# CoMDD: uma abordagem colaborativa para auxiliar o desenvolvimento orientado a modelos

***David Fernandes Neto***

# CoMDD: uma abordagem colaborativa para auxiliar o desenvolvimento orientado a modelos

***David Fernandes Neto***

**Orientadora:** ***Profa. Dra. Renata Pontin de Mattos Fortes***

Monografia apresentada ao Instituto de Ciências Matemáticas e de Computação — ICMC/USP, para o exame de Qualificação, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

**USP - São Carlos**
**Janeiro/2012**

# Resumo

O

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas e Siglas

API *Application Programming Interface*

CDE *Collaborative Development Environment*

CAPÍTULO

1

# Introdução

Este capítulo descreve a área de colaboração do desenvolvimento orientado a modelos, na qual o trabalho está inserido, apresenta o problema de pesquisa e a justificativa para sua escolha, define a hipótese e o objetivo do trabalho. Os resultados esperados e o escopo do trabalho também são apresentadas aqui.

## 1.1 Descrição geral da área, o Problema e a Justificativa

O desenvolvimento orientado a modelos (*Model Driven Development* - MDD) é um método de desenvolvimento de software que foca na criação de modelos, ou de abstrações, como principais artefatos de desenvolvimento de software e nas transformações desses modelos para gerar código-fonte. Seu objetivo é reduzir a distância semântica que existe entre o domínio do problema e o domínio da implementação/solução, utilizando modelos mais abstratos que protegem os desenvolvedores de software das complexidades inerentes às plataforma de implementação (Teppola et al., 2009; France e Rumpe, 2007).

A principal vantagem do MDD é poder expressar modelos usando conceitos menos vinculados a detalhes de implementação, além do fato de modelos serem mais próximos do domínio do problema. Isto torna os modelos mais fáceis de se especificar, entender e manter. E, em alguns casos, ainda é possível os especialistas do domínio produzirem os sistemas ao invés dos especialistas da tecnologia de implementação (**??**)

E ainda são vários os benefícios que o MDD possui, como: produtividade, interoperabilidade, reuso, ... cite???. São diversos trabalhos (cite???) que propõe e que usam o MDD como abordagem para desenvolvimento de software. Ainda, entre as áreas da indústria que usam MDD, a de sistemas

embarcados já se beneficia desta abordagem antes mesmo do surgimento da UML ou da MDA[1] (Model Driven Architecture).

Entretanto, em relação à modelos textuais[2], como equipes de desenvolvimento de software trabalham?

Apesar de alguns trabalhos cite??? dizerem que as ferramentas para MDD são quase inexistentes; quando trabalha-se com modelos textuais as ferramentas para MDD são as mesmas que existem para as abordagens tradicionais como por exemplo Java, C, C++ e etc. Ou seja, assim como pode-se usar o Eclipse[3] como IDE de desenvolvimento e o SVN[4] para compartilhamento de código, pode-se usar ambas ferramentas para desenvolver usando MDD. Os modelos textuais são textos da mesma forma que um código-fonte Java é texto, assim não há razão técnica para não usar-se o uma IDE que suporte MDD e um sistema de controle de versão[5] qualquer.

Assim, uma possibilidade para trabalhar colaborativamente com o MDD é o uso de uma IDE associada à um sistema de controle de versão[6], entretanto, os sistemas de controle de versão podem causar conflitos quando dois ou mais desenvolvedores modificam o mesmo documento e submetem ao servidor[7].

Neste momento, desenvolvedores terão que dedicar certo tempo para resolver os conflitos, e às vezes esses conflitos só podem ser resolvidos manualmente. Além disso, para os desenvolvedores saberem como resolver o conflito corretamente eles precisam estar em contato entre si e para decidirem como realizar o *merge* precisam estar disponíveis no momento (ao Gustavo Prudêncio et al., 2012).

Ainda há um outro ponto ao se usar as IDEs e os SCV: a necessidade de instalação desses softwares, ou seja, para as equipes usarem uma IDE e um SCV é preciso que cada desenvolvedor tenha instalado esses softwares em seus computadores [8]. O que significa dizer que se um *stakeholder* estiver interessado em acompanhar o desenvolvimento de um software ele irá precisar instalar em seu computador essas ferramentas [9], o que pode-se dizer ser uma tarefa difícil e trabalhosa. Isso sem contar que o *stakeholder* deverá saber como funciona essas ferramentas. Se ele quiser participar da edição dos modelos[10] deverá ainda conhecer como funciona a geração de códigos da IDE,

[1] A MDA (Model Driven Architecture) é um pradão da OMG como abordagem do MDA, com uso da UML para a modelagem

[2] Ver capítulo de DSLs

[3] http://eclipse.org/

[4] http://subversion.tigris.org/

[5] Este trabalho usará os termos sistema de controle de versão ou SCV ou sistema de versionamento como sinônimos.

[6] Este trabalho está focando somente no uso de versionadores, pois o estudo de sistemas de gerenciamente de configuração tornaria a pesquisa muito extensa.

[7] Isso no caso de uma política de versionamento *otimistic*, a qual permite o desenvolvimento concorrente (Sarma et al., 2003).

[8] Será discutido no Capítulo XXX ferramentas que podem ser usadas na web, entretanto a maioria dessas ferramentas são incipientes e não são usadas nas empresas/academia.

[9] No experimento deste trabalho (descrito no capítulo XXX, dois profissionais experimentes levaram cerca de quarenta minutos para instalar e configurar o Eclipse e o plugin do svn

[10] Este trabalho parte do uso de MDD como abordagem de desenvolvimento e por isso está-se falando de MDD

como fazer *check-in*[11], *check-out*[12], entre outros termos e suas variações de nome[13]. De forma que, embora para um usuário não desenvolvedor trabalhar com modelos seja mais fácil que código-fonte cite?? essa vantagem é minimizada quando há a necessidade do uso dessas ferramentas por parte deste usuário não desenvolvedor.

Além do que, todo esse processo de instalação e de check-in/check-out e etc, é relativamente demorado, de forma que é mais uma desvantagem somada a necessidade de conhecimento desses conceitos. Isto fica claro no trabalho de Jones (Jones e Scaffidi, 2011) quando ele afirma que cientistas raramente usam controle de versão devido ao alto custo inicial para configuração desses sistemas.

Mesmo que o usuário ainda receba um treinamento, cada ferramenta é diferente uma da outra, por exemplo, o svn é diferente do git que é diferente do team foundantion service da microsoft, sendo não só necessário o conhecimento da instalação, uso, como também da lógica de funcionamento dessas ferramentos, pois elas tem diferentes abordagens, principalmente o git que possui o paradigma mais diferente desses.

