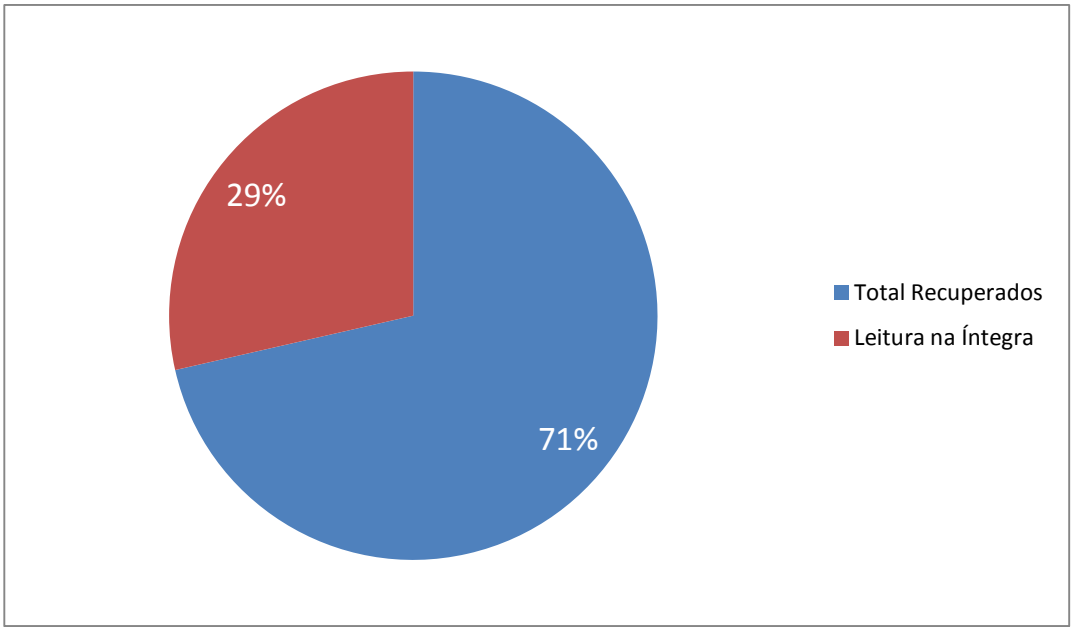

Figura 3 - Gráfico comparativo da quantidade de documentos

Após a leitura completa desses 10 documentos selecionados, foi possível extrair as características relevantes para responder a questão proposta inicialmente.

No capítulo a seguir, falaremos um pouco sobre as ferramentas selecionadas e em seguida mostraremos os resultados.

7. Ferramentas

7.1. Eclipse Modeling Project

O Eclipse Modeling Project centra-se na evolução e promoção de tecnologias de desenvolvimento baseadas em modelos dentro da comunidade Eclipse, fornecendo um conjunto unificado de estruturas de modelagem, ferramentas e implementações de padrões. [13]

7.2. Borland Together

O Borland Together é uma plataforma de modelagem visual desenhada para dar suporte a arquitetos, desenvolvedores, designers de UML, analistas de processos de negócios e modeladores de dados na criação acelerada de aplicações de software de alta qualidade. Independente da sua posição no projeto, o Borland Together propicia a todos os participantes a mesma compreensão visual das decisões com impacto sobre o design da arquitetura do software. [14]

7.3. OptimalJ

O OptimalJ é construído em cima do popular ambiente de desenvolvimento Netbeans , um produto de código aberto, que está na base de ferramentas da Sun Microsystems. Além disso, o OptimalJ é fornecido com três populares produtos de código aberto para desenvolvimento web: o JBoss EJB do servidor, o Apache Tomcat, e o Apache Axis web services engine..

