DANIELA BRAZÃO MAKSOUD
FELIPE VIEIRA CÔRTES
TÁSSIO ALBUQUERQUE BORGES DE MIRANDA

ESPECIFICAÇÃO DE REQUISITOS

Professor: Flavio Bevilacqua
Disciplina: Programação Modular
Código da Disciplina: INF1301
Período: 2015.2

RIO DE JANEIRO
2015

COMPONENTES DO GRUPO

Nome: Daniela Brazão Maksoud

Matrícula: 1321873

Nome: Felipe Vieira Côrtes

Matrícula: 1321881

Nome: Tássio Albuquerque Borges de Miranda

Matrícula: 1321931

RESUMO

Este trabalho apresenta as especificações de requisitos, restrições e hipóteses referentes ao terceiro trabalho da disciplina INF1301 de Programação Modular, lecionada pelo Prof. Flavio Bevilacqua, do período 2015-2.

Palavras-chave: Especificação requisitos restrições hipóteses.

SUMÁRIO

1 REQUISITOS FUNCIONAIS..... 5
1.1 JOGO..... 5
1.2 PRIMEIRO JOGADOR..... 6
1.3 MOVIMENTAÇÃO DAS PEÇAS..... 7
1.4 RETIRADA DAS PEÇAS..... 9
1.5 DOBRANDO E REDOBRANDO..... 9
1.6 OBJETIVO..... 10
1.7 VENCEDOR..... 10
1.8 VENCENDO POR GAMÃO OU BACKGAMMON..... 10
2 REQUISITOS NÃO FUNCIONAIS..... 12
2.1 ROBUSTEZ..... 12
2.2 CORRETUDE..... 12
2.3 REUSO..... 12
2.4 MANUTENIBILIDADE..... 13
3 REQUISITOS INVERSOS..... 14
3.1 GERAL..... 14
4 RESTRIÇÕES..... 15
4.1 GERAL..... 15
4.2 DADO..... 15
4.3 DADOPONTOS..... 15
4.4 PECA..... 15
4.5 TABULEIRO..... 16
4.6 JOGO..... 16
4.7 PEÇAS CAPTURADAS..... 16
4.8 PEÇAS FINALIZADAS..... 16
5 HIPÓTESES..... 17
5.1 GERAL..... 17

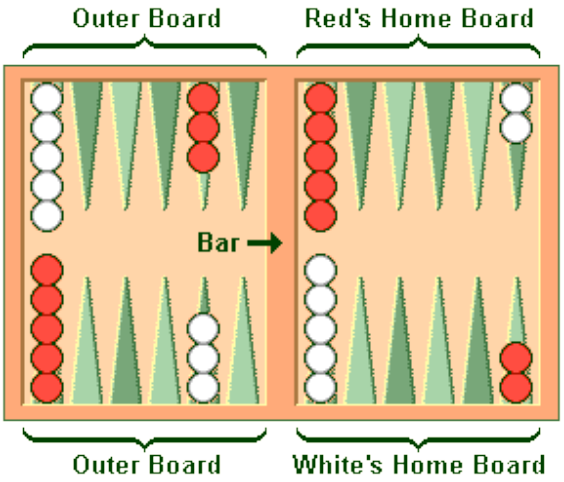
1 REQUISITOS FUNCIONAIS

Segundo Staa (2000, p. 339): “**Requisitos Funcionais** especificam as funções que o artefato deve ser capaz de executar, ou, dito de outra forma, especificam o serviço a ser prestado pelo artefato.”

1.1 JOGO

1. É praticado entre duas pessoas, num tabuleiro de 24 casas triangulares (também chamadas de pontos) de cores intercaladas agrupadas em 4 quadrantes de 6 triângulos cada.
2. Os quadrantes são chamados de tabuleiro interno (*home board*, em inglês) e tabuleiro externo (*outer board*, em inglês) de cada jogador, conforme a figura 1.