Por outro lado, as wikis são ferramentas conhecidas por serem simples de usar e ágeis para colaboração cite??. De forma que dependendo do problema o uso de uma wiki ao invés de uma IDE associado a um SVN (no caso da abordagem MDD) pode resolver todos os problemas descritos até aqui, pois elas funcionam em um navegador, possuem um controle de versões com histórico, possibilidade para comentários e, no caso da Xwiki[14] não há problemas de conflito, pois ela permite travar a edição concorrente. Assim, falta apenas implementar na wiki um suporte a modelagem e geração de código-fonte.

Ainda, analisando o histórico de aplicações como músicas, vídeos, jogos, o armazenamento de arquivos, entre outros, observa-se uma tendência dessas aplicações migrarem web, como exemplo disso tem-se o Grooveshark®[15] (Reprodutor de músicas on-line), o Youtube® [16](reprodutor de vídeos on-line), o Dropbox®[17](Armazenador de arquivos on-line) e até mesmo IDEs estão convergindo para a web, como é o caso do Cloud9[18]. De modo que ter uma ferramenta possível de editar, compartilhar e versionar modelos, gerar código-fonte e ainda permitir a entrada de comentários pode beneficiar desenvolvedores e stakeholders.

Contudo, seria possível uma wiki ser usada como uma ferramenta de desenvolvimento e compartilhamento de modelos, da mesma forma que uma IDE e SCV fazem? Ou melhor, ela poderia atender os mesmos requisitos? A partir deste questionamente este trabalho definiu uma abordagem colaborativa orientada a modelos (*Collaborative Model Driven Development* - CoMDD), a qual

[11] operation adds a new file revision to that file's archive. A check-in is sometimes called a commit.

[12] copies the latest (if not otherwise specified) version of a file from the archive to a local hard disk

[13] Por exemplo, no plugin do SVN para o Eclipse, o *check-out* é quando faz-se uma cópia completa de todos os arquivos do servidor e *update* quando apenas atualiza as modificações - confirmar isso

[14] xwiki site ...

[15] grooveshark.com

[16] youtube.com

[17] dropbox.com

[18] http://c9.io/

usa uma wiki para apoiar o MDD em uma equipe de desenvolvimento nas tarefas de modelagem, geração de código, versionamento e comunicação.

## 1.2 Hipótese e objetivo

A partir do cenário levantado na seção anterior, definiu-se a seguinte hipótese que norteia este trabalho:

Para desenvolver um sistema colaborativamente usando MDD[19], é possível usar uma wiki ao invés do processo tradicional[20] de desenvolvimento.

O objetivo principal deste trabalho é tentar evidenciar que a hipótese de pesquisa é verdadeira para um caso específico (a ser tratado no estudo de caso) e incentivar o uso da abordagem desenvolvida neste trabalho para outros casos também (ver sessão a seguir: método).

## 1.3 Escopo

Não faz parte do escopo deste trabalho comparar o desenvolvimento tradicional (que usa linguagens de amplo propósito, como Java, por exemplo) o MDD, uma vez que há uma vasta literatura sobre isso. Este trabalho parte do princípio de que o MDD é mais vantajoso do que o desenvolvimento tradicional para o desenvolvimento de software cite???

Este trabalho cita sobre a comunicação entre os integrantes (sejam desenvolvedores ou *stakeholders* e entende que é extremamente importante promover e incentivar a comunicação, contudo, não é escopo deste trabalho desenvolver ferramentas ou um processo de comunicação entre os desenvolvedores, mas apenas de incentivar a partir da forma que o CoMDD é concebido e pelo uso de mensageiros instântaneos ou ferramentas de vídeo-conferência.

## 1.4 Resultados Esperados

Espera-se que com este trabalho o uso de MDD seja cada vez mais empregado no desenvolvimento de sistemas e de que as Wikis evoluam de forma que tenham mais recursos estilo IDEs, até que um dia a computação tenha o suporte em termos de ferramentas e processo para desenvolver softwares de grande porte e complexos usando Wikis.

[19] No caso estamos nos referindo a dsls como abordagem de mdd. No capítulo de revisão bibliográfica isso ficará mais claro

[20] Definimos processo tradicional de desenvolvimento como sendo o uso de uma IDE e de um sistema de controle de versão *otimistic*

## 1.5 Estrutura deste trabalho

No capitulo 2 faz-se um levantamente bibliográfico dos trabalhos relacionados e uma revisão bibliográfica dos conceitos abordados no trabalho. No capítulo 3 tem-se a o desenvolvimento do trabalho, características do CoMDD, estudo de caso e resultados. Por fim, o trabalho apresenta as considerações finais e conclui a pesquisa no capítulo 4. -> Isso vai mudar.

CAPÍTULO

2

# Trabalhos Relacionados

Artigos

Programas semelhantes já em uso: sugestao do michetti, aquele que o chiquito mandou na lista, ...

Ferramentas de MDD: eclipse, MPS, ...

Ferramentas que auxiliam o trabalho colaborativo (sharepoint)...

CAPÍTULO

3

# Revisão Bibliográfica

## 3.1 Model Driven Development

Definição ...

Pq MDD? Vantagens do uso de Modelos (ainda é possível checar se um modelo é satisfatório ou nao, mas um código nao é possível) e do MDD, A importancia do MDD

Justificativa de MDD na abordagem: Modelos sao colaborativos e mais faceis de entender por nao desenvolvedores,

Uma criança pode programar usando uma wiki mas nao usando eclipse e svn, mesmo que no eclipse seja modelos. A crianca é do dominio e o desenvolvedor o especialista// nivel da solução: java, c .... nivel do problema: UML, p ex //

1-Código-fonte é mais complicado de ser desenvolvido por mais de uma pessoa, logo é também menos reutilizável. Só o especialista consegue fazer e nao o cara do dominio. -> falar mal do desenvolvimento tradicional // 2- Modelos são mais colaborativos, simples, genéricos e reusáveis (nao falar em reuso pra nao criar uma expectativa errada pro leitor) e permitem que não especialistas possam desenvolver/acompanhar; alem de poder-se checar por conformidade os modelos

## 3.2 O desenvolvimento tradicional usando svns

Praticamente cada software usa seus conceitos sobre checkin e checkout ... Aqui apresentamos o do eclipse+svn que sera comparado com o comdd no capitulo de desenvolvimento

## 3.3 Domain Specific Languages

Uma abordagem de MDD...

### 3.3.1 Templates

## 3.4 Colaboração

o que é colaboração e quais seus benefícios?