O desempenho geral do OptimalJ neste ambiente foi considerado muito bom e ele fornece uma integração excepcionalmente suave com esses outros produtos de desenvolvimento web. Apesar de ser recomendado 1 GB de memória, o OptimalJ usa aproximadamente 350 MB de memória total durante seu processamento normal e pode rodar em uma máquina com apenas 512 MB,

um verdadeiro feito considerando a quantidade de funcionalidades incluídas na ferramenta.[15]

7.4. Objecteering

Com a flexibilidade dos seus editores gráficos, o Objecteering faz com que seja fácil de criar, associar, conectar, inserir, organizar e lidar com os elementos do modelo. Além de auxiliar através do seu editor gráfico, ele possui mais de 250 verificações de consistência que checam imediatamente os modelos, garantindo a consistência permanente e gestão de rastreabilidade. [16]

7.5. MDDi

O projeto MDDi tem como objetivo a integração dos serviços projetados para a plataforma Eclipse Modeling Framework (EMF), bem como ferramentas externas. Alguns espaços tecnológicos, tais como XML ou transformação gráfica fornecem soluções para problemas MDD. Por exemplo, as tecnologias formais fornecem um conjunto de boas práticas e ferramentas para validar os modelos.

O problema é que estas tecnológicas contam com ferramentas de suporte heterogêneas que geralmente precisam ser ligadas manualmente ou programaticamente usando ligações específicas. O projeto MDDi aborda esta questão no contexto do MDD. [19]

7.6. GMT Project

O objetivo do projeto Generative Modeling Tools project (GMT) é produzir um conjunto de ferramentas de pesquisa na área de MDSD (sigla em inglês para Desenvolvimento de Software Orientado a Modelos). Esta contribuição tem como objetivo ilustrar a variedade de operações aplicáveis a modelos abstratos. Historicamente, a operação mais importante é a transformação de modelos e esta é a origem do nome do projeto. Atualmente transformação de modelos continua sendo uma operação essencial, mas outro modelo de gestão como

composição de modelos também estão sendo propostos, ampliando o escopo do projeto GMT. [20]

7.7. AndroMDA

Em poucas palavras, o AndroMDA [29] é um framework open source MDA que combina modelos (normalmente modelos UML armazenados em XMI produzido a partir de ferramentas CASE), com plugins AndroMDA (cartuchos e bibliotecas de tradução) e produz uma quantidade de componentes personalizados.

Com ele é possível gerar componentes para qualquer linguagem de programação (Java, .Net, HTML, PHP, etc.). Só é necessário escrever ou personalizar algum plugin existentes para suportar suas necessidades.

7.8. OLIVANOVA

OLIVANOVA - The Programming Machine [27] é o primeiro software comercialmente disponível que gera aplicações completas a partir de modelos de software. Ao contrário de outras soluções, o OLIVANOVA não se limita à construção de sistemas embarcados, infra-estrutura de banco de dados, ou de integração. Em vez disso o OLIVANOVA suporta modelos de classe, modelos funcionais e modelos de apresentação criando um aplicativo de software totalmente funcional e executável em questão de minutos.

O OLIVANOVA suporta uma variedade de diferentes plataformas (como o Windows NT, Windows 2000, Windows XP, Windows 2003, Linux), arquiteturas (como Web, Cliente/Servidor, Integração com Mainframes, Windows Service) e linguagens de programação (como JSF, EJB, C#, ASP.NET)

7.9. Objectif

ObjectiF é uma ferramenta para Desenvolvimento de Software Dirigido a Modelos com UML e BPMN em Java, C#, C++, BPEL, XSD e WSDL. Com ele é possível automatizar o seu desenvolvimento de software, baseado em UML e em conformidade com os princípios do Model-Driven Development.

Objectif oferece tudo que você precisa para o desenvolvimento orientado a modelos de aplicações em uma equipe de forma eficiente e altamente ágil. Quer se trate de uma aplicação complexa em uma organização com arquitetura orientada a serviços (SOA), um Rich Internet Application (RIA), software embarcado ou uma aplicação técnica, Objectif fornece suporte adequado. [22]

7.10. MOSkitt

Modeling Software KIT (MOSKitt) [23] é uma ferramenta CASE gratuita, construída sobre o Eclipse, que está sendo desenvolvida pela Conselleria de Infraestructuras, Territorio y Medio Ambiente [24] para apoiar a metodologia gvMétrica [25]. A qual utiliza técnicas baseadas na linguagem de modelagem UML.

A arquitetura de plugins do MOSKitt o torna não apenas uma ferramenta CASE, mas também uma plataforma de modelagem livre para desenvolver este tipo de ferramentas.

MOSKitt está sendo desenvolvido no âmbito do projeto gvCASE. Um dos projetos integrados ao gvPontis [26], projeto global da Conselleria para migração de todo seu ambiente tecnológico para Software livre.