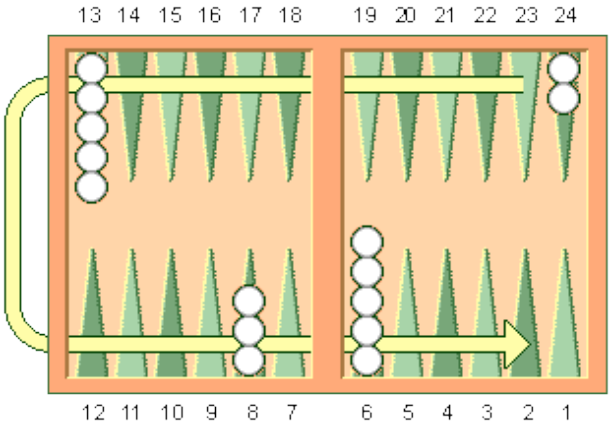
Figura 1 - Tabuleiro de gamão.



Fonte: www.bkgm.com/rules/Portugese/rules.html

3. Os tabuleiros internos são separados dos tabuleiros externos por uma faixa no centro do tabuleiro chamada de barra (*bar*, em inglês).
4. Os pontos (ou casas) são enumerados para cada jogador, começando no tabuleiro interno de cada um, conforme a figura 2.
5. O ponto mais distante do tabuleiro interno de um jogador é o seu ponto 24, o qual corresponderia, também, ao ponto 1 de seu oponente.

Figura 2 - Pontos enumerados para o jogador branco.



Fonte: www.bkgm.com/rules/Portugese/rules.html

6. Cada jogador possui 15 peças (*checkers* ou homens) de sua própria cor; controla 2 dados de 6 faces para realizar suas jogadas e tem como objetivo conseguir retirar todas as suas peças do tabuleiro antes do adversário.

7. A posição inicial das peças é mostrada na figura 1: 2 no ponto 24 de cada jogador; 5 em cada ponto 13; 3 em cada ponto 8 e 5 em cada ponto 6.

8. Toda partida vale inicialmente apenas 1 ponto.

9. O módulo **DADOPONTOS** representa o cubo duplicador (com os números 2, 4, 8, 16, 32, e 64).

10. Quando o dado é criado através da função **DADPnt_CriarDado**, o caractere '*s*' é atribuído ao **(*DadoPontoCriado)->CorDoJogador** que determina que até então nenhum jogador dobrou o valor da partida em andamento e que todos podem utilizar o dado.

11. Se algum jogador dobrar o valor da partida corrente, o dado fica com a cor daquele que dobrou o valor da partida e armazena em ***valorjogo** o valor da mesma.

1.2 PRIMEIRO JOGADOR

1. Precedendo o início da movimentação das peças, é necessário escolher o jogador que irá efetuar a primeira jogada.

2. Para escolher o primeiro jogador, cada um dos jogadores deve lançar apenas 1 dos seus dados.

3. O jogador que obtiver o maior valor, será o primeiro a jogar.

4. No caso dos dados saírem com os valores iguais (dobrados), cada jogador deve lançar o seu dado novamente.

5. O valor dos dados para efetuar a primeira jogada corresponderá ao valor dos 2 dados lançados para definir o primeiro jogador.

Exemplo: O jogador A lança um dado e obtém o valor 6. O jogador B lança um dado e obtém o valor 6 também, ou seja, os valores foram dobrados. O jogador A lança novamente um dado e obtém agora o valor 3. O jogador B lança um dado e obtém o valor 6. O jogador B será o primeiro a jogar com os valores dos dados 3 (valor do dado obtido pelo jogador A) e 6 (valor do dado obtido pelo jogador B).

1.3 MOVIMENTAÇÃO DAS PEÇAS

1. Cada jogador, na sua respectiva chance, deve lançar seus 2 dados e efetuar obrigatoriamente a movimentação das suas peças de acordo com os valores obtidos nos dados, e assim sucessivamente até o término da partida.

2. Os números em cada um dos dados constituem movimentos separados. Por exemplo: se um jogador tira 5 e 3, ele pode mover um de seus *checkers* por uma distância de 5 casas para uma casa livre e um outro *checker* por 3 pontos até uma casa também livre, ou ele pode mover um único *checker* por um total de 8 pontos até uma casa livre. Porém, nesse último caso, também deve estar livre pelo menos um dos pontos intermediários do movimento (nesse caso, ou a casa que se distancia 5 pontos da casa em que se encontra o *checker*, ou a casa que se distancia 3 pontos da casa em que se encontra o *checker* que se pretende mover por 8 pontos até uma casa livre).