Como as pessoas colaboram no desenvolvimento de software, Falar de como é feito o desenvolvimento tradicional colaborativo -> Arquitetura do desenvolvimento tradicional

## 3.5 Wiki

3- Artefatos online (wikis, gdocs) são mais colaborativos que controles de versao, bug tracks, ou outras ferramentas de gestao de configuração por terem interfaces mais amigáveis -> Assim a Web aumenta ainda mais colaboração: justificativa para wikis

Colaboracao e MDD: SVN

Colaboração na wiki

Assegurar a integridade e consistência de artefatos versionados num ambiente que suporte acesso concorrente é um problema difícil (Zhang e Ray, 2007), portanto para evitar os conflitos de edição e merge adotou-se uma abordagem *pessimistic*, ou seja, uma política em que o artefato é bloqueado quando já está sendo editado por outra pessoa. Aplicando essa política, conflitos de edição são evitados e com isso desenvolvedores não gastarão tempo ou esforços resolvendo conflitos. Ainda, a política *pessimistic* pode ajudar na comunicação, pois os desenvolvedores saberiam quando outra pessoa estaria modificando o mesmo artefato, incentivando a comunicação entre os desenvolvedores com os mesmos interesses (ao Gustavo Prudêncio et al., 2012; Sarma et al., 2003). Uma outra possível solução seria a edição simultânea estilo google docs citeMeu trabalho!...

CAPÍTULO

4

# Desenvolvimento

Este capítulo apresentará a abordagem desenvolvida neste trabalho, citará suas funcionalidades (com mais detalhes no anexo...), limitações....

## 4.1 O CoMDD - *Collaborative Model Driven Development*

O CoMDD é uma abordagem que consiste no uso de uma DSL para edição de modelos e geração de código-fonte, e de uma Wiki como ferramenta de suporte a DSL, possibilitar o trabalho em equipe e o versionamento de modelos.

O CoMDD permite:

1. **Edição de modelos:** criar ou alterar modelos de acordo com o metamodelo definido[1]. Os modelos serão a entrada do transformador e eles são editados na wiki

2. **Transformação de modelos em código-fonte:** o transformador, desenvolvido para a wiki, irá gerar o código-fonte de acordo com o modelo de entrada e apresentar erros de sintaxe[2]

3. **Colaboração:** a wiki permite que mais desenvolvedores possam editar o mesmo modelo usando uma política *pessimistic*, promovendo a colaboração

4. **Versionamento de modelos:** a wiki armazena um histórico dos modelos com a possibilidade de comparar as alterações e retornar à edições anteriores

[1] O metamodelo pode ser alterado na wiki, assim como as transformações, mas a princípio está sendo definido fora da wiki.

[2] Na atual versão do CoMDD, o transformador não retorna os erros sintáticos e sim os ignora. O transformador então retorna o código-fonte da parte dos modelos válidas

5. **Controle de acesso por grupos**: a wiki permite criar um grupo com permissão de edição de modelos e outro com permissão de apenas visualização

6. **Comentários**: tanto é possível definir um comentário para cada versão alterada quanto na página editada (versão do modelo atual). A figura abaixo XXX ilustra vários comentários na página principal (lado esquerdo) e comentários específicos para versões (lado direito).

No anexo XXX estão detalhes da implementação.

Possíveis de serem implementadas: Merge (ver como a wikipedia faz e um plugin) Bate papo dentro da wiki Highlight com javascript Wiki como ide: ver outras características de uma ide Metametamodelagem: é possível alterar a propria gramatica, bem como as transformações ainda pela propria wiki, mas isso seria um script Falar da edição de subartefatos: numa wiki é muito simples voce criar uma pagina dentro de outra [[]]

## 4.2 Vantagens

O CoMDD tem as seguintes vantagens devido ao uso de uma wiki e de MDD:

1. **Produtividade:**

2. **DDS:**

3. **Comunicação**

4.

## 4.3 Arquitetura do CoMDD

Os passos para criar a dsl e para exportar para xwiki e sua interação com o usuário. –> Isso vai no anexo. A

## 4.4 Método

Para tentar evidenciar que a hipótese é possível/verdadeira o CoMDD foi implementado usando o ANTLR[3] e a Xwiki[4] (ver detalhes de implementação no Apêndice A) e dois estudos de caso foram planejados.

[3] http://www.antlr.org
[4] http://www.xwiki.org

### 4.4.1 Estudo de Caso I:

O objetivo do primeiro estudo de caso foi avaliar o CoMDD por pessoas não desenvolvedoras.

#### Descrição dos participantes

Alunos de pré-iniciação científica do Laboratório de Robótica Móvel da Universidade de São Paulo. Três deles tinham 16 anos e um 17 anos, sendo dois garotos e duas garotas. Todos os quatro haviam concluído o segundo colegial. Já tinham programado para o TAZ[5] usando uma dsl própria. Não possuiam conceitos de algoritmos ou de programação orientada a objetos.

#### Instruções e Problema Apresentado

Foi formado duas equipes com dois integrantes. Cada equipe ficava em um único computador, ou seja, eram dois computadores com duas pessoas em cada. Neste momento, o intuito era avaliar a linguagem e o ambiente do CoMDD, verificando se participantes jovens, com relativa baixa formação, sem conhecimentos de programação e sem conhecimentos do domínio, conseguiriam entender os conceitos e a linguagem, por isso foram colocados juntos para que pudessem discutir amplamente.

Antes de passar o comando foi lido junto com os participanetes dois textos, o primeiro explicando o que é o CoMDD (ver *Texto I: O que é o CoMDD?* no apêndice A) e o segundo texto explicando como usar a DSL criada para o estudo de caso (ver *Texto II: Como programar para o CoMDD* no apêndice A.2).

O seguinte comando foi passado:

[5]Um robô de pequeno porte físico com intuito educaional

O código é de um robô que ao receber uma lista de coordenadas deve ser capaz de passar por elas. Seu algoritmo de funcionamento é o seguinte:

Os sensores devem ler a informação
O robô recebe a próxima coordenada
O robô processa a informação
O robô recebe uma regra a aplicar
O robô anda

Esse algoritmo deve ficar sendo executado eternamente, até o robô ser desligado.
Agora você deve a partir do algoritmo e das explicações no site ser capaz de escrever um código usando a linguagem do CoMDD.

Requisitos:
O robô deve ser da plataforma pionner
O robô deve usar os pacotes de localização e o do player
O robô deve usar um gps e uma bússola
O robô seguir um conjunto de coordenadas

O tempo entre o momento que o pesquisador começou a falar com os participantes até o fim foi de aproximadamente 2h45. Neste tempo inclui a instrução do pesquisador aos participantes, a resolução do problema e a avaliação feita pelos participantes.

Tiverem a ajuda do Jefferson, apenas para explicar algumas intruções que não ficaram claras o suficiente, coisa que eu poderia ter feito, mas pelo skype tava dificil.