8. Resultados

Nessa sessão serão demonstrados os resultados obtidos na condução da revisão sistemática, seguindo os objetivos e a questão proposta. Vale lembrar que a revisão tinha como objetivo levantar os requisitos presentes em ferramentas MDD que satisfizessem as ares de processo (TS e RD) do CMMI.

Após a investigação detalhadas das ferramentas e trabalhos selecionados, foi possível obter as características presentes em cada um destes e analisar o estado da arte no que diz respeito a ferramentas MDD.

As características destacadas foram:

- **Multiplataforma:** Ferramenta capaz de executar em diferentes sistemas operacionais.
- **Modelagem de Negócios:** Ferramenta que possibilita a modelagem de fluxos e regras de negócios
- **Editar modelos:** Ferramenta capaz de alterar modelos
- **Validar modelos:** Ferramenta com validações e verificações de consistência de modelos
- **Testar modelos:** Ferramenta capaz de realizar testes sobre os modelos
- **Refatorar modelos:** Ferramenta capaz de refatorar modelos
- **Modelagem com textos:** Ferramenta que suporta a criação de modelos através de textos de domínios específicos
- **Suporte a UML**
- **Modelos criados a partir de códigos em linguagens de programação**
- **Gerar documentação:** Ferramenta capaz de gerar relatórios e documentação a partir dos modelos
- **Importar modelos:** Ferramenta capaz de importar modelos de outras ferramentas ou a partir de arquivos
- **Modelos XMI:** Suporte a modelos escritos na linguagem XMI
- **Controle de versão**

A tabela 1, mostra uma matriz das ferramentas, citadas na sessão 7, e as características que cada uma possui. A partir dessa tabela, analisamos quais características serão relevantes para uma nova ferramenta e montamos um diagrama de casos de usos, mostrado na figura 6. Tal diagrama será utilizado para construção de uma nova ferramenta.

Tabela 1 - Ferramentas e suas características

Ferramenta	Eclipse Modeling Project	Together Borland	OptimalJ	Objecteering	MDDi	GMT Project	AndroMDA	OLIVANOVA	Objectif	MosKitt
Multi Plataforma	X		X		X	X	X	X		X
Modelagem de Negócios	X	X			X			X	X	
Editar Modelos	X	X	X	X	X			X	X	
Validar Modelos	X	X		X	X					
Testar Modelos	X	X		X						
Refatorar Modelos	X			X				X		
Modelagem com textos	X		X	X				X	X	X
Suporte a UML	X	X	X	X	X	X	X	X	X	X
Modelos ling. programação		X	X	X	X	X	X		X	
Gerar Documentação		X	X		X			X	X	X
Importar Modelos			X		X			X		
Modelos XMI		X	X		X			X		
Controle de Versão									X	X

Através da análise da tabela 1, notamos que algumas ferramentas se destacam por possuir grande parte das funcionalidades. Como é o caso da MDDi e da OLIVANOVA que possuem a maior quantidade de funcionalidades. Englobando ao todo, 69% das características citadas.

Em seguida, Eclipse Modeling Project, Together Borland e Objecteering, aparecem com 62% das características. As demais ficaram abaixo de 55% das características. Destacando-se AndroMDA e GMT Project que só possuíam 23% destas. A figura 4 demonstra o comparativo entre percentuais.