3. Cada jogador sempre movimenta as peças em um único sentido (um no sentido horário, e o outro no sentido anti-horário), caminhando no sentido de seu próprio ponto 24 para seu próprio ponto 1.

4. Caso o jogador atual não consiga efetuar jogadas permitidas com os valores obtidos nos dados, este deve passar a sua vez para o adversário.

5. Quando somente um dos números obtidos com os dados pode ser legalmente jogado, o jogador deverá jogar somente esse número.

6. Se qualquer um dos números obtidos com os dados pode ser jogado, porém não ambos, o jogador deverá jogar o número maior.

7. A casa de destino de uma peça nunca pode ser uma que já tenha 2 ou mais peças do adversário.

8. Se a casa de destino estiver livre ou aberta, ou seja, tiver somente 1 peça adversária, esta é capturada, sai do tabuleiro e vai para a “barra”.

9. Quando um jogador tem uma ou mais peças capturadas, ele só pode fazer movimentos de resgate, ou seja, movimentos que tirem suas peças capturadas da “barra” e as coloquem em uma casa novamente (obrigatoriamente no tabuleiro interno do adversário), sempre obedecendo a regra 7.

10. As peças na barra são movidas para uma casa livre no tabuleiro interno do oponente, correspondente a um dos números tirados nos dados.

11. Se um jogador tem mais de uma peça na barra e os números dos dados não permitem que ele reentre com todas, ele deve entrar com tantas quantas os números dos dados permitirem, deixando as demais ainda na barra.

12. Após a reentrada do último homem da barra, qualquer número dos dados que não tenha sido usado para esgotar as reentradas deve ser usado para mover qualquer *checker*, incluindo os que acabaram de reentrar, desde que, obviamente, o *checker* tenha um movimento legal a ser feito (vide regra 7).

13. Se um jogador obtém nos seus 2 dados valores iguais (dobrados), ele tem o direito de fazer 4 jogadas com os valores dos dados ao invés das 2 jogadas rotineiras.

14. No caso de dobras (ou duplos), se nem todos os 4 números podem ser jogados, o jogador deve jogar o máximo possível deles.

Exemplo: Tira-se um 5-5 nos dados, porém é legalmente impossível realizar 4 pulos de 5 pontos. Se, no entanto, 3 pulos são legalmente possíveis, o jogador é obrigado a realizar todos os 3.

1.4 RETIRADA DAS PEÇAS

1. Um jogador só pode iniciar a retirada das peças quando todas elas estiverem no seu próprio tabuleiro interno (também conhecido como seção interna).
2. Para retirar uma peça, o jogador deve obter nos dados o valor equivalente ao número da casa em que se encontra uma de suas peças.
3. Caso obtenha nos dados um valor maior do que o necessário pelas peças mais distantes no tabuleiro, o jogador deve fazer um movimento legal usando um *checker* que se encontre em um ponto de número mais alto que o indicado pelo dado.
4. Se não houver *checker* em um ponto de número mais alto que o indicado pelo dado, o jogador é obrigado a retirar um *checker* do ponto de mais alto número que ele ainda ocupe.
5. Um jogador não está obrigado a retirar qualquer peça se ele tem outro(s) movimento(s) legal(is) além do(s) de retirada.
6. Se, durante o processo de retirada, um *checker* é capturado, o jogador é obrigado a usar seus dados para tentar trazê-lo de volta ao seu próprio tabuleiro interno, e não poderá fazer mais retiradas até que consiga trazer o homem capturado de volta para casa.

1.5 DOBRANDO E REDOBRANDO

1. No início da partida, qualquer jogador pode dobrar o valor da mesma.
2. Um jogador pode dobrar somente quando é sua vez de jogar e antes de lançar seus dados naquele turno.
3. O jogador que não realizou o dobre passa a ser, então, o dono do cubo duplicador, e somente ele poderá fazer o próximo dobre na partida em andamento.
4. Subsequentes dobres na mesma partida são chamados de redobres.
5. Os jogadores podem dobrar a partida até que a mesma atinja o valor de 64 pontos.