### Resultados

A resposta correta é a seguinte:

```
plataforma pionner
robo david
adicionar includes
adicionar defines
importar pacote player;
importar pacote localizacao;
criarSensor gps
criarSensor bussola
    int main() {
        gps.ligar();
        bussola.ligar();
        carregarListaCoordenadas();
        inicializarPlayer();
        while(true) {
            bussola.ler();
            gps.ler();
            receberCoordenada();
            processaInfo();
            defineRegraSeguirMultiplasCoordenadas();
            andar();
        }
    }
```

### Resultados das equipes - aqui tinha que entrar mais um sub

A equipe 1 apresentou a seguinte resposta, observando que a identação apresentada é proporcional à original (leu duas vezes a "api"):

```
plataforma pioneer
robo MLJ123
adicionar defines
adicionar includes
adicionar while

importar pacote player;
importar pacote localizacao;

criarSensor gps
criarSensor bussola

int main() {
    «
    gps.ligar();
    bussola.ligar();
    carregarListaCoordenadas();
    inicializarPlayer();
  while(true) {
   gps.ler();
    bussola.ler();
    carregarListaCoordenadas();
    receberCoordenada();
    processaInfo();
    defineRegraSeguirMultiplasCoordenadas();
    andar();
    }
    »
}
```

A equipe 2 apresentou a seguinte resposta, observando que a identação apresentada é proporcional à original (eu estava conversando todo o tempo no skype com essa dupla):

```
plataforma pioneer
importar pacote player;
importar pacote localizacao;
criarSensor gps
criarSensor bússola
carregarListaCoordenadas();
int main()
«Inicializações
Loop
Funções
»
```

O grupo A teve resultados melhores que o grupo B.

### Opiniões dos participantes

*Prós:* A linguagem é fácil de entender e o fato de ser em português facilita.

*Contras:* Foi mais complicado entender as intruções do experimento; ficou confusa a função *carregarListaCoordenadas()*.

### Ruídos e mal planejamento

O experimento foi passado a distância, via skype e com a ajuda de um aluno de doutorado do LRM que acompanhou os participantes*****ajudou???****

### Conclusões e observações

1. Ao observar as equipes estima-se a necessidade de pelo menos duas horas para que eles aprendessem a usar e instalar o Eclipse e SVN;

2. Os participantes tinham dificuldade ou preguiça de lerem o texto que ensinava como programar para o CoMDD (A.2) e tinha o hábito de perguntarem constantemente se o que faziam estava certo;

3. Os participantes copiavam do texto A.2 e colavam na página de edição;

4. Para pessoas que nunca programaram fica difícil elas entenderem o funcionamento da linguagem. Era esperado que elas compreendessem mais rapidamente a linguagem.

### Lições aprendidas

Como aprendizado do estudo de caso I é necessário planejar melhor em dois aspéctos:

- Organizar melhor a infra-estrutura: verificar quantos computadores serão necessários, se a comunicação for on-line preparar os softwares necessários e orientar bem uma pessoa que possa ajudar e que esteja presente com os participantes. No caso deste experimento houve um mal planejamento da infra-estrutura e da orientação ao ajudante;

- Para um experimento deste tipo o ideal é evitar ajudar os participantes para se ter uma opinião mais real. No caso deste experimento há quatro hipóteses que explicasse a dependência dos participantes com relação ao pesquisador: (i) relativa baixa instrução dos participantes, (ii) ruídos, (iii) mal planejamento por parte do pesquisador e/ou (iv) a disposição do pesquisador para tirar dúvidas.

### 4.4.2 Estudo de Caso II

O objetivo do segundo estudo de caso foi avaliar comparativamente o CoMDD com a abordagem Eclipse+SVN, entre desenvolvedores e coletar suas opiniões.

#### Descrição dos participantes

Todos os quatro participantes são formados em ciências da computação, atuaram na indústria desenvolvendo sistemas e estão cursando mestrado ou doutorado na Universidade de São Paulo em ciências da computação.

#### Instruções e Problema Apresentado

Os quatro integrantes foram divididos em duas equipes: equipe A (usando o CoMDD) e equipe B (usando Eclipse+SVN). Ambos integrantes de cada equipe estavam em locais (mais especificamente cidades) diferentes na hora do experimento. A equipe A realizou o experimento em um dia diferente da equipe B.

Antes de passar o comando todos os grupos receberam as seguintes instruções (depois cada grupo recebeu suas instruções específicas):

- Ler o *Texto I: O que é o CoMDD?* (ver A.1) no caso da equipe A e ler o *Texto IB: Instalação do Eclipse+SVN e importação da DSL* (ver A.3);

- Ler o *Texto II: Como programar para o CoMDD* (ver A.2);

- Pode-se comentar o modelo usando "//";

- A função carregarListaCoordenadas() já está configurada para a lista de coordenadas, desta forma vocês não devem se preocupar em passar um lista de coordenadas, apenas chamar a função;

- O desenvolvimento é feito em turnos e em cada turno apenas um desenvolvedor pode trabalhar no código (editá-lo), para evitar conflitos e merges. Cada turno tem 3 minutos para desenvolver e mais 2 minuto para comentar e fazer o commit, ou comentar no CoMDD. Ex: 0'-3': O dev1 começa programando enquanto o dev2 fica esperando (o dev2 pode conversar via e-mail ou gtalk com o dev1 caso o dev1 precise). 3'01s - 5': o dev1 tem de salvar suas alterações e comentar o código caso ache necessário. 5'01s - 6': ninguém faz nada para que o servidor atualize os dados 6'01s-9': o dev2 assume o comando de edição do código e o dev1 fica aguardando ou conversando via e-mail ou gtalk com o dev2 caso este precise 9'01s-11': o dev2 tem de salvar suas alterações e comentar o código caso ache necessário. 11'01s-14': tudo se repete ...

O seguinte comando foi passado:

O código é de um robô que ao receber uma lista de coordenadas deve ser capaz de passar por elas. Seu algoritmo de funcionamento é o seguinte:

Os sensores devem ler a informação
O robô recebe a próxima coordenada
O robô processa a informação
O robô recebe uma regra a aplicar, de acordo com os requisitos
O robô atua no ambiente

Esse algoritmo deve ficar sendo executado eternamente, até o robô ser desligado.
Agora vocês devem a partir do algoritmo e das explicações no site serem capazes de escrever um código usando a linguagem do CoMDD.

