Projeto Ludo

Especificação de requisitos de software

Kalleo Ouriques
Rafael Luis Sol Veit Vargas
Thiago Augusto Bewiahn

1. Introdução

1.1. Objetivo

Desenvolvimento de um programa distribuído para executar partidas do jogo de tabuleiro Ludo de forma usuário contra usuário;

1.2. Definições, abreviaturas

RF: requisito funcional;

Casa: referente ao local no tabuleiro com a cor de cada jogador onde cada um dos peões permanecem no início de jogo, antes de serem colocados no caminho;

Caminho: Conjunto de posições ordenadas percorridas por um peão;

Caminho geral: caminho percorrido pelos peões de todos os jogadores, porém com pontos de início diferentes para cada jogador;

Caminho final: caminho colorido com a cor de cada jogador. Um peão só pode entrar no caminho final após terminar de percorrer o caminho geral.

2. Visão geral

2.1. Arquitetura do programa

Cliente-servidor distribuído;

2.2. Premissas

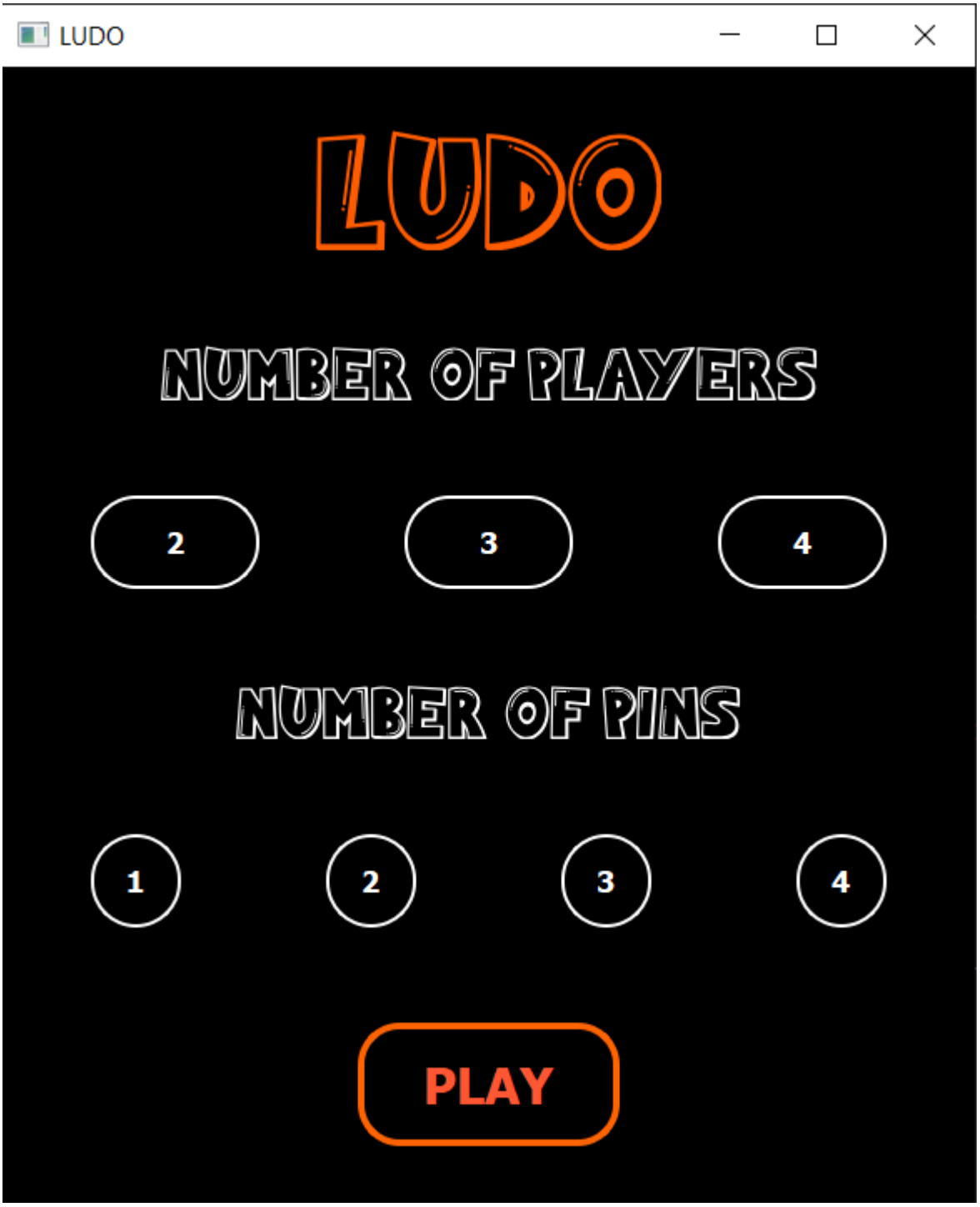
- O programa deve ser desenvolvido em Python;
- O framework DOG deve ser usado para suportar execução distribuída;

- Especificação de projeto baseada em UML segunda versão deve ser produzida.