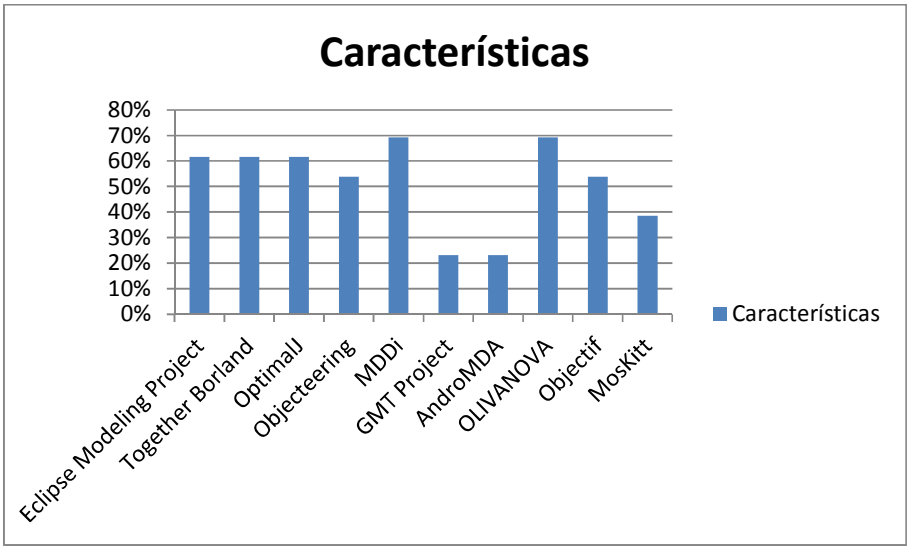

Figura 4 – Gráfico percentual de características por ferramenta

Em relação às funcionalidades levantadas, seis aparecem em mais de 60% das ferramentas. São elas: Multiplataforma; Editar modelos; Modelagem com textos; Suporte a UML; Modelos criados a partir de códigos em linguagens de programação; Gerar documentação. Destaca-se positivamente o Suporte a UML, registrado em todas as ferramentas. Já o controle de versão foi observado em apenas 20% das ferramentas.

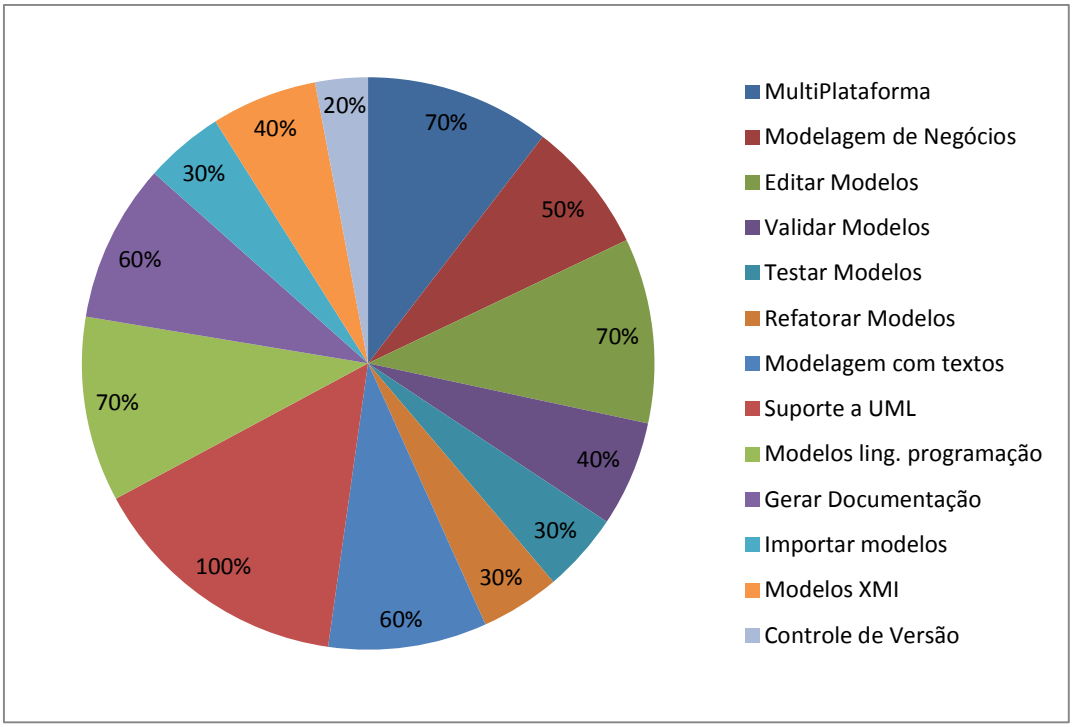

Figura 5 – Gráfico percentual de ferramentas por característica

Ao compararmos a quantidade de funcionalidades e quais dessas estão presentes em grande parte das outras ferramentas, podemos destacar o OLIVANOVA como ferramenta mais relevante. Porque além de apresentar mais funcionalidades, contém as funcionalidades mais presentes nos demais.