Requisitos:
O robô deve ser da plataforma pionner——————————————————— Dev1
O nome do robê deve ser uma concatenação dos nomes dos desenvolvedores———- Dev2
O robô deve usar os pacotes de localização——————————————— Dev1
O robô deve usar os pacotes do player———————————————————— Dev2
O robô deve usar um gps————————————————————————- Dev1
O robô deve usar uma bússola——————————————————————- Dev2
O robô seguir um conjunto de coordenadas———————————————— Dev1 + Dev2

### Equipe A: CoMDD

Além das instruções comuns apresentadas na seção 4.4.2, para equipe A foram apresentadas as seguintes intruções:

1. Pode-se usar todas as funcionalidades da wiki como comentários, histórico e comparação de versões[6];

2. Dêem preferencialmente para comunicação que a própria wiki fornece: os comentários, caso contrário usem um instant messanger ou o e-mail e depois fornecer o log para o pesquisador

### Equipe B: Eclipse+SVN

Além das instruções comuns apresentadas na seção 4.4.2, para equipe B foram apresentadas as seguintes intruções:

1. Pode-se usar todas as funcionalidades do plugin SVN como check-in check-out, comentários, histórico e comparação de versões[7];

2. Dêem preferencialmente para comunicação que o próprio SVN fornece: os comentários, caso contrário usem um instant messanger ou o e-mail e depois fornecer o log para o pesquisador

3. Não usar as funções comuns de uma IDE como highlight, auto-complete e etc[8][9]

### Resultados

A resposta correta é a mesma apresentada na seção...

### Resultados da Equipe A aqui entra mais um sub

A equipe A apresentou a seguinte resposta:

[6]Um exemplo de edição normal (sem ser usando a DSL) foi apresentado para que os participantes pudessem saber como e onde editar, comentar e comparar versões.

[7]Os participantes já conheciam o plugin.

[8]Assim as equipes ficam mais iguais, uma vez que no CoMDD essas funcionalidades não foram implementadas.

[9]O Eclipse tem um ambiente que não usa as funções citadas.

```
//O robô deve ser da plataforma pionner - Danilo
plataforma pionner

//O nome do robê deve ser uma concatenação dos nomes dos desenvolvedores - Davi
robo davnilo

//O robô deve usar os pacotes de localização - Danilo
importar pacote localizacao

//O robô deve usar os pacotes do player - Davi
importar pacote player;

//O robô deve usar um gps - Danilo
criarSensor gps

//O robô deve usar uma bússola- Davi
criarSensor bussola

//função principal do código
int main() {
bussola.ligar();
gps.ligar();

//O robô seguir um conjunto de coordenadas - Danilo + Davi
while(true) {
bussola.ler();
gps.ler();
receberCoordenada();
processaInfo();
defineRegraSeguir();
defineRegraNaoBater();
andar();
}
}
```

Comunicação do grupo A: colocar o historico da wiki - ver figura

O tempo entre o momento que o pesquisador começou a falar com os participantes até o fim foi de aproximadamente 1h45. Neste tempo inclui a instrução do pesquisador aos participantes, a resolução do problema e a avaliação feita pelos participantes.

Em um momento eu indiquei que o modelo tinha erros e ainda assim, no final ainda faltou algumas coisas no modelo.

O danilo teve uma duvida e eu sugeri ele ler a api ou perguntar para a dupla dele, mas perguntaram menos do que no estudo de caso 1

Sugeri para eles conversarem entre si e lerem a API

O tempo não era muito rígido, mas apenas uma pessoa editava por vez durante um turno.

O modelo apresentado no final foi o seguinte:

<>

### Resultados da Equipe B aqui entra mais um sub

A Equipe B apresentou a seguinte resposta:

```
// Definindo a plataforma do robo como pioneer
plataforma pioneer

// Define o nome do robo como uma concatenacao dos nomes dos desenvolvedores
robo filipeAlexandre

adicionar includes
adicionar defines

importar pacote localizacao;
importar pacote player;

criarSensor gps
criarSensor bussola

int main() {

    // Ligar o GPS
    gps.ligar();

    // Ligar Bussola
    bussola.ligar();

    // Carrega lista de coordernadas e inicializa Player
    carregarListaCoordenadas();
    inicializarPlayer();

    // Define que robo ira seguir um conjungo de coordenadas
    defineRegraSeguirMultiplasCoordenadas();

    // Loop
    while(true) {
        // Le os dados atuais
        gps.ler();
        bussola.ler();

        // Processa informacoes
        processaInfo();
        receberCoordenada();

        andar();
    }
}
```

Conversa do grupo B: ver figura

O tempo entre o momento que o pesquisador começou a falar com os participantes até o fim foi de aproximadamente 2h45. Neste tempo inclui a instrução do pesquisador aos participantes, a resolução do problema e a avaliação feita pelos participantes.

O grillo e o michetti perguntou se sempre vai ter que adicionar defines, includes, robo michetti, plataforma pioneer Eles pensaram que era um requisito por turno, mas esclareci que não. O tempo foi mais rigido e eles conversaram pouco pelo skype, até que os instruí para que usassem algum instant messanger ao invés de uma voice conference. Este grupo está usando um instant messagener, coisa que o grupo anterior não usou. Abaixo o log da conversa: ver apendice **??**

Depois do experimento elas são questionadas (ver questionário) e por fim são apresentadas para a outra abordagem, a que não usaram.

equipe 1... equipe 2... O grupo A, que leu pela segunda vez a API teve melhores resultados que o grupo B.

### Opiniões dos participantes

### Equipe A

aqui entra mais um sub A seguir são apresentadas as perguntas feitas aos participantes da Equipe A com suas respectivas respostas:

**O que vcs acharam da linguagem?** Danilo: tá legal mas as funcoes estão muito genéricas, includes e defines estao confusos (redundante a parte de adicionar includes e importar pacotes) Davi: linguagem primitiva, abrir fechar parênteses e chaves, os pacotes já deveriam estar todos inclusos sem a necessidade de importar

**Vocês mudariam algo na sintaxe?** Danilo: a estrutura da linguagem é comum a C, Java ao contrario do begin. É melhor deixar tudo em português ou tudo em inglês. Davi: a sintaxe esta boa para quem consegue programar em C ou java, aí consegue entender, mas para quem nunca programou (), e ; não significa nada.

**Os comentários que vocês escreveram foram úteis de alguma forma?** Danilo: interessante. Mas no mundo real um chat ou gtalk é mais útil, caso os dois estejam programando na mesma classe. Mas para documentação ou quando os desenvolvedores não estão ao mesmo tempo e na mesma classe, é útil. Davi: o sistema de comentário do svn funciona melhor, porque ao dar o commit você é obrigado a comentar e ao dar o check out você é obrigado a ler. Na wiki fica dissociado do código.

**E o histórico, foi útil?** Danilo: Não usou. É mais util quando alguém faz alguma cagada. Davi: não usou

**De uma maneira geral, o que vocês acharam da abordagem?** Dan: não é muito interessante uma vez que tenho que esperar pela pessoa terminar programar para depois programar. No tradicional os dois podem programar juntos e depois fazer o merge ou edição simultanea estilo o

google seria interessante. **E se fosse em classes separadas ao memso tempo?** Aí é interessante. Programar na web também é bom, o lance do histórico tambem. É bom a flexibilidade de estar em qualquer computador. Davi: programar na web é bom porque evita problemas de Infraestrutura (a infraestrutura é instantanea, independente do computador).