1.6 OBJETIVO

1. O objetivo do jogo é mover todas as peças para seu tabuleiro interno e, então, fazer a retirada final de cada uma delas do tabuleiro.

1.7 VENCEDOR

1. O jogador que retirar todas as suas peças do tabuleiro antes do adversário será o vencedor

1.8 VENCENDO POR GAMÃO OU BACKGAMMON

1. Terminada a partida, se o jogador que a perdeu conseguiu fazer a retirada final de pelo menos um de seus *checkers*, o vencedor ganhará somente o valor mostrado no cubo duplicador (ou 1 ponto, caso ninguém tenha dobrado na partida).
2. Se o perdedor não fez a retirada final de nenhum de seus *checkers*, ele terá perdido de gamão. Nesse caso, o vencedor receberá o dobro dos pontos mostrados pelo cubo duplicador.
3. Se o perdedor não retirou nenhum de seus *checkers* e ainda tem peças na barra ou no tabuleiro interno do vencedor, a derrota será de *backgammon*, dando, ao vencedor, o triplo do valor mostrado pelo cubo duplicador.

2 REQUISITOS NÃO FUNCIONAIS

Segundo Staa (2000, p. 339): “**Requisitos Não Funcionais** são propriedades que o artefato deve possuir.”

2.1 ROBUSTEZ

1. Todos os dados de entrada (e.g.: movimentação de peças) são validados pelo jogo.
2. Caso alguma entrada não seja válida, uma mensagem de erro será exibida ao jogador informando que o dado inserido é inválido e ele terá nova oportunidade de digitar.

Exemplo: Após o jogador A lançar os seus dados, o jogo fornece ao mesmo as opções de movimentação de peças possíveis (e.g.: 1- mover para a casa 6; 2- mover para a casa 7; 3- mover para a casa 11). Se o jogador A digitar uma opção inexistente, uma mensagem de erro será exibida e ele deverá digitar novamente a opção escolhida até que ela seja válida.

2.2 CORRETUDE

1. Todos os módulos devem ser testados individualmente, tanto o comportamento normal quanto o anormal de cada função dos módulos.
2. Restrição: deve-se reutilizar o “Arcabouço” de teste automatizado.

2.3 REUSO

1. Com o objetivo de acelerar o processo de reutilização de projeto, implementação e teste, deve-se maximizar a reutilização de módulos.

2. Restrição: deve-se reutilizar os módulos LISTA, GENERICO e TST_EXPEC do “Arcabouço”.

2.4 MANUTENIBILIDADE

1. A manutenção do programa é fácil, pois obedece padrões de modularidade.
2. Os módulos (DADO, DADOPONTOS, PECA, PECASCAPTURADAS (BAR), PECASFINALIZADAS, TABULEIRO E JOGO) estão devidamente comentados e seguem padrões de documentação, nomeação de variáveis e funções.

3 REQUISITOS INVERSOS

Segundo Clua: “**Requisitos Inversos** representam funcionalidades que estão fora do escopo da solução.”

3.1 GERAL

1. O jogo somente será implementado em idioma nacional.

4 RESTRIÇÕES

Segundo Staa (2000, p. 339): “**Restrições** são condições que restringem a liberdade de escolha de alternativas de construção do artefato sendo especificado.”

4.1 GERAL

1. O programa deverá ser redigido em Linguagem ANSI C.
2. Deve-se reutilizar o “Arcabouço” de teste automatizado.
3. Deve-se reutilizar os módulos LISTA, GENERICO e TST_EXPEC do “Arcabouço”.

4.2 DADO

- Deverá utilizar as bibliotecas **stdio.h**, **stdlib.h**, **string.h**, **assert.h**, **malloc.h** e **time.h** e o header referente a si (**DADO.h**).

4.3 DADOPONTOS

- Deverá utilizar as bibliotecas **stdio.h**, **stdlib.h**, **string.h** e **malloc.h**, o header referente a si (**DADOPONTOS.h**) e **GENERICO.h**.