3. Requisitos

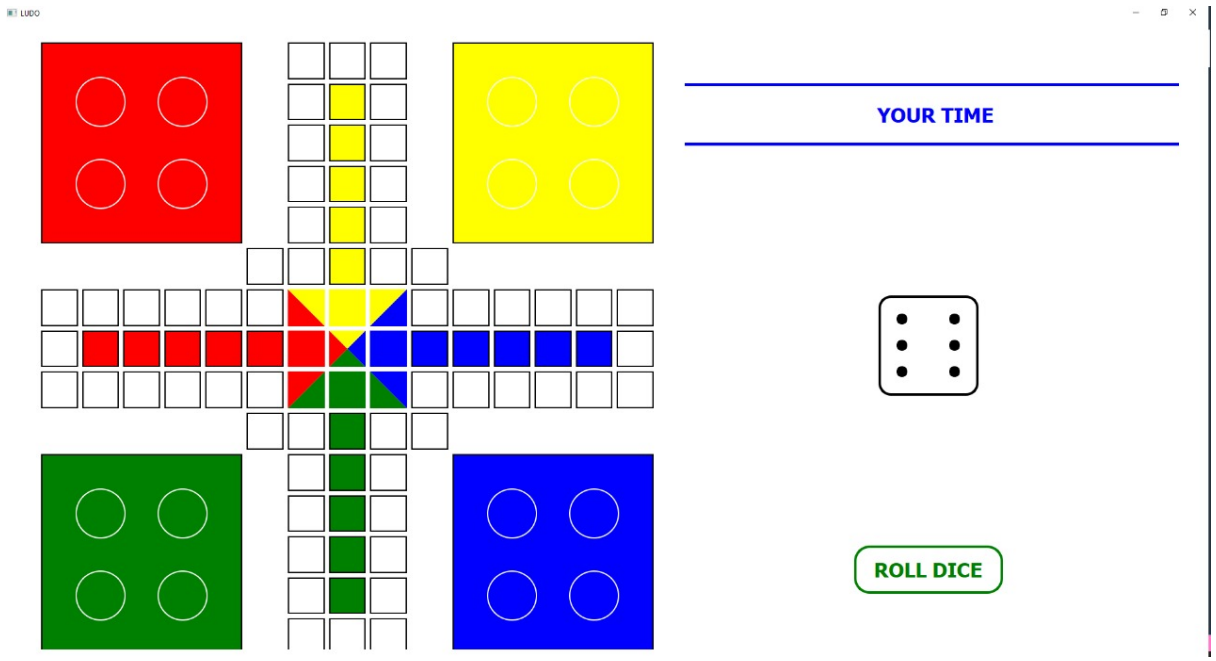
3.1. Requisitos funcionais

Requisito funcional 1: Iniciar programa: ao entrar em execução, o programa deve solicitar o nome do jogador e realizar a conexão com o DOG Server. Após isso, deve apresentar uma interface para escolha do número de jogadores (de 2 a 4) e de peões (1 a 4), conforme imagem abaixo:



Requisito funcional 2: Iniciar o jogo: o programa deve apresentar em sua primeira interface um botão para iniciar a partida. Caso haja jogadores suficientes conectados quando o botão de jogar for clicado, isto é, no mínimo o número de jogadores informado nesta mesma interface, uma nova interface será inicializada para começar o jogo (ver esboço da interface abaixo). Senão, o programa é encerrado. Se a partida foi iniciada com sucesso, o DOG Server retornará a ordem

dos jogadores. Logo, o tabuleiro será inicializado com o primeiro a jogar habilitado para executar o RF 4.



Requisito funcional 3: Reiniciar partida: o programa deve apresentar um botão para reiniciar partida, isto é, todos os peões de volta às suas respectivas casas e necessitando condições específicas para poderem ser retirados (ver RF 5). A ordem de jogo deverá ser embaralhada. Esta funcionalidade só deve estar habilitada ao término do jogo.