A fim de obter uma resposta mais precisa para nossa pergunta, consideraremos apenas as características presentes em mais de 40% das ferramentas levantadas. Com isso, não daremos relevância aos itens: Refatorar modelos, Testar Modelos, Importar modelos e Controle de versão.

Alguns pontos entrariam como requisitos não funcionais. Seriam eles: Multiplataforma, Suporte a UML e Modelos XMI. Os demais estão presentes no diagrama de casos de uso apresentado na figura 6.

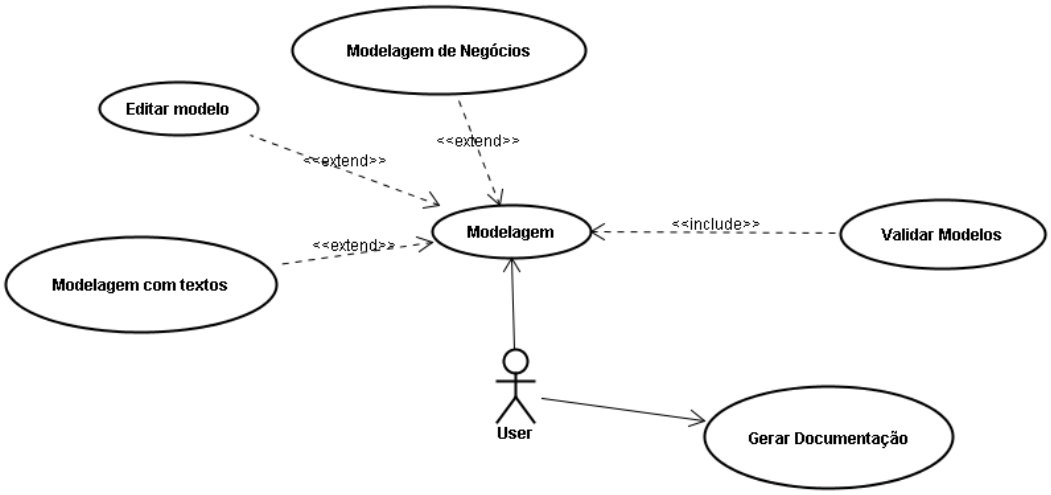

Figura 6 - Diagrama de Casos de Uso

9. Conclusão

Neste trabalho, tivemos como resultado inicial poucas ferramentas, o que levou a uma quantidade pequenas de ferramentas a serem analisadas a fundo. Uma reformulação na consulta poderia melhorar esse resultado, trazendo mais dados para análise.

No entanto, isso não afetou a qualidade dos requisitos levantados, pois, as ferramentas analisadas contemplavam em sua maioria requisitos relevantes para o tema proposto.

Através do quadro gerado e dos gráficos foi possível obter uma visão geral das ferramentas e suas características. Tal quadro pode auxiliar engenheiros na escolha da ferramenta a ser utilizada, ou quais requisitos selecionar para implementação de uma nova ferramenta, tarefa que será realizada, futuramente, baseada no diagrama de casos de uso elaborado.

10. Referências

[1] BIOLCHINI, J., et al; Systematic Review in SoftwareEngineering. 2005.

[2] HAUG, T. H.; A Systematic Review of Empirical Research on Model-Driven Development with UML. Tese de Mestrado. Fevereiro, 2007.

[3] Blog do CMMI. Disponível em: <http://www.blogcmmi.com.br/>. Acesso em Setembro de 2011.

[4] LONIEWSKI, G; INSFRAN, E; ABRAHÃO, S.; A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development. In MODELS'10 Proceedings of the 13th international conference on Model driven engineering languages and systems: Part II . 2010

[5] FONDEMENT, F.; SILAGHI, R.; Defining Model Driven Engineering Processes. In Proceedings of WISME. 2004.

