**UNIVERSIDADE FEDERAL FLUMINENSE**

**INSTITUTO DE COMPUTAÇÃO**

**CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**DISCIPLINA: ENGENHARIA DE SOFTWARE II**

**PROFESSOR: LEONARDO GRESTA PAULINO MURTA**



**PROJETO XADREZ ES2**

**Grupo 1: Guilherme Rangel Ferreira**

**João Felipe Nicollaci Pimentel**

**Luis Antônio Vieira Júnior**

**Roberto Sampaio**

**Thayanne Antão Viegas**

**Niterói, Junho de 2014**

**SUMÁRIO**

1**. Introdução 03**

1.1 Objetivo 03

1.2 Tecnologias Utilizadas 03

1.3 Equipe 03

2. **Planejamento 05**

2.1 Cronograma 05

2.2 Marcos 05

2.3 EAP 06

2.4 Atividades 06

2.5 Estimação das atividades 08

2.6 Caminho Crítico 08

2.7 Custo e Orçamento 09

2.8 Plano de Riscos 10

3. **Monitoramento e Controle 12**

3.1 Burnup e Burndown 12

3.2 Análise de Valor Agregado 16

4. **Gerência de Configuração 18**

5. **Testes e Cobertura 19**

5.1 Testes 19

5.2 Cobertura 20

6. **Produto Final 23**

7. **Conclusão 27**

Apêndice A - **Manual de Instalação 28**

**1. Introdução**

**1.1 Objetivo**

O projeto teve como objetivo principal o desenvolvimento de um jogo de xadrez que permitisse o usuário jogar contra uma inteligência artificial, também criada por nós; jogar contra um outro jogador; ou assistir uma partida da inteligência artificial contra ela mesma. Além disso, outro objetivo importante do trabalho foi o uso e aprendizado de diversas técnicas de Engenharia de Software, como Scrum, gerência de configuração, testes, técnicas de monitoramento e controle, etc.

**1.2 Tecnologias e Ferramentas Utilizadas**

Para o nosso projeto, escolhemos usar a linguagem de programação Python, por ser uma linguagem onde vários integrantes do grupo já possuíam experiência, além de possuir inúmeras ferramentas de apoio à linguagem, que serão descritas ao longo deste relatório.

Optamos também por utilizar o Pygame, que consiste em uma biblioteca de código aberto de jogos multiplataforma (independente de sistema operacional) feita para auxiliar a criação de jogos e programas multimídia com a linguagem Python.

Usamos também o Git como controle de versionamento, através do serviço GitHub, que detalharemos mais no capítulo de Gerência de Configuração.

Para executar os testes do projeto, usamos a ferramenta unittest, própria da linguagem python. Para medir a cobertura do código usamos o Coverage.py, uma ferramenta que mede a cobertura em códigos em Python. Essas ferramentas serão apresentadas em mais detalhes no capítulo de Testes.

**1.3 Equipe**

A equipe é composta por 5 integrantes, tendo como gerente o João Felipe Pimentel. A responsabilidade de cada integrante está descrita na tabela abaixo:

|  |  |
| --- | --- |
| **Integrante** | **Responsabilidade** |
| Guilherme Ferreira | Documentação |
| João Felipe Pimentel | Gerente/Desenvolvimento |
| Luis Antônio Vieira | Desenvolvimento |
| Roberto Sampaio | Desenvolvimento/Documentação |
| Thayanne Viegas | Desenvolvimento/Documentação |

**2. Planejamento**

**2.1 Cronograma**

A data de início do projeto foi em 23/02/2014 e a entrega foi definida em 11/06/2014. É importante ressaltar que este projeto tem fins estritamente acadêmicos, logo o não comprimento do prazo acarreta a perda de pontos na disciplina, e é impossível tentar negociar uma extensão do prazo. Abaixo está o cronograma detalhando cada iteração e a duração da mesma.

|  |  |  |
| --- | --- | --- |
| **Iteração** | **Descrição** | **Duração** |
| 1 | Telas do sistema, peças no tabuleiro e planejamento inicial | 23/02 - 08/03 |
| 2 | Jogadores movendo as peças corretamente, alternando entre eles, verificando vitória. Relógio | 09/03 - 22/03 |
| 3 | IA, Prova ES2, Seminário | 23/03 - 05/04 |
| 4 | IA e Melhorias na aplicação | 06/04 - 19/04 |
| 5...8 | Melhorias, IA e Otimização | 20/04 - 11/06 |

O nosso projeto foi bem fiel ao cronograma, já tendo uma versão parcial do jogo funcionando a partir da 3a iteração. Desde então, todas as melhorias feitas no código foi visando o aperfeiçoamento e otimização da inteligência artificial e jogabilidade em geral. Requisitos adicionais, como a integração do jogo à API feita pelo monitor da disciplina, foram incorporadas ao projeto sem grandes impactos sobre o cronograma inicial.

**2.2 Marcos**

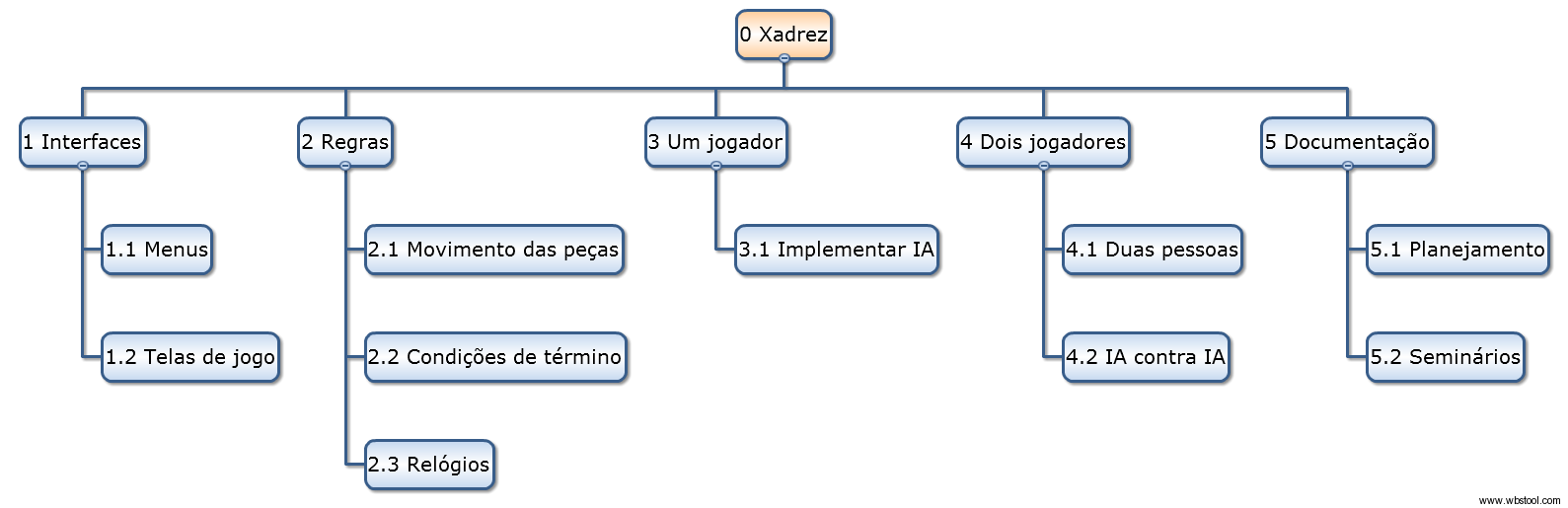
A seguir está uma tabela mostrando os marcos planejados inicialmente:

|  |  |  |
| --- | --- | --- |
| **Marco** | **Descrição** | **Iteração Planejada** |
| 1 | Motor de jogo | 1 |
| 2 | Modo dois jogadores | 2 |
| 3 | Modo um jogador com IA | 4 |
| 4 | Versão Final | 8 |

Com exceção do Marco 3, todos os outros marcos foram atingidos com sucesso. Ao chegarmos na iteração 4, a iteração referente ao Marco 3, já tínhamos uma versão da IA funcionando, porém ela não estava tão otimizada a ponto de ser considerada uma versão final, por isso consideramos que a o Marco 3 não foi atingido com sucesso.

**2.3 EAP (Estrutura Analítica do Projeto)**

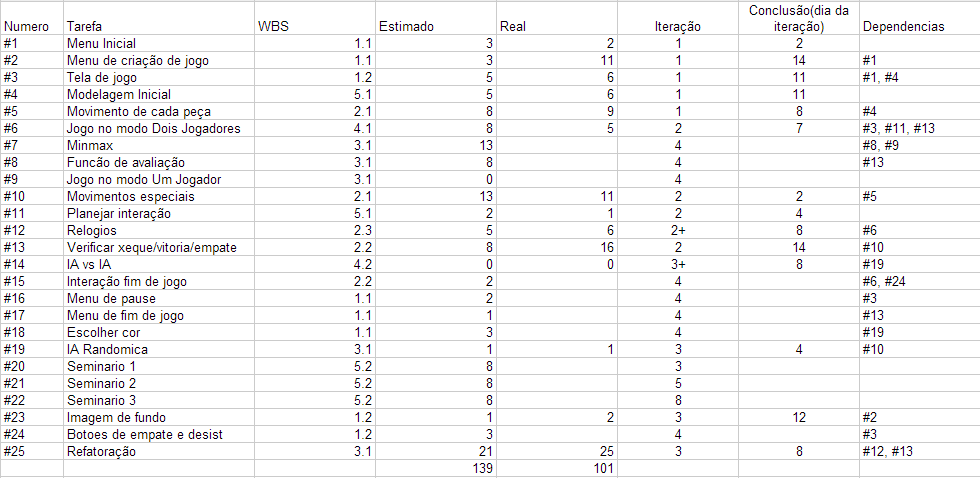
A seguir está o nosso diagrama de EAP, que mostra o panorama geral do trabalho, além de especificar fases e produtos parciais. É importante ressaltar que, a junção de todas as fases e produtos parciais resulta no projeto completo, onde se é possível agrupar qualquer atividade do projeto em uma das fases ou produtos parciais especificados pela EAP.