**Então seria interessante uma wiki ou seria melhor um outro site?** Danilo: a wiki um site mais dinamico, mais facil de editar, as pessoas já conhecem. Davi: poderia ser um site mais especializado

**O que voce achou da abordagem comdd e da abordagem eclipse+svn? Que comparação vocês fazem entre o eclipse+svn e o CoMDD? Qual vocês preferem?** Danilo: estar na web é mais flexivel, configurar uma ide, svn e tudo o mais é trabalhoso Davi: em geral é melhor programar na wiki do que usar uma ide, mas fica dependente da conexão. Aí seria interessante ter uma forma de usar a wiki mesmo seminternet.

**Vocês usariam uma wiki no lugar do eclipse+svn? Por que não? E se fosse para fazer programas como este, vocês usariam o CoMDD?** Danilo: usaria desde que o processo de compilação funcionasse, agora sem compilar e sem executar não. Davi: concordo.

### Equipe B

aqui entra mais um sub A seguir são apresentadas as perguntas feitas aos participantes da Equipe A com suas respectivas respostas:

**A empresa de voces seguia essa abordagem?** Usam o eclipse e o subversion

**O que vcs acharam da linguagem?** Michetti: fácil e simples, poucos comandos Grillo: curva de aprendizagem rapida, mas faltou pratica, não sei se o codigo funciona, a sintaxe é simples outra coisa seria melhorar a documentação para incluir qual o simbolo que indica comentário, e quando se deve usar ou não ponto e virgula

**Vocês mudariam algo na sintaxe?** Grillo e Michetti: a parte dos includes poderia ser uma linha só já que sempre vai precisar dela poderia ser numa linha.

**Como voces colaboraram? O que foi mais importante, os comentarios?** Eles: O comentário do log não usaram porque o codigo é pequeno e usaram apenas os comentarios no codigo. Michetti: o comentário do svn é mais quando tem algum problema e não para notificar o que foi feito.

**Os comentários que vocês escreveram no svn foram úteis de alguma forma?** Não usaram

**E o histórico, foi útil?** Não usaram

**E quais funcoes do SVN voces acham importantes alem das que voces usaram?** Michetti: Histórico caso tenha algum problema, merge caso editassem ao mesmo tempo. Grillo: quando o codigo é muito extenso é importante comparar as versoes de código para ver o que tá sendo alterado (as versoes do repositorio, com a local, o que alterou...)

**Voces acham que a wiki atenderia esses propositos que o svn+eclipse trás?** M: para editar código eu nunca pensei numa wiki para isso, usaria mais para colaboração na documentação. G: auxilio de sintaxe no eclipse, auto-complet, import automatico, clicar em um arquivo e ir para outro arquivo. A parte de comentário, versionar, alterar o texto é praticamente a mesma coisa. Não sei se caso editasse ao mesmo tempo não sei como se comportaria.

**O que voce achou da abordagem comdd e da abordagem eclipse+svn? Que comparação vocês fazem entre o eclipse+svn e o CoMDD? Qual vocês preferem?** M: to acostumado com o eclipse+svn, mas uso tambem o controle de versao integrado com um bugtrack. G: cor da grama, muita coisa já tem pronta pro eclipse, costume principal barreira, pensando como programador ainda usaria o eclipse. Tá, mas pensando que usariamos uma dsl e não java, logo seria algo bem mais simples.

**Seria bom todos esses programas para matar uma formiga?** Para um projeto pontual e pequeno o comdd supriria mas para algo maior e complexo.

### Ruídos e mal planejamento

### Conclusões e observações

Quanto a colaboração do grupo A: Foram instruídos a usarem os comentarios da wiki e puderam comentar no modelo usand

Quanto ao uso do svn o grupo B: usou Check in, check out, update, não merge ou comparar codigo

no experimento acho que as pessoas colaboraram igualmente, dando ainda um ponto positivo para o svn que "obriga"a pessoa a comentar, contudo fica muito mais facil da pessoas ler um comentário na wiki, só que com uma wiki é muito mais facil e simples de qualquer pessoa colaborar, bastando para isso ter permissões.

Para atender um determinado dominio, a wiki é o suficiente, para coisas simples, para algo mais complexo e que necessite de varias pessoas editando ao mesmo tempo e que podem dar erro, ai talvez o melhor seja a tradicional.

### Lições aprendidas

Como aprendizado do estudo de caso II ...

Será pedido para duas equipes diferentes a mesma tarefa, sendo que uma delas usara o eclipse+svn e a outra uma wiki apenas. Ambas usarão dsls e geração de código, mas nao terao os recursos tradicionais de uma ide como import, highlight sugestao ...

Para isso será implementado a dsl no eclipse e numa wiki usando o xtext/xpand e o antlr respectivamente.

### 4.4.3 Estuo de caso 3: análise de vídeos e planilha de ticar

BLAHBLHABLHA...

## 4.5 Limitações e Soluções

Deficiencia do CoMDD: nao compila, entao subentende-se que o codigo que ela gera esta sintaticamente correto, mas ela pode retornar erros

como integrar testes? debug? ou seria isso inexistentes em dsls ja que elas nao tem um compilador?

### 4.5.1 Do trabalho

Escopo do trabalho: avaliar em que casos pode-se usar uma wiki no lugar de um svn nao faz parte do escopo deste trabalho

A abordagem pessimistica possui suas desvantagens como "In contrast, the pessimistic policy has some disadvantages. First, locks can cause an unnecessary serialization over the development process. Sometimes different developers can independently modify different parts of the same CI, but using a lock-based approach that is not possible. In addition, locks can cause a false sense of security (Collins-Sussman et al., 2004). Developers may think that their work will not be affected by the work of other developers due to the locks. However, sometimes CIs rely on other CIs that can be modified by other developers. This situation leads to indirect conflicts (Sarma et al., 2003). (ao Gustavo Prudêncio et al., 2012) Assim, um trabalho futuro seria no estudo de estratégias otimísticas ou até mesmo na adoção de ambas como propõe o trabalho de Prudencio et al. (ao Gustavo Prudêncio et al., 2012).

### 4.5.2 Da ferramenta

Desvantagem da linguagem apontada por mim: o código gerado não pode ser testado uma vez que precisa usar o robô, mas o mais importante é ver como as pessoas interagem e colaboram.

### 4.5.3 Do estudo de caso

## 4.6 Contribuições do CoMDD

MDD e tudo na web: mais produtividade Modelos e wiki: aproximação maior de todos os envolvidos no processo, e com isso softwares mais próximos dos que o cliente precisa.

O Danilo perguntou se podia usar comentários. Embora a linguagem não dê suporte a isso eu deixei. Seria interessante a linguagem aceitar código C, C++ ou Java puro.