[6] MELLOR, S. J.; CLARK, A. N.; FUTAGAMI, T.; Model-Driven Development. IEEE Software. IEEE Computer Society. 2003

[7] MACDONALD, A.; RUSSEL, D.; ATCHISON, B.; Model-driven Development within a Legacy System: An industry experience report. In Proceedings of the Australian Software Engineering Conference (ASWEC'05). 2005

[8] FRANCE, R.; RUMPE, B.; Model-driven Development of Complex Software: A research Roadmap, Future of Software Engineering (FOSE '07). IEEE Computer Society. 2007

[9] DUBY, C.; Accelerating Embedded Software Development with a Model Driven Architecture. Pahtfinder Solutions. 2003

[10] Ambler, S. W.. Examining the Model Driven Architecture (MDA). Disponível em: <http://www.agilemodeling.com/essays/mda.htm>. Acesso em Outubro de 2011

[11] FOUSTOK, M.; Experiences in large-scale, component based, model-driven software development. 1st Annual IEEE Systems Conference. 2007

[12] KITCHENHAM, B.: Procedures for performing systematic reviews. Tech. rep., Keele University and NICTA. 2004

[13] Eclipse Modeling Project. Disponível em: <http://www.eclipse.org/modeling/>. Acesso em Outubro de 2011.

[14] Borland Together. Disponível em: < http://www.borland.com/br/products/together/index.html >. Acesso em Outubro de 2011.

[15] OptimalJ. Disponível em: < http://www.objectsbydesign.com/tools/oj/optimalj.html>. Acesso em Outubro de 2011.

[16] Objecteering. Disponível em: <http://www.objecteering.com/>. Acesso em Outubro de 2011.

[17] Centro Internacional de Tecnologia de Software. Disponível em: <http://www.cits.br/cmmi.do>. Acesso em Novembro de 2011.

[18] DROMOS Tecnologia e Gestão. Uma visão geral do CMMI. Disponível em: <http://www.dromostg.com.br/CMMI.PDF>. Acesso em Novembro de 2011.

[19] Eclipse MDDi proposal. Disponível em: <http://www.eclipse.org/proposals/eclipse-mddi/>. Acesso em Novembro de 2011.

[20] GMT Project. Disponível em: <http://www.eclipse.org/gmt/>. Acesso em Novembro de 2011.

[21] Sparx Systems. Disponível em: <http://www.sparxsystems.com/>. Acesso em Novembro de 2011.

[22] SOARES, M. ; VASCONCELOS, A. M. L. ; GUSMAO, C. M. G. ; CASTRO, Jaelson Freire Brelaz de . Ferramentas de Suporte a MDD: Um Quadro Comparativo. In International Conference on Information Systems and Technology Management, 2011, São Paulo. Anais do 8th CONTECSI - International Conference on Information Systems and Technology Management, 2011.

[23] MOSkitt. Disponível em: <http://www.moskitt.org/>. Acesso em Outubro de 2011.

[24] Generalitat Valenciana. Conselleria D'Infraestructures, Territori i medi ambiente. Disponível em: <http://www.cit.gva.es>. Acesso em Outubro de 2011.

[25] gvMétrica. Disponível em: <http://www.gvpontis.gva.es/fileadmin/conselleria/images/Documentacion/migracionSwAbierto/gvMETRICA/introduccion/gvMetrica_Introduccion.htm#d0e4>. Acesso em Outubro de 2011.

[26] gvPONTIS. Disponível em: <http://www.moskitt.org/eng/gvpontis/>. Acesso em Outubro de 2011.

[27] OLIVANOVA. Disponível em: <http://www.care-t.com/index.asp>. Acesso em Outubro de 2011.

[28] Grupo do OLIVANOVA no LinkedIn. Disponível em: <http://www.linkedin.com/groups?about=&gid=3594463&trk=anet_ug_grppro>. Acesso em Outubro de 2011.

[29] AndroMDA. Disponível em: <http://www.andromda.org>. Acesso em Outubro de 2011.

[30] SEI - CMMI. Disponível em: <http://www.sei.cmu.edu/cmmi/>. Acesso em Outubro de 2011.

[31] UML. Disponível em: <http://www.uml.org/>. Acesso em Novembro de 2011.

[32] BATE R., et al; A Systems Engineering Capability Maturity Model, Version 1.1. Technical Report. Software Engineering Institute, Carnegie Mellon University. Fevereiro, 1993.

[33] PAUL M. C., et al; Capability Maturity Model for Software, Version 1.0. Software Engineering Institute, Carnegie Mellon University. Dezembro, 1994.

[34] CMMI Product Team; CMMI for Development, Version 1.3. Technical Report. Software Engineering Institute, Carnegie Mellon University. Novembro, 2010.