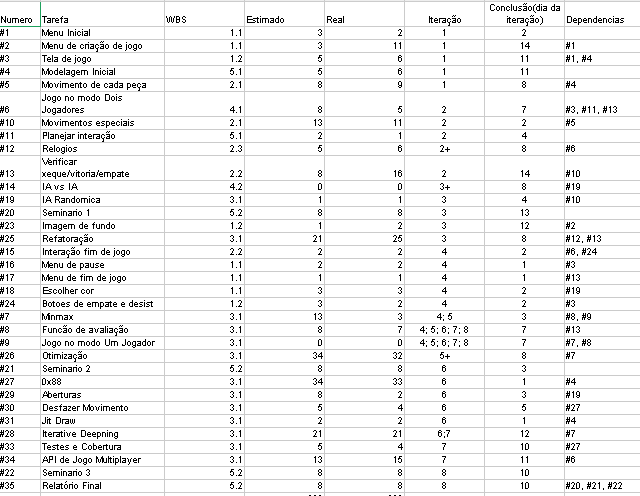
A EAP nos ajudou a estimar com mais segurança o esforço dos pacotes de trabalho, além de nos proporcionar uma visão geral gráfica do projeto.

**2.4 Atividades**

Cada pacote da EAP foi então decomposto em atividades, e cada requisito adicional que apareceu no meio do projeto foi adicionado em seu pacote correspondente. A seguir está uma tabela detalhando cada atividade, a que pacote da EAP ela pertence, o esforço estimado e o esforço real de cada atividade, à qual iteração a atividade pertence, em qual dia a atividade foi finalizada, e as dependências da atividade.



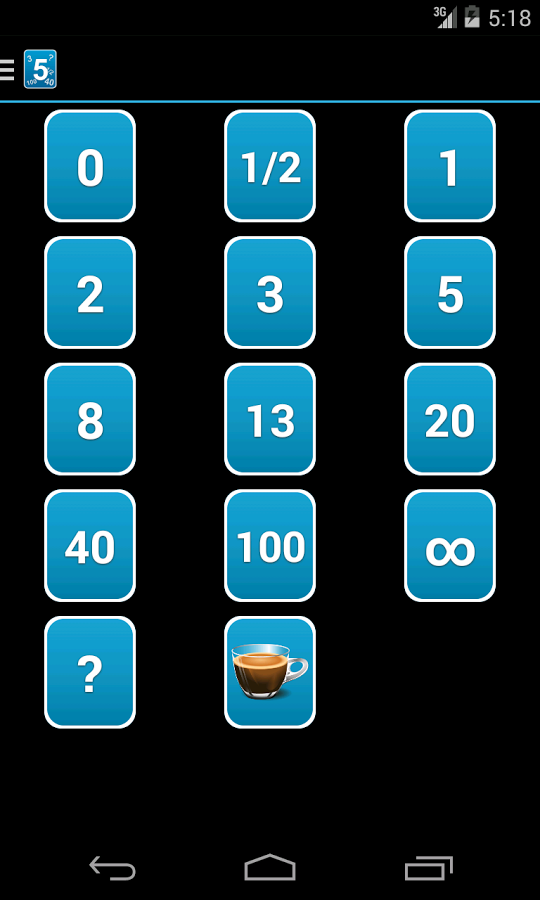
A imagem acima retrata as atividades basicamente até a iteração quatro, ao decorrer do projeto houve a necessidade de inclusão de novas tarefas e consequentemente novas iterações, mais voltadas a otimização, de modo que a tabela de atividade ao final do projeto ficou assim:



**2.5 Estimação das Atividades**

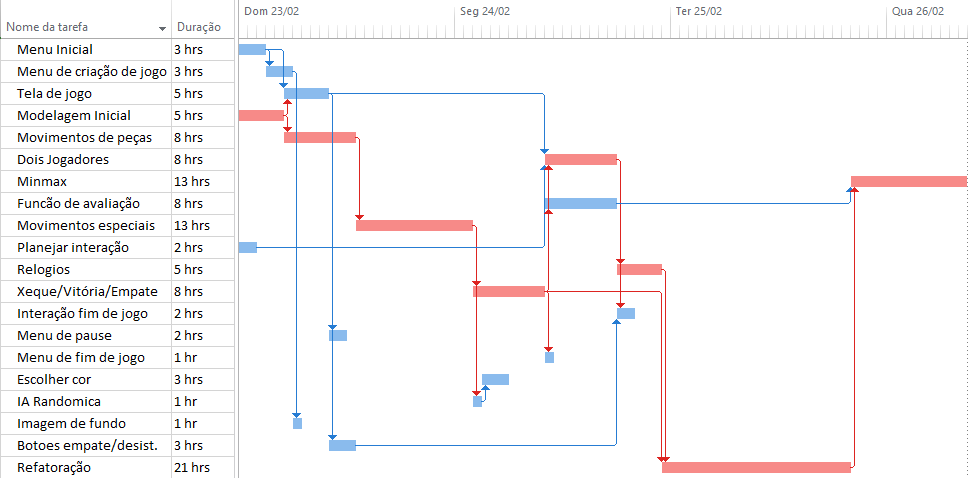
Para estimar o esforço das atividades, nós usamos a técnica de Planning Poker, que consiste em cada integrante mostrar qual é a sua estimativa para uma atividade, e após todos os integrantes da equipe tenham mostrado suas estimativas, chegar em um consenso sobre qual é o esforço para a atividade.

Para realizar o Planning Poker, usamos duas ferramentas mostradas a seguir: o site Hat.jit.su, para quando realizávamos reuniões pela internet, com cada integrante da equipe em sua residência; e o aplicativo para celulares Scrum Poker, para quando a equipe estava junta.

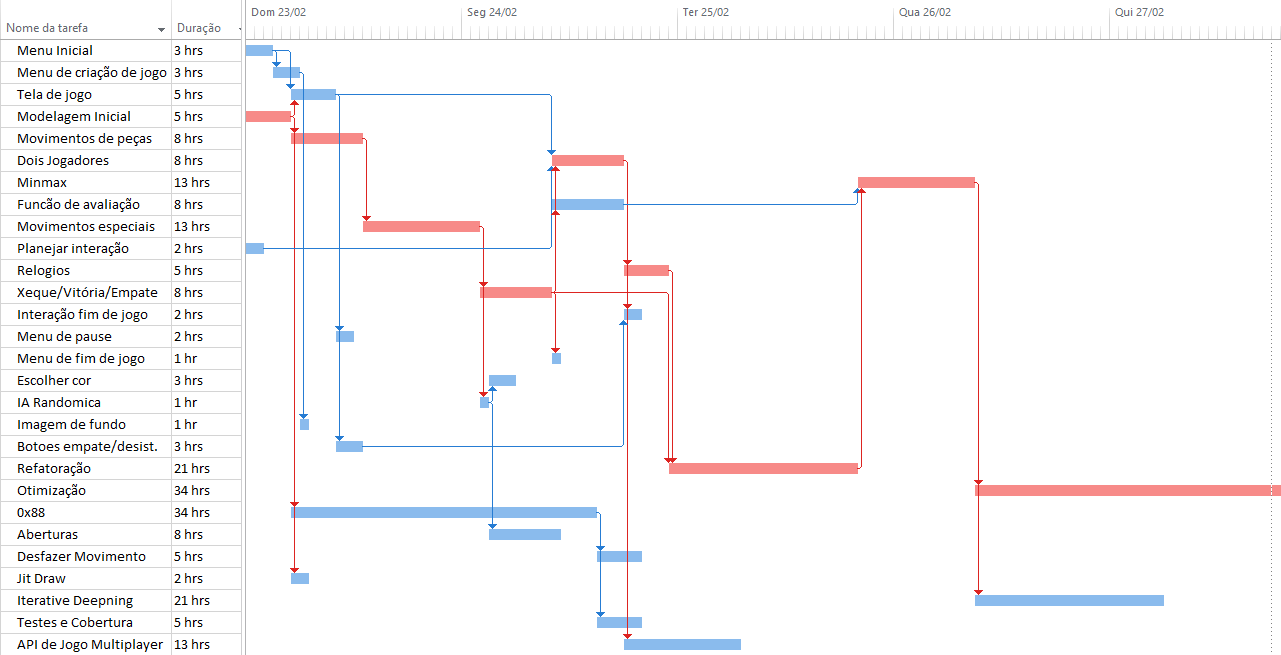


**2.6 Caminho Crítico**

Para achar o caminho crítico, fizemos um gráfico de Gantt para as atividades inicialmente planejadas, para que assim pudéssemos identificar as atividades que, caso fossem atrasadas, impactariam no atraso de todo o projeto.



Após a adição de tarefas de otimização da IA e API de jogo multiplayer, o gráfico de Gantt foi alterado:



**2.7 Custo e Orçamento**

Para definir o custo do projeto, assumimos o valor/hora médio da equipe em R$50.00/h. Inicialmente estimamos um esforço total do projeto em 139 homens-hora, chegando assim a um custo total de recursos humanos em 50\*139 = R$6.950,00. Os custos relacionados a recursos de capital e recursos de consumo foram desconsiderados, já que já possuíamos previamente todo o material necessário, além de não sermos os responsáveis de arcar com os custos de consumo, como energia.