CAPÍTULO

5

# Conclusão

## 5.1 Considerações finais

O CoMDD é uma abordagem que prega mdd colaboartivamente. No caso foi usado uma wiki, mais especificamente a xwiki, mas poderia ser qualquer outra wiki ou plataforma web de simples uso e que promovesse velocidade. Com as atuais tecnologias a wiki é que melhor se encaixa nesse perfil. Em relação ao domínio também, espera-se que para qualquer domínio o CoMDD possa servir, entao no caso a dsl foi para robos autonomos moveis, mas também poderia ser para qualquer outro domínio // O comdd nao exige uma wiki necessariamanete, mas ela é tecnologia atual mais proxima da necessaria. Entretanto, muito ainda deve ser desenvolvido em cima das wikis para que elas possam de fato serem usadas como ferramentas de desenvolvimento de codigo, ao inves de apenas documentação ou afins.

Este trabalho está longe de criar um ambiente de desenvolvimento tão robusto e avançado quanto hoje são as ides, bug trackers, svns... até porque essas ferramentas já tem anos de maturidade. Este trabalho objetiva sim mostrar que o alinhamente de duas tecnologias aparentemente nao interligáveis é capaz de desempenhar papel semelhante e motivar novos estudos nesse caminho. O ideal seria transformar a wiki numa ide de fato, criar um merge de modelos, permitir o desenvolvimento simultaneo e a edição em tempo real que hoje tem o google. Mas a ideia de que quando voce edita um arquivo esta editando diretamente no servidor e portanto a cópia final simplifica (talvez agiliza tbm) o trabalho evitando conflitos de mege (afirmação forte). As wikis nunca foram pensadas para esse fim e por isso podem melhorar para este propósito.

É interessante para projetos menores, que tem a interedependencia entre suas classes nao muito grande, com poucas classes e que estas dependem pouco entre si, mas que necessitam fortemente

da colaboração de várias pessoas, e se torna mais interessante quando essa pessoas nao sao desenvolvedoras.

Ainda pode ser util para elaboração de algoritmos, mas que nao foi testado, ou até mesmo para código-fonte no caso da edição em tempo real.

concluir a hipotese da seguinte forma, mais ou menos: MDD é bom por causa ...

E Wikis sao boas por causa ...

E é possível integrar mdd e wiki para desenvolvimento (implementação e estudo de caso)

Portanto, ... afirmar a hipótese: 4-MDD na Wiki permite mais colaboração(wiki) e produtividade(MDD) (porque o comdd é mto mais acessível e pela sua simplicidade as pessoas podem colaborar mais facilmente), no desenvolvimento de software -> isso que tenho q avaliar , do que traria com eclipse e svn (ideal de avaliar)

Premissa I: o MDD implica em benefícios de produtividade para o desenvolvimento, modelos possibilitam mais comunicação principalmente por nao especialistas Premissa II: a wiki é excelente para comunicação, troca de conhecimento, rápida e dinâmica. Conclusão: um mdd na wiki traria beneficios associado ao uso de MDD com ao uso de uma wiki no desenvolvimento de sistemas

## 5.2 Contribuições

Será que o CoMDD traz inovação qnd comparado ao uso do eclipse+svn?

Quais suas contribuições?????

Que beneficios em termos de colaboração o comdd traz qnd comparado a abordagens eclipse+svn?

Que outros benefícios posso trazer?

As minhas contribuições estao totalmente relacionadas as contribuições do mdd e da wiki, por ex: por que as pessoas nao usam o drupal ou o joomla ou fazem um site para documentação? Da mesma forma. Nao estamos propondo o fim dos svns ou (mas sim das ferramentas locais, pois tudo pode ser web hoje, vide o c9 - ide na nuvem), mas estamos dizendo: olha, com uma wiki e uma dsl vc pode fazer mta coisa e colaborar mto mais.

O principal diferencial deste resultado em relação a outros trabalhos existentes é o aspecto colaborativo, permitindo a edição de modelos e a transformação de software por diferentes desenvolvedores sem a necessidade de ferramentas/ ambientes individuais instalados nas estações de trabalho dos desenvolvedores.

que beneficios o comdd traz quando comparado ao eclipse e svn? nao instalar programas vc ganha agilidade, maior participação, mais pessoas tem acesso a edição de maneira facilitada, ou seja, quando vc facilita o acesso e a edicao vc permite que mais pessoas possam colaborar. É só pensar no contrario para ver que é verdadeiro, pois se o desenvolvedor precisar do dobro de tempo para instalar um svn/ide e os mesmos forem 3 vezes mais dificeis de serem usados temos com isso menos pessoas nao desenvolvedoras participando. Talvez para o desenvolvedor nao importe

o quanto seja dificil usar um svn e o quanto isso va demandar tempo no desenvovlimento de software ( na realidade isso importa pq vai atingir diretamente na produtividade), mas para o nao desenvolvedor isso importa muito -> posso colocar uma pergunta assim no questionario dos videos.

## 5.3 Trabalhos Futuros

Usar uma wiki para densenvolvimento que nao seja mdd talvez nao seja tao interessante porque o nível de colaboração de um mesmo artefato é menor que este artefato como modelo. Mas talvez essa seja uma pergunta interessante (para que casos a colaboração numa wiki é eficaz?) a qual este trabalho não responde.

eu tenho que dizer que o estudo de caso foi o suficiente para provar ... (ser bem especifico), mas que para (aqui posso ser mais abrangente) é necessário melhorar a ferramenta (deixar ela mais proxima de uma ide) e partir para problemas mais reais/complexos.

implementações futuras: Edição das transformações e da gramática Retorno de erros (validação do modelo) Edição de subartefatos e que compoem um artefato que ira gerar o codigo Ed em tempo real Como testar o código gerado ou o modelo?

Como validar modelos?

Hightlight, auto sugestão e linkagem das palavras -> questoes tecnologicas de implementação

Definir a gramática e as transformações-> selado!

Retorno de erro-> aparentemente nao é complicado

Analisar para que casos é interessante usar o comdd ao inves do uso de uma ide+svn. Pois o estudo de caso mostrou que o comdd adequa as necessidades, contudo, não podemos generalizar tal afirmação. Ainda pode-se a partir da evolução das wikis para esse propósito verificar se realmente há a necessidade do uso dessas ferramentas locais, de merges ou dos modelos de versionamente atual.

## 5.4 Considerações Finais

## 5.5

# Referências Bibliográficas


FRANCE, R.; RUMPE, B. Model-driven development of complex software: A research roadmap. In: *29th International Conference on Software Engineering 2007 - Future of Software Engineering*, Minneapolis, MN, USA: IEEE Computer Society, 2007, p. 37–54.