4.4 PECA

- Deverá utilizar as bibliotecas **stdio.h**, **string.h** e **malloc.h** e o header referente a si (**PECA.h**).

4.5 TABULEIRO

- Deverá utilizar as bibliotecas **stdio.h**, **string.h**, **assert.h**, **memory.h** e **malloc.h** e o header referente a si (**TABULEIRO.h**), **LISTA.h**, **PECA.h** e **GENERICO.h**.

4.6 JOGO

- Deverá utilizar as bibliotecas **stdio.h**, **stdlib.h**, **Windows.h** e os headers **TABULEIRO.h**, **BAR.h**, **DADO.H**, **DADOPONTOS.h**, **LISTA.h**, **PecasFinalizadas.h**, **PECA.h**.

4.7 PECASCAPTURADAS (BAR)

- Deverá utilizar as bibliotecas **stdio.h**, **string.h**, **malloc.h** e os headers **BAR.h**, **LISTA.h** e **PECA.h**.

4.8 PECASFINALIZADAS

- Deverá utilizar as bibliotecas **stdio.h**, **string.h**, **malloc.h** e os headers **LISTA.h** e **PECA.h**.

5 HIPÓTESES

Segundo Garcia, “**Hipóteses** são condições ou características assumidas como satisfeitas externamente à equipe de desenvolvimento.”

5.1 GERAL

1. A quantidade de memória disponível será sempre suficiente para executar o programa do jogo de Gamão.
   a. Mesmo assim, sempre verificamos se a função ***malloc*** retornou *NULL*.
2. O usuário assume unicamente o papel de jogador, ou seja, tem apenas permissão para realizar jogadas na interface do jogo.
3. O usuário não possui permissão para editar funções anteriormente implementadas nos módulos.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS – ABNT. **NBR 10719**: apresentação de relatórios técnico-científicos. Rio de Janeiro, 1989. 9 p.

CLUA, Esteban Walter Gonzalez. **Requisito - Instituto de Computação – UFF**. Disponível em <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CBwQFjAAahUKEwjTkcOO397IAhVEEpAKHahUCAU&url=http%3A%2F%2Fwww.ic.uff.br%2F~esteban%2Ffiles%2Freq1.ppt&usg=AFQjCNHuHWXz-ANQFRpHaKgomYHzbbePEw&sig2=PtjuxRnVfXIhZRdcAgYU-g>. Acesso em: 25 outubro 2015.

ENGENHARIA PITÁGORAS. **Modelo de Relatório padrão ABNT NR 10719 – Word**. Desenvolvida por GOMES, Cristiano. Disponível em **<**https://engepit.wordpress.com/2014/09/21/modelo-de-relatorio-padrao-abnt-nr-10719-word/>. Acesso em: 27 setembro 2015.

STAA, Arndt von. **Programação Modular**: Desenvolvendo programas complexos de forma organizada e segura. Rio de Janeiro: Campus, 2000.

GARCIA, Alessandro. **Aula 9: Especificação de Requisitos**. Disponível em <http://www.inf.puc-rio.br/~inf1301/docs/2013_1/INF1301_Aula09_Requisitos_2013_1.pdf>. Acesso em: 25 outubro 2015.

BIBLIOGRAFIA RECOMENDADA

STAA, Arndt von. **Programação Modular**: Desenvolvendo programas complexos de forma organizada e segura. Rio de Janeiro: Campus, 2000.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO. **Aula 7 – Especificação de Requisitos e Arquitetura**. Desenvolvida por STAA, Arndt von. Disponível em: <http://www.inf.puc-rio.br/~inf1301/docs/2014_2/INF1301_Aula07_Requisitos_Arquitetura_2014_2_final.pdf>. Acesso em: 20 setembro 2015.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO. **Aula 10 - Assertivas**. Desenvolvida por STAA, Arndt von. Disponível em: <http://www.inf.puc-rio.br/~inf1301/docs/2014_2/INF1301_Aula10_Assertivas.pdf>. Acesso em: 25 setembro 2015.