Com a adição de novas tarefas ao longo do período estimado para o projeto, o esforço estimado aumentou para 269 homens-hora, resultando em um custo total de recursos humanos em 50\*269 = R$13.450,00

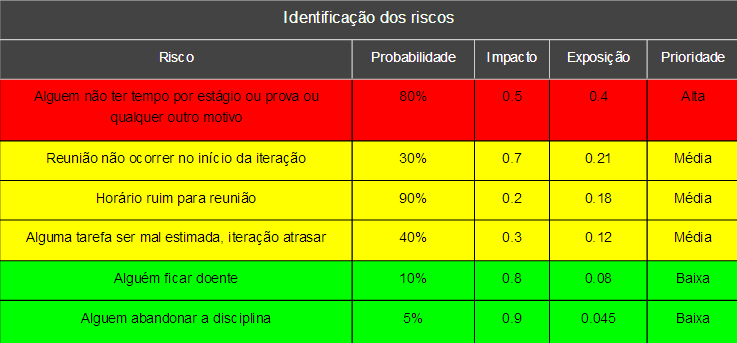
Logo, o custo total ficou em R$13.450,00. Considerando um lucro de 30%, teríamos um orçamento de 13.450,00\*1,3 = R$17.485,00.

Porém, como este projeto possui fins puramente acadêmicos, ele é considerado um projeto estratégico, onde trabalhamos sem verba alguma com o intuito de aprendermos as técnicas ensinadas na disciplina.

**2.8 Plano de Riscos**

Como todo projeto, o nosso também teve riscos. Devido a isso, fizemos um plano de riscos para que pudéssemos, desde o início, identificar os riscos; estabelecer a probabilidade de ocorrência, o impacto e a exposição de cada risco identificado; priorizar e mitigar os riscos; monitorá-los e tomar ações corretivas quando necessário.

A seguir estão duas tabelas, a primeira com a identificação dos riscos, e a segunda com os planos de contenção e contingência.



|  |  |  |
| --- | --- | --- |
|  | **Mitigação** |  |
| **Risco** | **Contenção** | **Contingência** |
| Alguém não ter tempo por culpa de problemas externos | Iteração de 2 semanas, tarefas curtas | Algum vagal pegar mais tarefas |
| Horário ruim para reunião | Definir horário melhor | Resolver os items mais urgentes e remarcar o resto da reunião |
| Alguma tarefa ser mal estimada, iteração atrasar | Quebrar tarefas grandes | Planejar terminar projeto antes da data máxima de entrega |
| Alguém ficar doente | Evitar ficar doente | Se tratar |
| Alguém abandonar a disciplina | Fazer o coleguinha desistir de desistir da matéria | Avisar o resto do grupo com antecedência e dividir as tarefas pertencentes à ele |

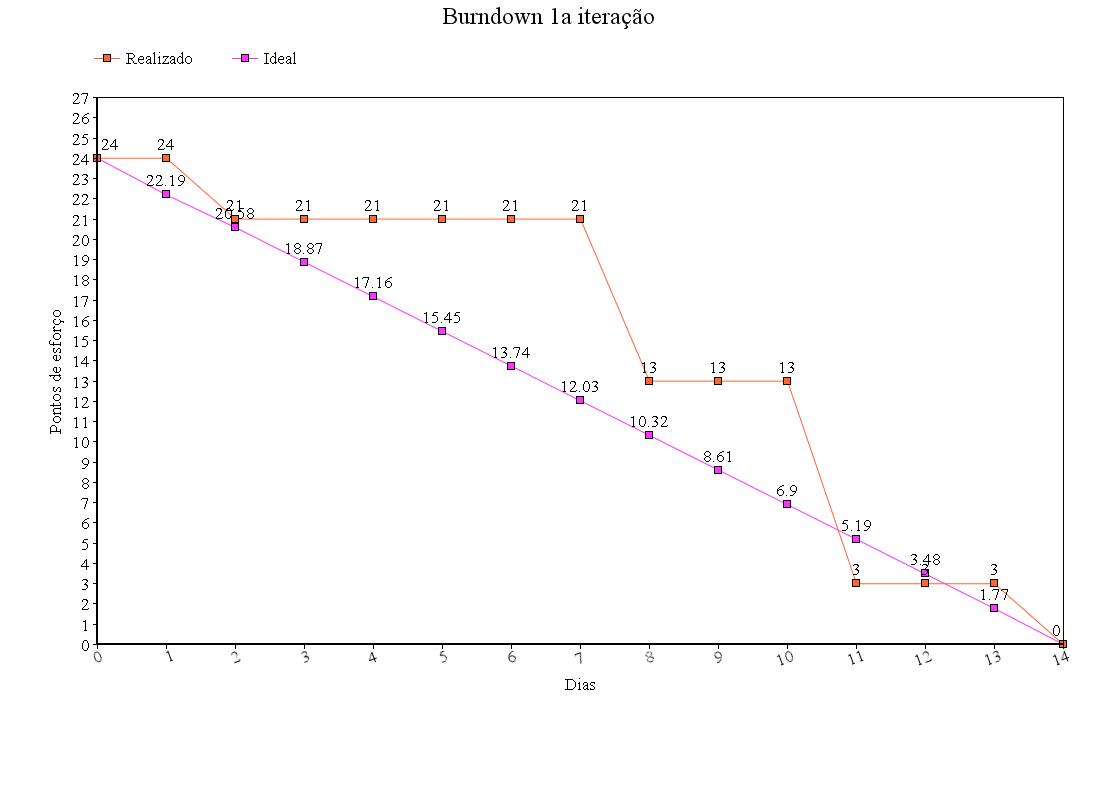
**3. Monitoramento e Controle**

**3.1 Burndown e Burnup**

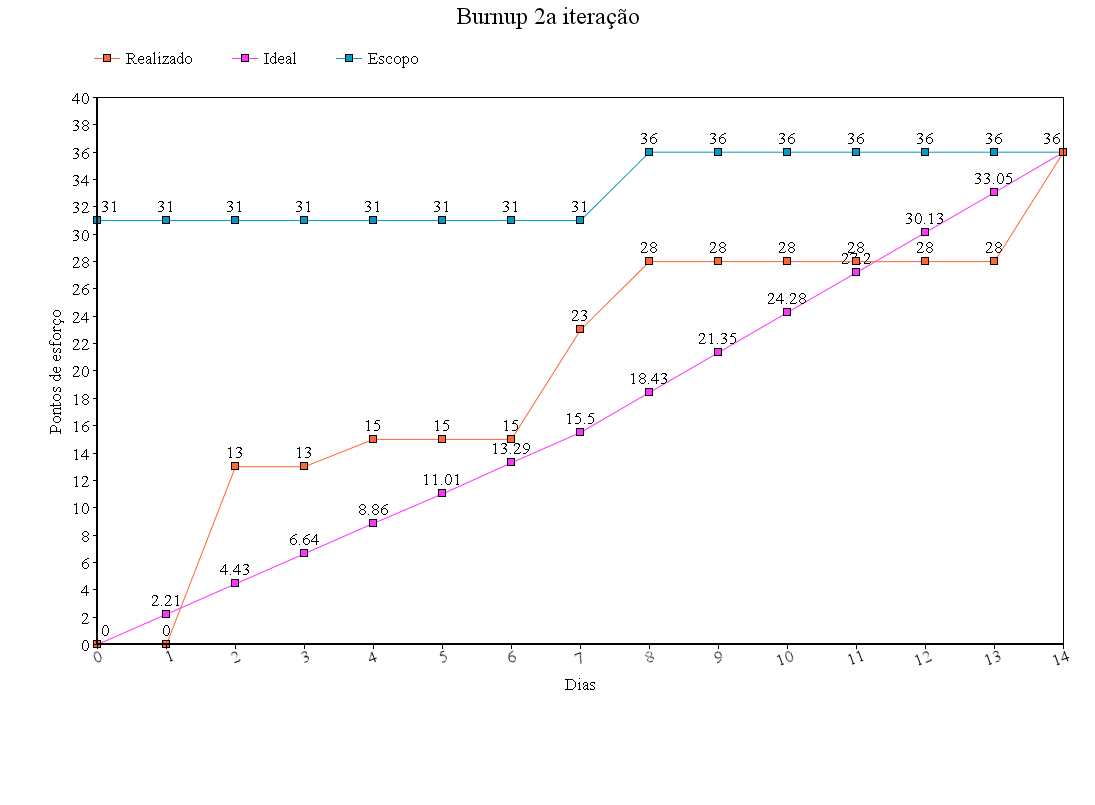
Como forma de monitoramento, usamos os gráficos de Burndown e Burnup para mostrar uma representação gráfica entre o esforço estimado restante em relação ao tempo. Optamos por usar o Burnup ao invés do Burndown na maioria das iterações pois o Burnup representa melhor o fato de atividades novas terem sido criadas no meio da iteração.

A seguir estão os gráficos de cada iteração, com os seus respectivos esforços estimados e reais.