Requisito funcional 4: Rolar o dado: o jogador, no seu turno, deve poder clicar num botão que simula o lançamento de um dado. O jogador pode jogar o dado novamente caso o seu valor seja 6 e selecionar uma peça de acordo com o especificado no RF 5.

Requisito funcional 5: Selecionar peão: o jogador, no seu turno, deve poder selecionar uma peça para mover/retirar da casa após executar o RF 4. Caso nenhum peão tenha sido retirado da casa ainda, o jogador poderá fazê-lo caso tire 1 ou 6 no dado. Caso contrário, isso só deve ser permitido quando o valor do dado for 6. Deve haver uma indicação sobre o peão selecionado para o jogador, assim como um botão de confirmação de seleção de peão. Ao confirmar, se o peão selecionado estiver na casa, ele deverá ir para a posição inicial do jogador no tabuleiro. Senão, o peão deve andar o valor do dado em posições no tabuleiro, respeitando as regras descritas no apêndice.

Requisito funcional 6: Receber início: o programa deve poder receber e processar corretamente a notificação de início de partida fornecida pelo DOG Server. Assim que receber a ordem dos jogadores pelo servidor, deve iniciar o procedimento descrito em RF 2, habilitando o primeiro jogador para executar o procedimento em RF 4.

Requisito funcional 7: Receber turno: o programa deve poder receber as jogadas de um turno de outro jogador ao final do turno. As informações do turno devem conter no mínimo o valor tirado no dado assim como o peão que foi movido, se houver, e sua posição. Se houver movimentação de peão, deve-se avaliar se ela gera condição de encerramento do jogo. Se sim, o jogo se encerra com o nome do vencedor na tela juntamente com um botão para executar o RF 3. Senão, o próximo jogador deverá poder começar seu turno.

Requisito funcional 8: Receber desistência: o programa deve poder receber notificação de desistência por parte de um dos jogadores de forma remota e vindo do DOG Server. Caso haja menos do que 2 jogadores, o programa encerra. Senão, o jogador desistente é retirado da ordem de jogadas, seus peões são retirados do tabuleiro e o próximo jogador é habilitado para executar seu turno.

3.2 Requisitos não funcionais

Requisito não funcional 1: Tecnologia de interface gráfica: o framework PyQt deve ser utilizado para a construção da interface gráfica.

Requisito não funcional 2: Suporte para a especificação de projeto: a especificação de projeto e seus diagramas devem ser produzidos utilizando o software Visual Paradigm.

Requisito não funcional 3: Modelo da interface gráfica: a interface gráfica deve ser construída com base nas imagens mostradas em RF 1 e RF 2.

Apêndice: Regras do jogo Ludo

O jogo pode ser jogado de 2 a 4 jogadores, com uma variação para escolha do número de peões (1 a 4). Para vencer, o jogador deverá levar todos os seus peões até a última posição do caminho, ou seja, até o triângulo da cor do jogador no meio do tabuleiro. O jogo se encerra após o primeiro jogador vencer. Ao iniciar a partida e tendo a ordem dos jogadores determinada (nesse caso, será dada pelo DOG

Server), para retirar o primeiro peão da casa, o jogador deve rolar o dado e apenas poderá colocar um peão no caminho geral caso tenha tirado 1 ou 6. Tendo retirado o primeiro peão, novos peões podem apenas ser retirados caso o valor do dado seja 6, porém isso não é obrigatório. Se, ao início de seu turno, o jogador já possui algum peão no caminho, pode rolar o dado e movimentar um dos peões *n* posições, onde *n* é o valor tirado no dado. As posições percorridas pelos peões é pré-definida: um peão começa uma posição após a entrada para o caminho específico e precisa dar uma volta completa no tabuleiro para então entrar no caminho específico. Peões só podem entrar no caminho específico de sua respectiva cor, ou seja, ao entrarem, ficam “protegidos” dos peões adversários (ver regra adicional 2). Além disso, há algumas regras adicionais:

1. Sempre que o jogador tirar 6 no dado, ele poderá jogar o dado de novo (inclusive ao retirar um peão da casa);
2. Caso um peão termine sua movimentação na posição de um peão adversário, o peão adversário deverá voltar para casa;
3. Se 2 peões do mesmo jogador estiverem na mesma casa, eles formam uma “barreira”, impedindo peões adversários de avançar. Isso também anula a mecânica esclarecida na regra acima;
4. Se a movimentação levar o peão para a última posição do caminho final de forma não-exata, isto é, o número de posições andadas for menor do que o valor do dado, o peão deve voltar no tabuleiro o número de posições restantes.