GUSTAVO PRUDÊNCIO, J.; MURTA, L.; WERNER, C.; CEPÊDA, R. To lock, or not to lock: That is the question. *Journal of Systems and Software*, v. 85, n. 2, p. 277 – 289, special issue with selected papers from the 23rd Brazilian Symposium on Software Engineering, 2012. Disponível em: `http://www.sciencedirect.com/science/article/pii/S0164121211001063`

JONES, M.; SCAFFIDI, C. Obstacles and opportunities with using visual and domain-specific languages in scientific programming. In: *Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on*, 2011, p. 9 –16.

SARMA, A.; NOROOZI, Z.; HOEK, A. Palantir: raising awareness among configuration management workspaces. In: *Software Engineering, 2003. Proceedings. 25th International Conference on*, 2003, p. 444 – 454.

TEPPOLA, S.; PARVIAINEN, P.; TAKALO, J. Challenges in deployment of model driven development. In: *Software Engineering Advances, 2009. ICSEA '09. Fourth International Conference on*, 2009, p. 15 –20.

ZHANG, J.; RAY, I. Towards secure multi-sited transactional revision control systems. 2007, p. 365 – 375. Disponível em: `http://www.sciencedirect.com/science/article/pii/S0920548906000742`

CAPÍTULO

6

# Apêndice A

Antlr: conceitos e explicação da nossa gramática

CAPÍTULO

7

# Apêndice B

CoMDD: como implantar uma dsl em uma wiki?

CAPÍTULO

8

# Apêndice C

Eclipse: xtext/xpand

APÊNDICE

A

# Textos apresentados no experimento

Aqui são dispostos os textos apresentados no experimento e referenciados na seção 4.4

## A.1 Texto I: O que é o CoMDD?

**O que é o CoMDD?** O CoMDD vem da sigla Colaborative Model Driven Development. É basicamente uma linguagem de programação que funciona em uma wiki

**Certo, mas para que serve?** Com o CoMDD você pode programar robôs! Isso mesmo! Robôs!

**Então eu posso programar qualquer robô com ela?** Não. O CoMDD é uma dsl, ou seja, uma Linguagem Específica de Domínio (do inglês Domain Specific Language). O que significa dizer que com ela você não programa qualquer coisa como o C, C++, Java e etc. Com o CoMDD você programa apenas para um domínio muito específico, o de robôs móveis autônomos. E pelo fato do CoMDD ser um protótipo em desenvolvimento, mesmo para o seu domínio ele é pouco tão versátil.

**Tudo bem então, agora que entendi o que é o CoMDD, como posso programar para ele?** Muito bem! Estamos felizes que o CoMDD esteja lhe interessando. Para você desenvolver nele conheça melhor ele aqui.

*Fonte:* http://143.107.183.158:8008/xwiki/bin/view/Main/Tutorial

## A.2 Texto II: Como programar para o CoMDD

A-Estrutura A estrutura do código deve ser seguida dessa forma: 1-Declarações 2-Includes e Defines 3-Pacotes 4-Adicionando sensores 5-Main 5.1-Inicializações 5.2-Loop 5.3-Funções Nota Atente para as palavras. A linguagem é case sensitive, o que significa que as palavras devem ser escritas exatamente como são mostradas. Caso falte falte um "s", como por exemplo: adicionar define ao invés de adicionar defines o código gerado estará errado. 1-Declarações 1.1 - Plataforma Você pode desenvolver código para 3 plataformas. São elas pioneer, srv e carrinho de golfe. Veja como escrever: plataforma [tipo da plataforma] Ex: plataforma srv Os tipos são pioneer, srv, golfe 1.2 - Nome do robô robo [nome qualquer] Ex: robo XyZ33 2-Includes e Defines O CoMDD trás um conjunto de includes e defines. Para usá-los basta escrever: adicionar [includes ou defines] Ex: adicionar defines 3-Pacotes Há dois pacotes disponíveis. São eles o player e o localizacao. Para usá-los basta escrever: importar pacote [tipo do pacote]; Ex: importar pacote player; 4-Adicionando Sensores Para usar os 3 tipos de sensores (gps, bussola e camera) você deve adicioná-los antes no código, da seguinte forma: criarSensor [tipo de sensor] Ex: criarSensor gps

5-Main A Main é a função principal do código, todo a lógica do código vai dentro desta função. Para isso você precisa escrever: int main() «Inicializações Loop Funções » O que está dentro de «» deve ser substituído de acordo com as explicações dos itens 5.1, 5.2, 5.3. 5.1 - Inicializações Para você usar os sensores criados ou outras funções você deve iniciá-las antes. No caso do sensor: [tipo do sensor].ligar(); Ex: gps.ligar(); Ainda é possível carregar uma lista de coordenadas e inicializar o player da seguinte forma: carregarListaCoordenadas(); inicializarPlayer(); 5.2 - Loop Loop é algo que vai ser executado continuamente. Para usá-lo você deve escrever: while(true) «Funções» Não se esqueça de fechar a chave depois de escrever as funções. 5.3 - Funções São possíveis as seguintes funções, a explicação delas está após a seta (->): gps.ler(); -> O sensor gps lê suas coordenadas bussola.ler(); -> O sensor bússola obtém a sua orientação camera.ler(); -> O sensor câmera obtém a sua imagem carregarListaCoordenadas(); -> O robô carrega uma lista de coordenadas receberCoordenada(); -> O robô recebe uma coordenada que deve seguir processaImagem(); -> Processa a imagem recebida pela câmera. Para usá-la você deve antes obter a imagem da câmera. processaInfo(); -> Processa todas as informações dos sensores. defineRegraSeguir(); -> Define a regra seguir um outro robô. defineRegraNaoBater(); -> Define a regra andar pelo ambiente sem bater. defineRegraSeguirMultiplasCoordenadas(); -> Define a regra percorrer um conjunto de coordenadas dadas. andar(); -> O robô começa a se lo-

comover no ambiente de acordo com a regra definida. Para usar uma das funções basta escrever igual ao que está antes da seta

Fonte: http://143.107.183.158:8008/xwiki/bin/view/Main/aqui

## A.3 Instalação do Eclipse+SVN e importação da DSL

Texto 1B: Instalação do Eclipse+svn e importação da dsl 1. Instalar o Eclipse Modeling Tools versão helios http://www.eclipse.org/downloads/packages/eclipse-modeling-tools-includes-incubating-components/heliossr2 2. A versão helios precisa do java 1.5 (pode-se colar o jre 1.5 direto na pasta do eclipse, assim não precisa alterar o path) 3. Instalar os plugins: xpand e xtext 4. Instalar o svn 5. Configurar o svn no eclipse com o projeto http://code.google.com/p/comdd-eclipse-svn/source/checkout

APÊNDICE

A

# Detalhes de implementação

aqui vem todos os programas usados, os scripts, o procedimento....