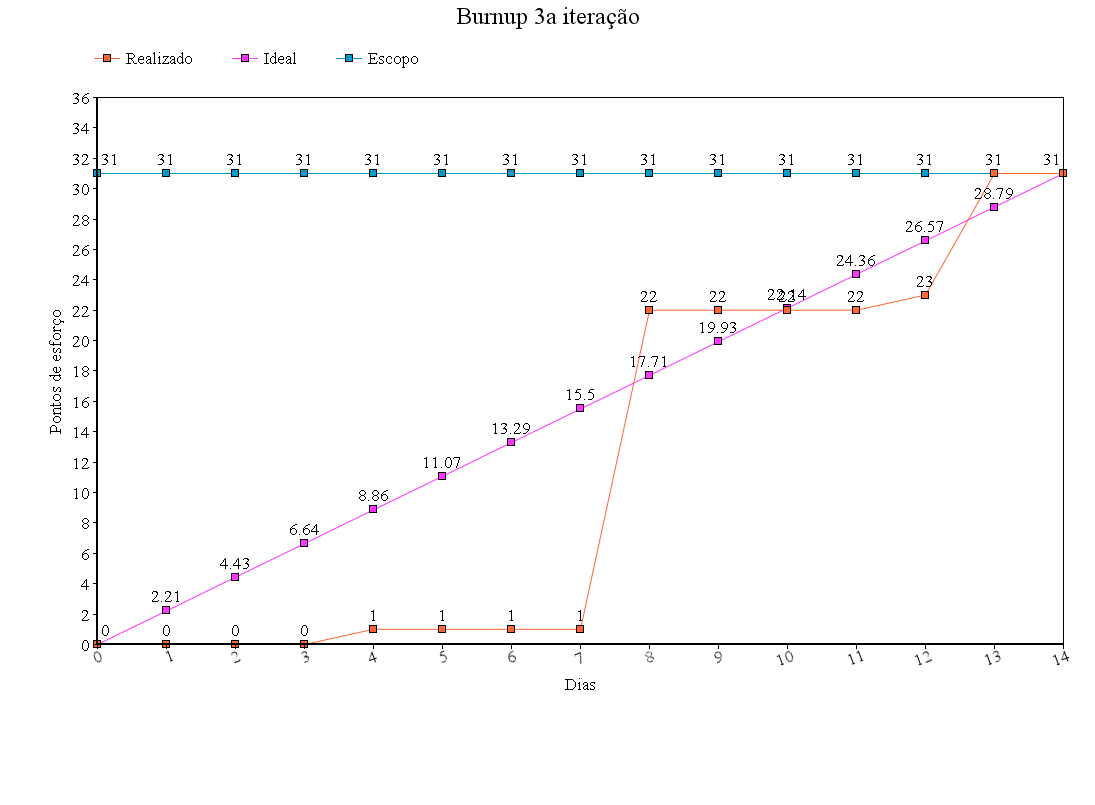
Iteração 1 (23/02 - 08/03) - Esforço estimado: 24hh; Esforço real: 34hh



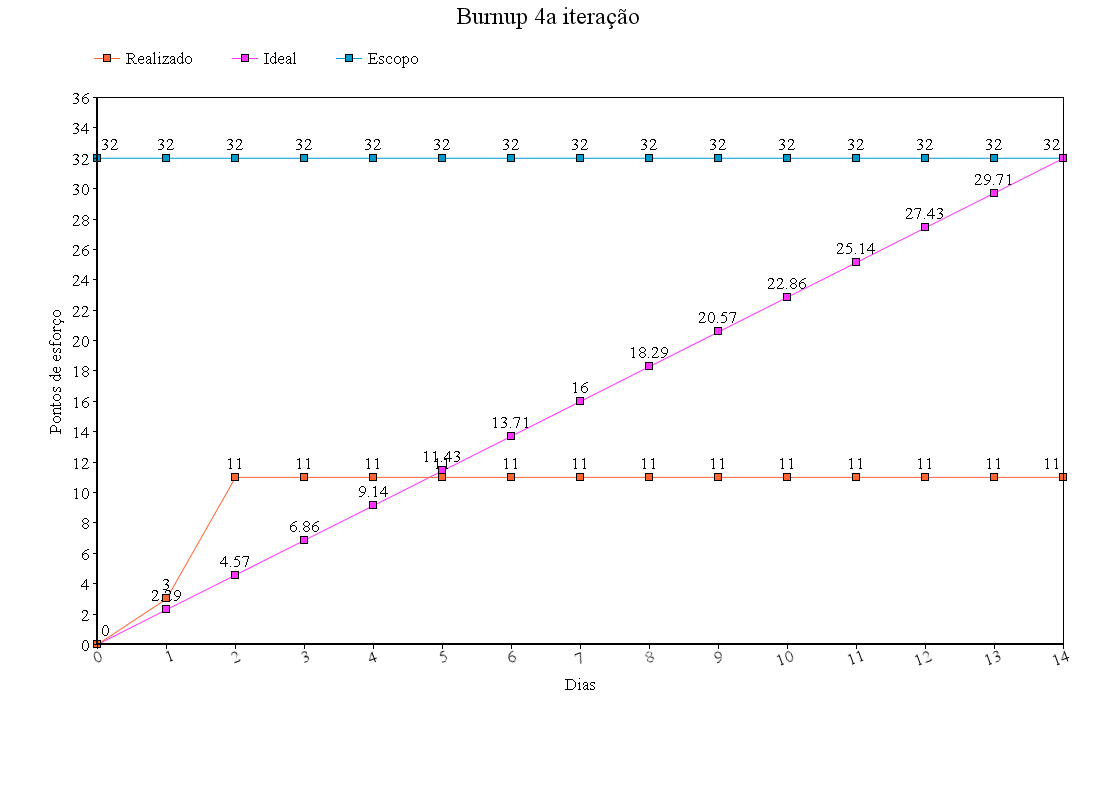
Iteração 2 (09/03 - 22/03) - Esforço estimado: 31+5 = 36hh; Esforço real: 39hh



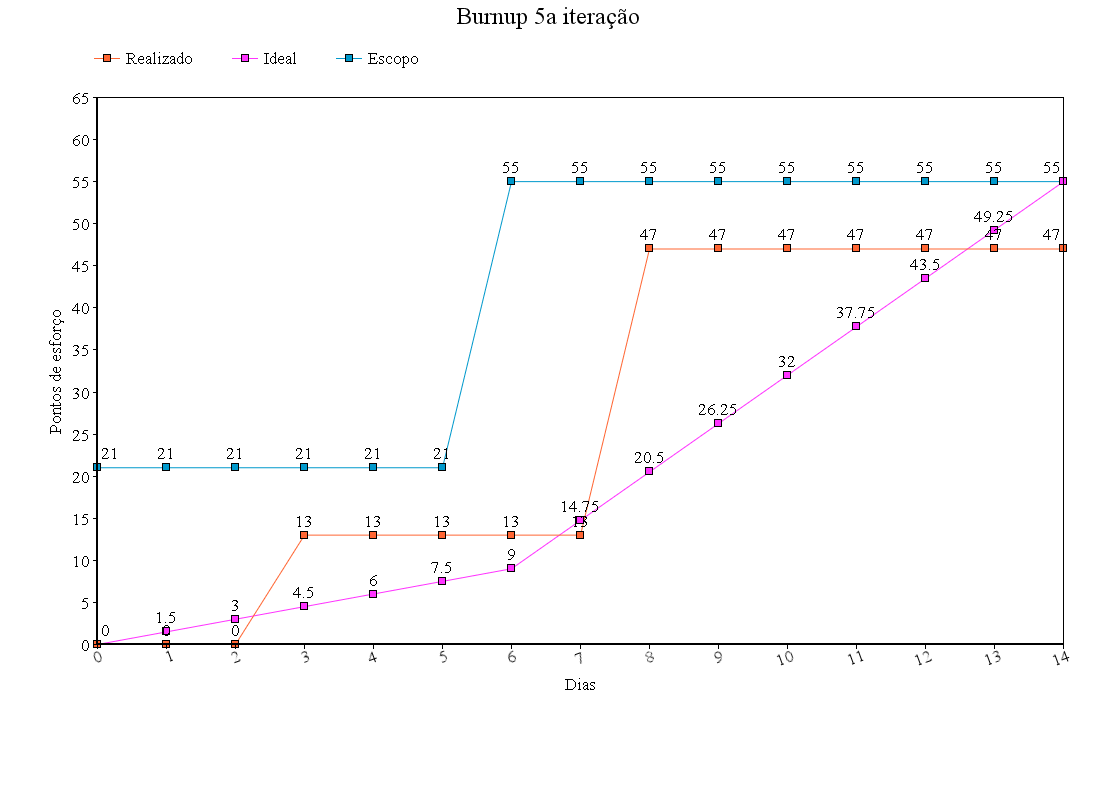
Iteração 3 (23/03 - 05/04) - Esforço estimado: 31hh; Esforço real: 36hh



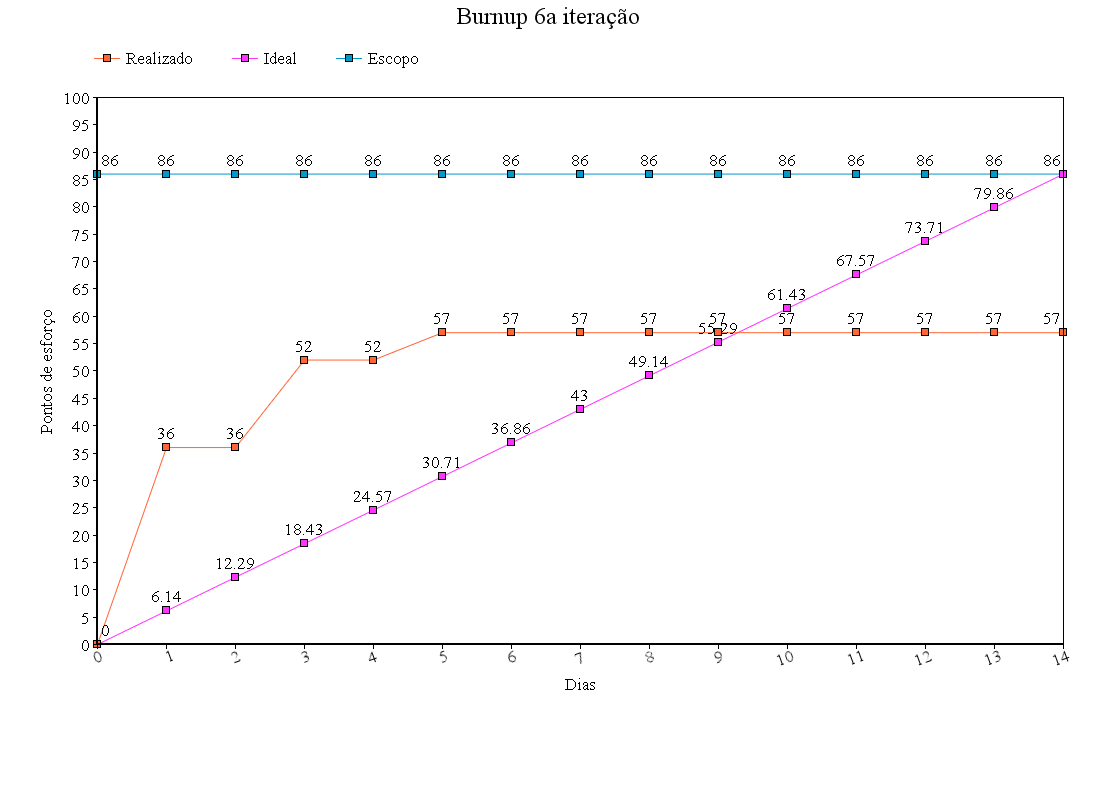
Iteração 4 (06/04 - 19/04) - Esforço estimado: 32hh; Esforço real: 10hh



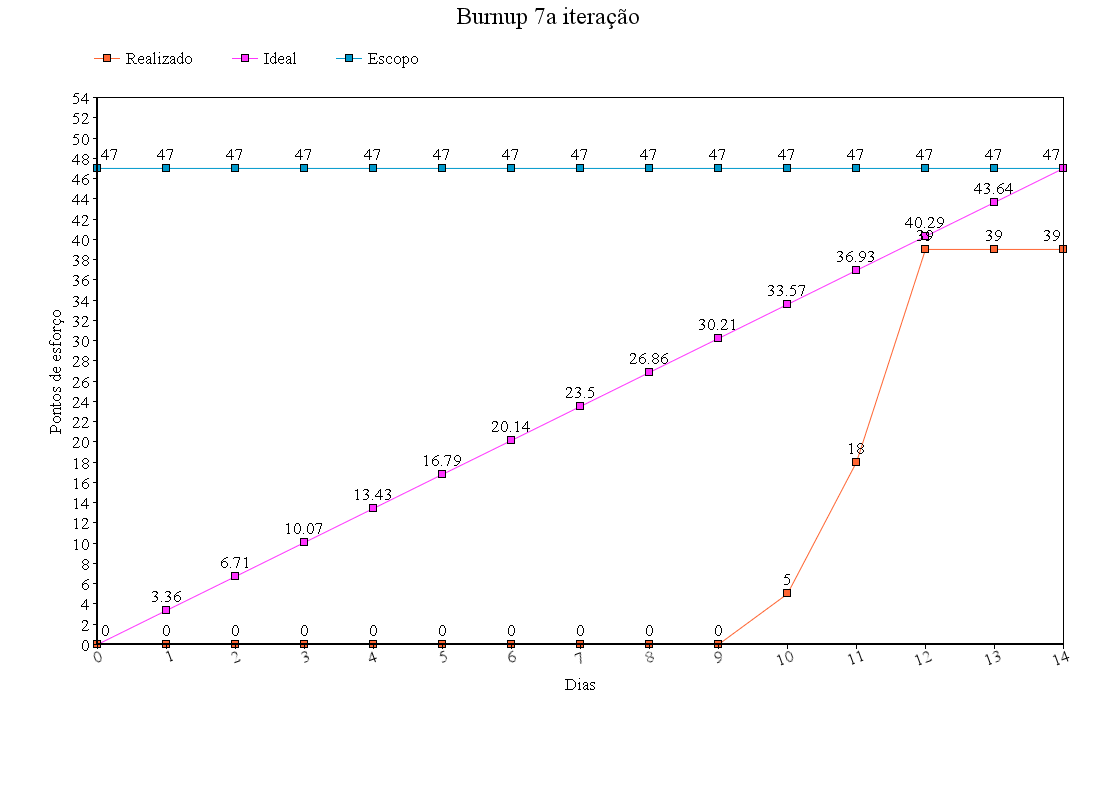
Iteração 5 (20/04 - 03/05) - Esforço estimado: 21 + 34 = 55hh; Esforço real: 35hh



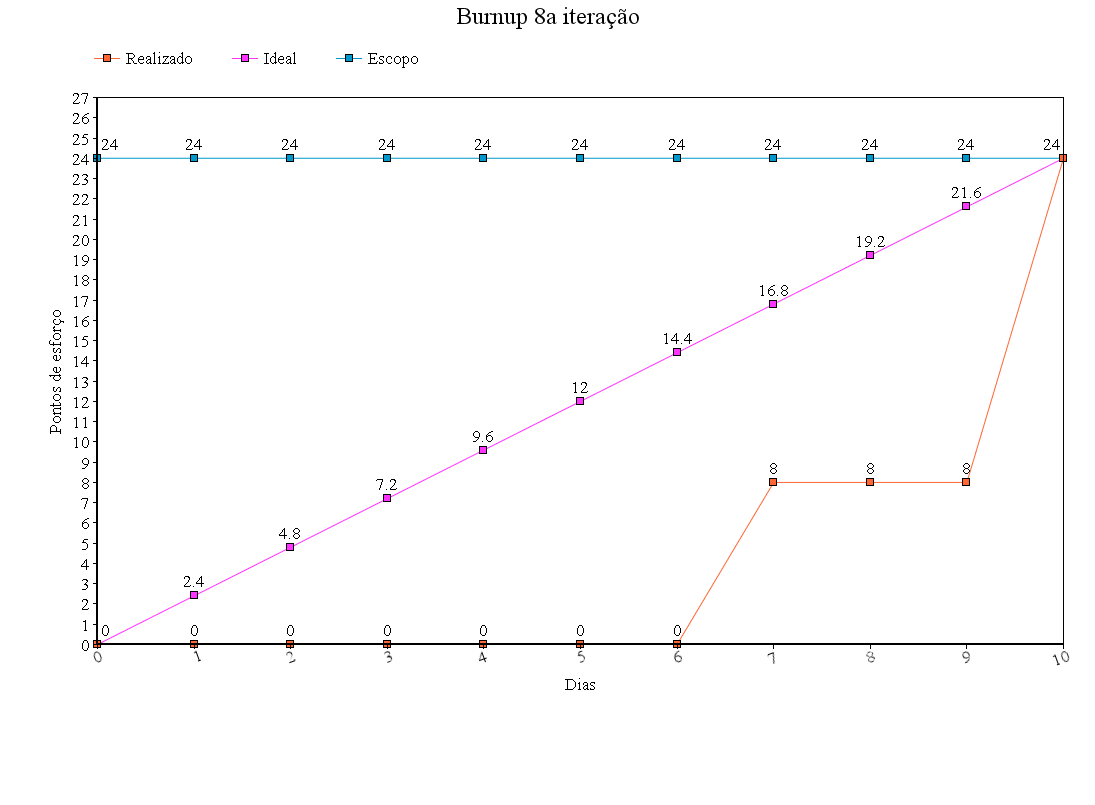
Iteração 6 (04/05 - 17/05) - Esforço estimado: 86hh; Esforço real: 49hh



Iteração 7 (18/05 - 31/05) - Esforço estimado: 47hh; Esforço real: 40hh



Iteração 8 (01/06 - 11/06) - Esforço estimado: 24hh; Esforço real: 23hh



**3.2 Análise de Valor Agregado**

Para podermos analisar a evolução do projeto ao longo do tempo em relação ao custo, usamos a técnica de Análise de Valor Agregado. Abaixo estão os cálculos referentes a cada iteração, e a análise dos resultados dos mesmos, e o gráfico feito no final do trabalho mostrando a evolução ao longo de todas as iterações.

Legenda: PV = Valor Planejado; EV = Valor Agregado; AC = Custo Real;

SPI = Indicador de Desempenho do Cronograma; CPI = Indicador de Desempenho do Custo

Iteração 1:

PV: 24\*50 = R$ 1200;

EV: 24\*50 = R$ 1200

AC: 34\*50 = R$ 1700

SPI: EV/PV = 1

CPI: EV/AC = 0.71

Iteração 2:

PV: 55\*50 = R$ 2750

EV: 60\*50 = R$ 3000

AC: 73\*50 = R$ 3650

SPI: EV/PV = 1.09

CPI: EV/AC = 0.82

Iteração 3:

PV: 86\*50 = R$ 4300

EV: 91\*50 = R$ 4550

AC: 109\*50 = R$ 5450

SPI: EV/PV = 1.06

CPI: EV/AC = 0.83

Iteração 4:

PV: 118\*50 = R$ 5900

EV: 102\*50 = R$ 5100

AC: 119\*50 = R$ 5950

SPI: EV/PV = 0.86

CPI: EV/AC = 0.86

Iteração 5:

PV: 157\*50 = R$ 7850

EV: 149\*50 = R$ 7450

AC: 154\*50 = R$ 7700

SPI: EV/PV = 0.95

CPI: EV/AC = 0.97

Iteração 6:

PV: 235\*50 = R$ 11750

EV: 206\*50 = R$ 10300

AC: 203\*50 = R$ 10150

SPI: EV/PV = 0.88

CPI: EV/AC = 1.01

Iteração 7

PV: 253\*50 = R$ 12650

EV: 245\*50 = R$ 12250

AC: 243\*50 = R$ 12150

SPI: EV/PV = 0.97

CPI: EV/AC = 1.01

Iteração 8

PV: 269\*50 = R$ 13450

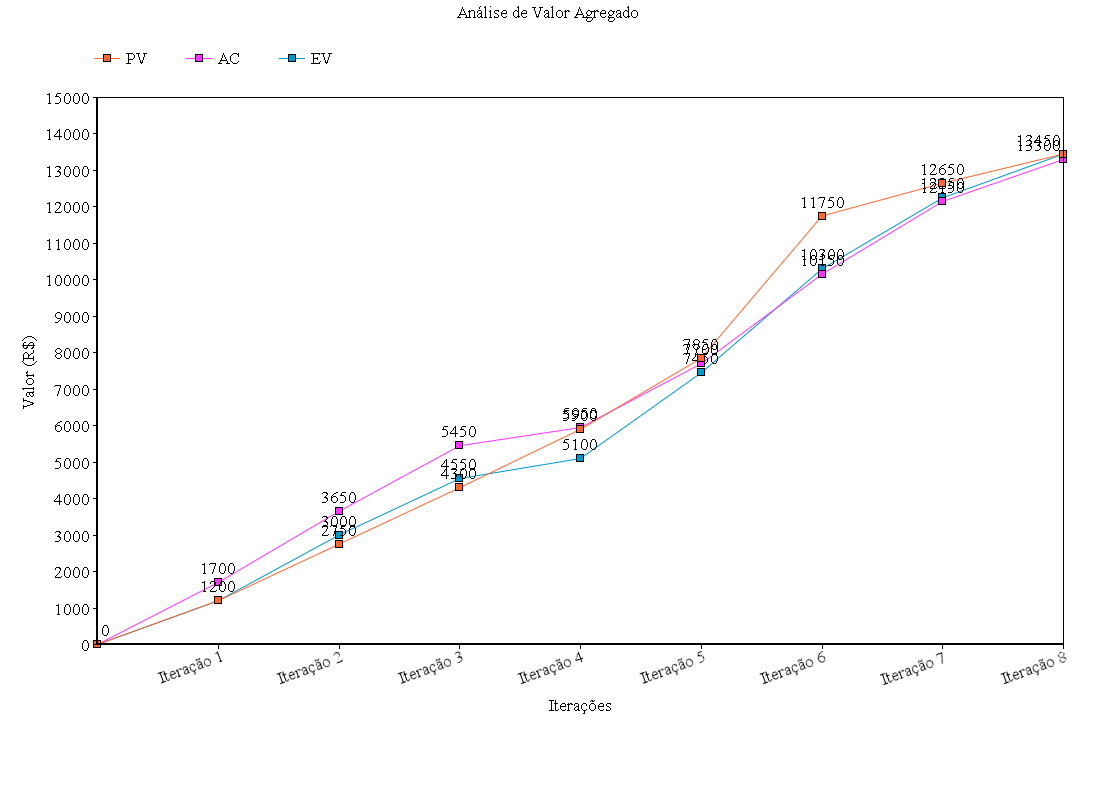
EV: 269\*50 = R$ 13450

AC: 266\*50 = R$ 13300

SPI: EV/PV = 1

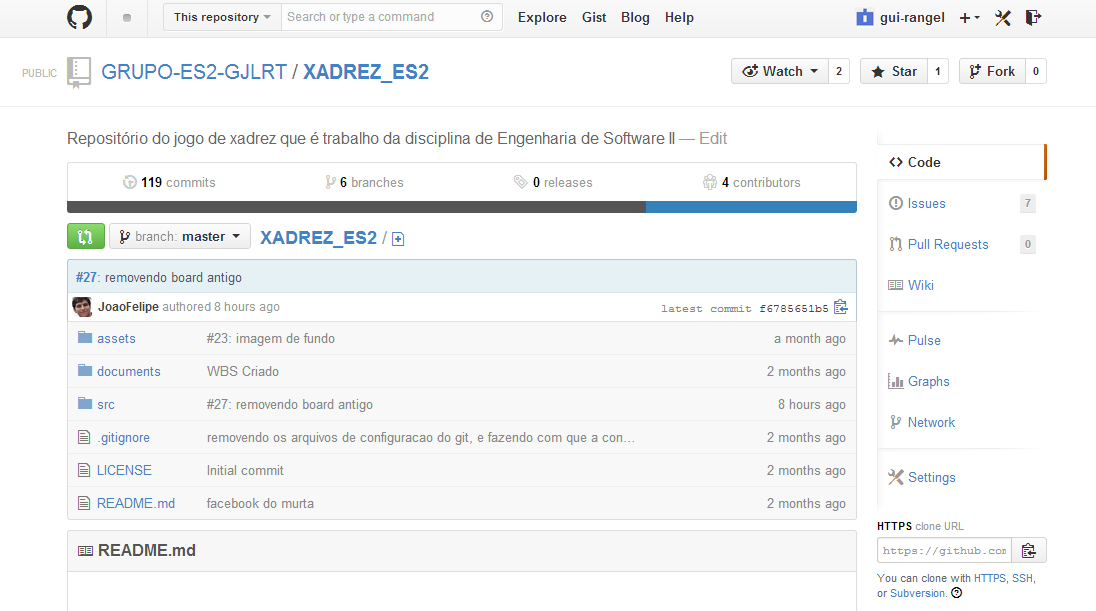
CPI: EV/AC = 1.01

Através da análise do SPI e CPI, que são iguais a 1 e 1.01 na última iteração planejada, podemos observar que o projeto foi concluído dentro do prazo e teve um custo levemente menor do que o planejado.



**4 Gerência de Configuração**

Como controle de versionamento, escolhemos usar o Git, junto com o serviço GitHub, que também atua como repositório do projeto. Escolhemos ele devido à familiaridade que já tinhamos com a ferramenta, e pelas diversas funcionalidades que possui, como gráficos de frequência de commits, criação de marcos e issues, visualização da comparação de dois commits, etc.

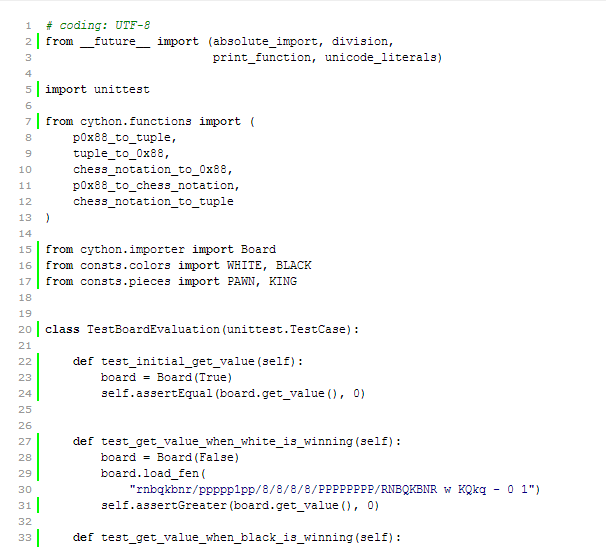


**5 Testes e Cobertura**

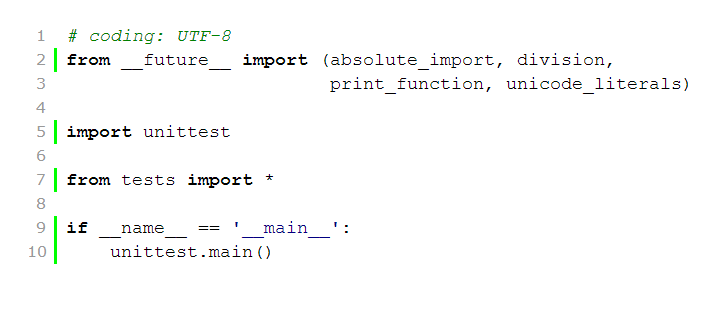
**5.1 Testes**

Para rodar os testes nós usamos o unittest, popularmente chamado de PyUnit, que é um framework nativo da linguagem python. Optamos por usar ele por ser uma ferramenta já própria da linguagem python, onde nenhuma instalação adicional foi necessária, e por ser de fácil uso. A seguir estão exemplos de como usamos os testes e a ferramenta em nsso projeto.

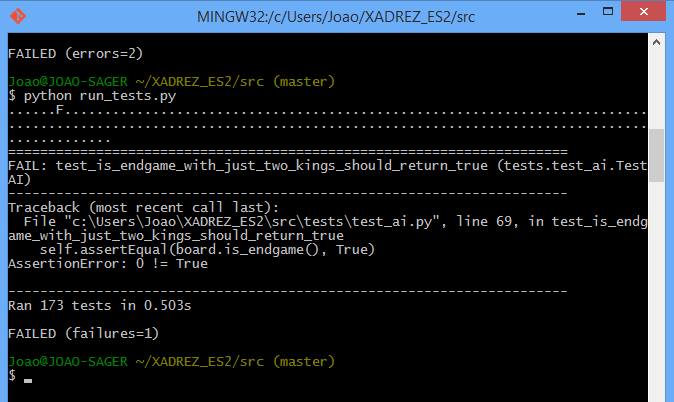
Exemplo de teste:



Arquivo run\_tests, que contém todos os testes e faz a chamada do unittest:



Arquivo run-testes.py sendo rodado pela linha de comando, onde quando há um teste que falha, o unittest indica qual teste que falhou, aonde e porque:

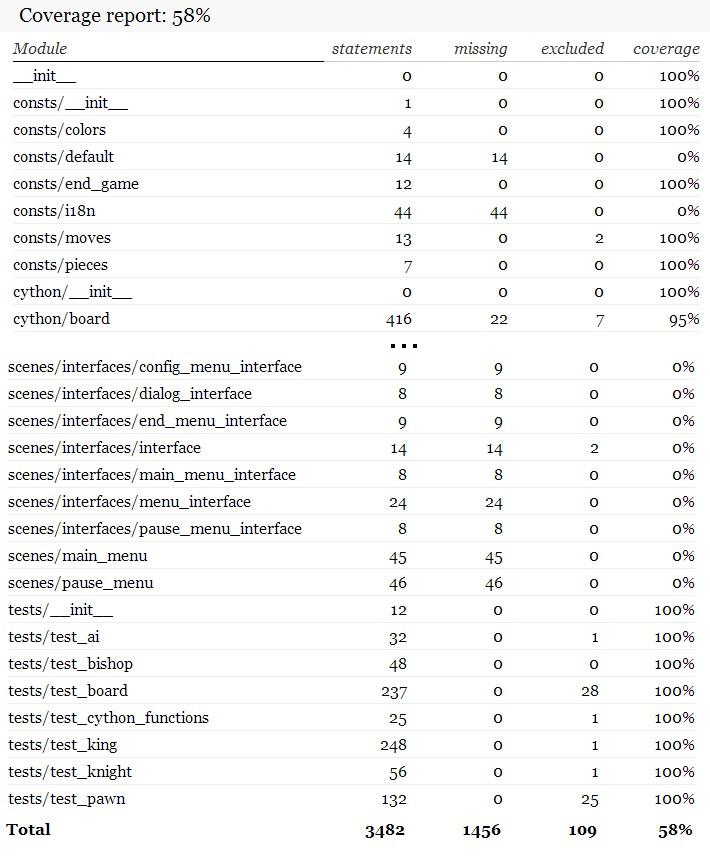


**5.2 Cobertura**

Para testar a cobertura dos testes em relação ao código, nós optamos por usar a ferramenta Coverage.py, que é uma ferramenta que monitora um programa em linguagem python analisando que partes do código foram executadas, e que partes do código poderiam ter sido executadas mas não foram.

Além de ser de fácil uso e efficiente, o Coverage .py também possui a opção de retornar os resultados detalhados em arquivos no formato HTML, para fácil visualização e entendimento. Dentro de cada arquivo de teste, ele mostra não só a porcentagem de cobertura do teste, mas também identifica, na cor verde, as linhas que foram executadas, e na cor vermelha as linhas que poderiam ter sido executadas mas não foram. Abaixo seguem exemplos do uso da ferramenta no nosso projeto.

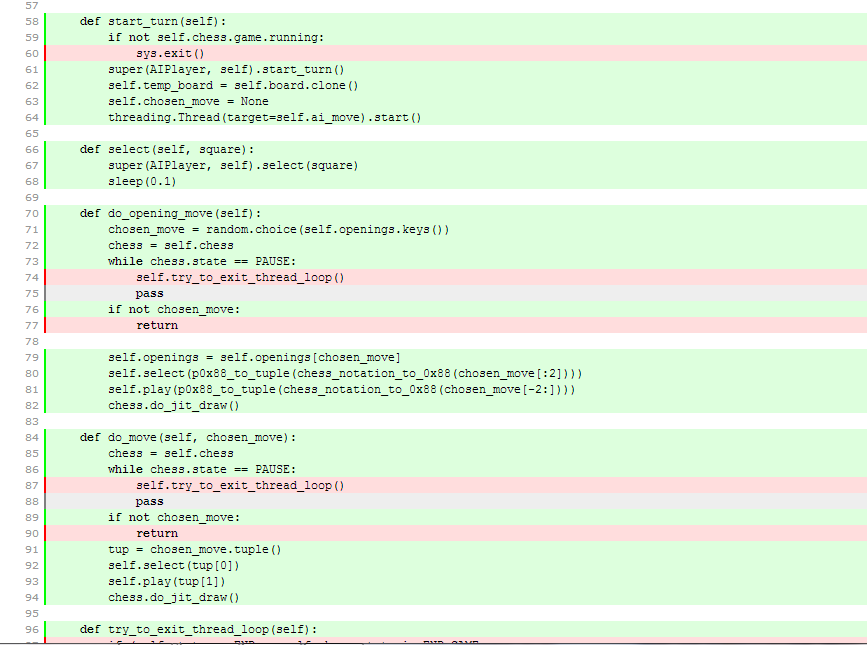
Relatório geral, com todos os testes e suas devidas coberturas:



Exemplo de cobertura no teste game\_elements/ai\_player:

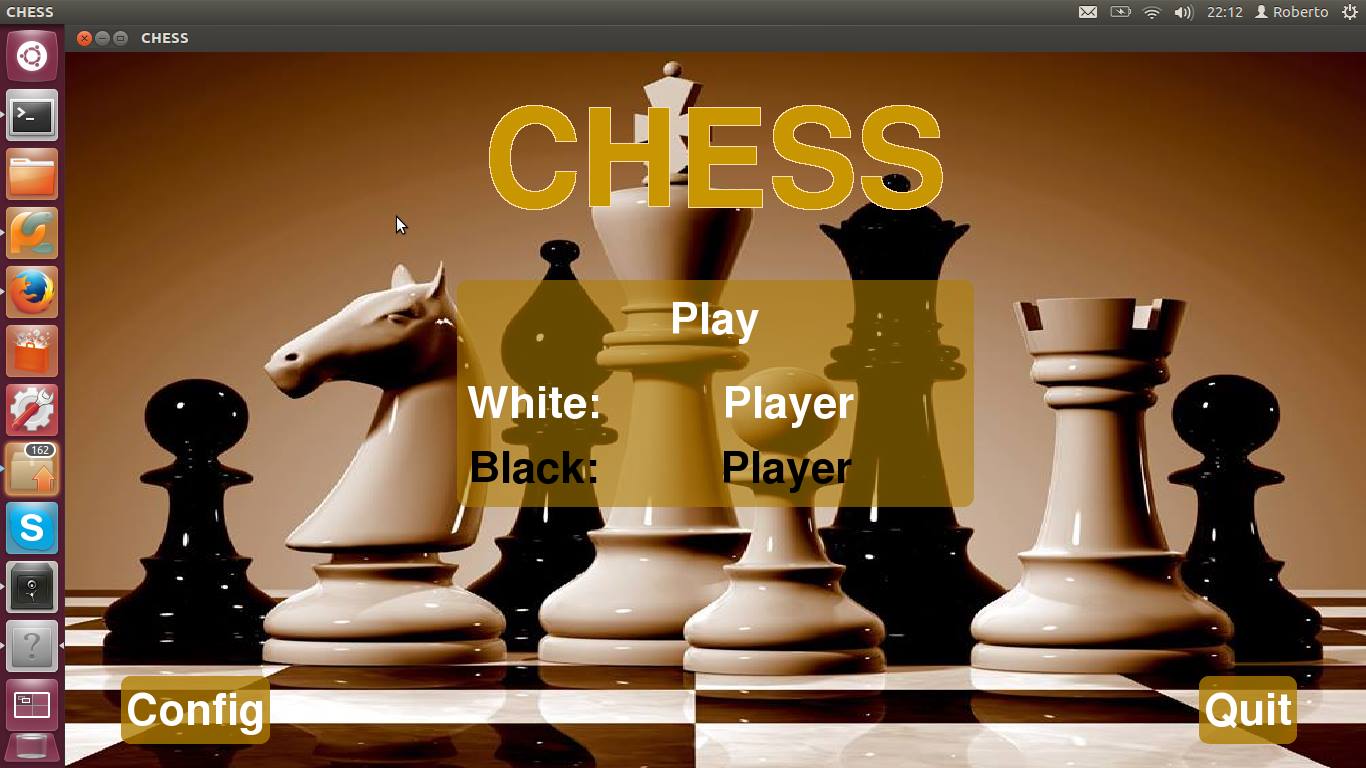


Continuação do exemplo, onde em verde estão as linhas executadas e em vermelho as que não foram executadas:



**6 Produto Final**

Esta abaixo é a ilustração da interface inicial que deve ser vista ao abrir o jogo.



O centro da interface inicial é composto por 3 botões: o botão **Play**, que inicia o jogo, e os botões de **Player**, que permitem que o usuário escolha se jogará contra outro oponente humano ou contra uma instância de inteligência artificial, com 6 possibilidades de nível diferentes. É também possível configurar para que duas inteligências artificiais se enfrentem. O botão **Config**, localizado no canto inferior esquerdo leva à tela de configuração do jogo. O botão **Quit** encerra o jogo.



A tela de configurações oferece ao usuário algumas opções de configuração das partidas, entre elas:

**Timer**: Minutes per Game, Moves per Minutes ou Fischer Time.

**Fifty Move Rule**: Auto, Button ou Disable

**Jit Draw**: Enable ou Disable

**AI Time Out**: 1s, 2s, 3s, 5s, 8s, 13s, 21s ou 34s.

Os modos de tempo possuem também suas sub-opções, como o tempo de jogo no caso do modo de tempo ser “Minutes per Game”.

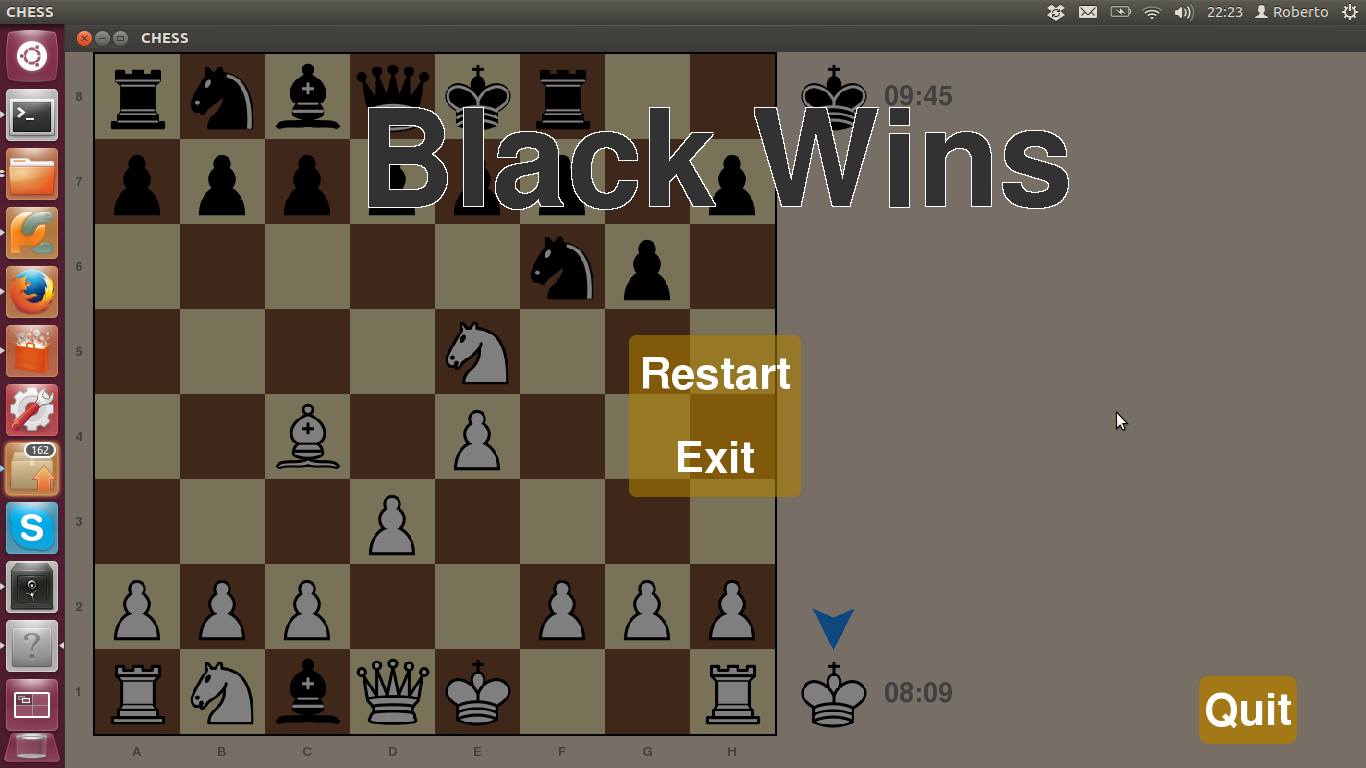
Ao pressionar o botão **Play** o jogo iniciará e durante o jogo, o jogador sempre terá a opção de propor um empate ou desistir, através dos botões **Draw** e **Resign**, respectivamente.



No caso de um empate recusado, a tela mostrada é a seguinte:



Abaixo vemos a tela exibida caso haja uma vitória por check-mate, tempo esgotado ou desistência. No caso abaixo em particular, foi uma desistência.



Por fim, caso ocorra um empate, a tela exibida é a seguinte:



**7 Conclusão**

Ao longo do projeto, usamos várias das técnicas aprendidas durante a disciplina Engenharia de Software II, não apenas porque fomos instruídos a usá-las, mas porque vimos necessidade de utilizá-las. Percebemos que seria difícil planejarmos um projeto dessa magnitude sem técnicas próprias para isso, como por exemplo o planning pokerpara estimação das atividades e o gráfico de Gantt para o caminho crítico. Além disso, as técnicas de monitoramento e controle, como o cálculo do valor agregado e os gráficos de burndown e burnup, nos permitiram acompanhar a evolução gradual do projeto, nos fornecendo informações úteis como, por exemplo, se estávamos dentro do prazo ou dentro do orçamento.

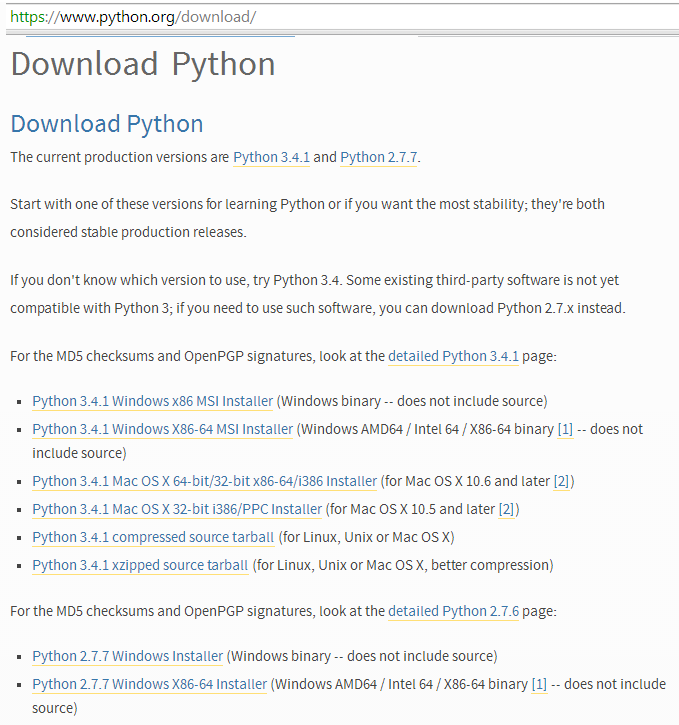
Apesar de ter sido desenvolvido com fins acadêmicos, este projeto nos permitiu uma maior familiaridade com as técnicas e desafios reais encontrados no mercado de trabalho, nos fornecendo um maior preparo para o desenvolvimento de projetos, sejam eles de âmbito acadêmico ou mesmo no mercado de trabalho.

**Apêndice A - Manual de Instalação**

**1 - Instalação do Python**

1.1 - Dirija-se ao website <https://www.python.org/download/>

1.2 - Dentre as opções apresentadas, escolha a mais adequada ao seu computador



1.3 - Baixe o arquivo escolhido

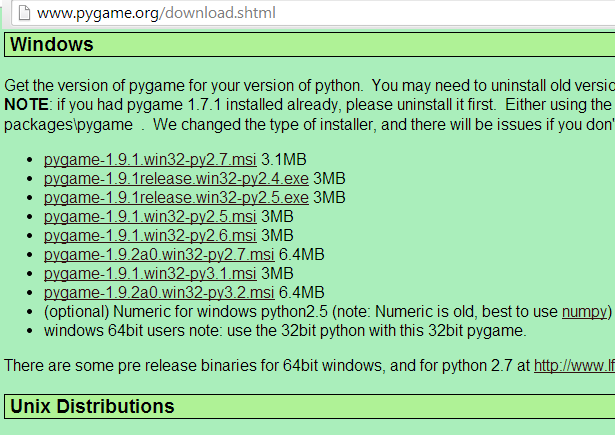
1.4 - Abra o arquivo baixado

1.5 - Siga com a instalação do arquivo até o final

**2 - Instalação do Pygame**

2.1 - Dirija-se ao site <http://www.pygame.org/download.shtml>

2.2 - Dentre as opções apresentadas, escolha a mais adequada ao seu computador



2.3 - Baixe o arquivo escolhido

2.4 - Abra o arquivo baixado

2.5 - Siga com a instalação do arquivo até o final

**3 - Instalação do Cython (Opcional)**

A instalação do Cython é opcional, sua vantagem é deixar a inteligência artificial mais eficiente.

3.1 - Dirija-se ao site http://cython.org/#download

3.2 - Baixe o arquivo

3.3 - Abra o arquivo baixado

3.4 - Extraia para a pasta desejada

3.5 - Abra o arquivo “INSTALL”

3.6 - Siga as instruções a partir de lá

3.7 - Abra a pasta do Cython no projeto

3.8 - Execute o comando:

python setup.py build\_ext --inplace

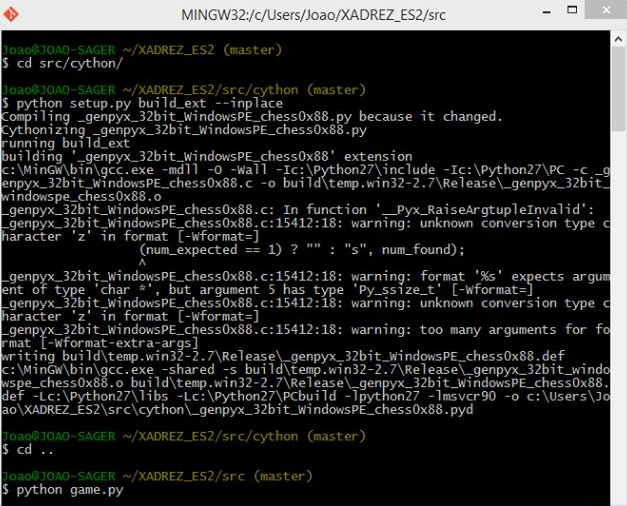
**4 - Abrindo o jogo pelo terminal**

4.1 - Abra o terminal

4.2 - Dirija-se a pasta “src”

4.3 - Execute o comando:

python game.py



**5 - Executando os testes**

4.1 - Abra o terminal

4.2 - Dirija-se a pasta “src”

4.3 - Execute o comando:

python run\_tests.py