Estimativas de Esforço - Monopoly

1. Introdução e Propósito

Este documento detalha o processo e os resultados da estimativa de esforço para o projeto MONOPOLY. Para garantir uma visão abrangente e mitigar as incertezas inerentes ao processo de estimativa, foram empregadas duas metodologias distintas e complementares:

- **Análise de Pontos de Função (APF):** Uma abordagem paramétrica, "top-down", que mede o tamanho funcional do software a partir dos requisitos do usuário.
- **Planning Poker:** Uma abordagem ágil, "bottom-up", baseada em consenso da equipe para estimar a complexidade relativa de cada pacote de trabalho da EAP. O objetivo é estabelecer uma linha de base de esforço que servirá como fundamento para o planejamento de custos e do cronograma do projeto.

2. Análise de Pontos de Função (APF)

2.1. Propósito da Análise

Este documento apresenta a estimativa de tamanho do projeto MONOPOLY utilizando a metodologia de Análise de Pontos de Função (APF). O objetivo é quantificar as funcionalidades descritas no Documento de Definição de Escopo para prover uma estimativa paramétrica do esforço de desenvolvimento, que servirá de base para o planejamento e como contraponto à estimativa ágil (Planning Poker).

2.2. Fronteira da Aplicação

A fronteira da aplicação engloba todas as funcionalidades necessárias para uma partida completa do jogo Monopoly em um ambiente de desktop local. Interações com sistemas externos, bases de dados online ou outras aplicações estão fora desta fronteira. O usuário interage diretamente com a aplicação através de uma interface gráfica.

2.3. Identificação e Classificação das Funções

As funções foram identificadas a partir dos Requisitos Funcionais (REQ-FUNC) e classificadas nos cinco tipos definidos pela APF.

2.3.1. Arquivos Lógicos Internos (ALI)

Grupos de dados lógicos mantidos e gerenciados pela aplicação.

ID	Nome do ALI	Descrição	Complexidade
ALI-01	Dados dos Jogadores	Mantém o estado de cada jogador (nome, peça, dinheiro, posição, propriedades, estado "preso").	Média
ALI-02	Estado do Tabuleiro	Mantém o estado de cada uma das 40 propriedades (dono, nível de construção, hipotecada).	Média
ALI-03	Monte de Cartas	Mantém o estado das cartas de Sorte e Cofre (ordem, cartas utilizadas).	Baixa

2.3.2. Entradas Externas (EE)

Ações do usuário que alteram o estado de um ou mais ALIs.

ID	Nome da EE	Descrição	Complexidade
EE-01	Configurar Partida	Jogador insere nomes e escolhe peças antes do início (REQ-FUNC-1.1).	Média
EE-02	Lançar Dados	Jogador aciona o lançamento de dados para mover sua peça (REQ-FUNC-2.2).	Baixa
EE-03	Comprar Propriedade	Jogador confirma a compra de uma propriedade sem dono (REQ-FUNC-3.1).	Baixa
EE-04	Dar Lance em Leilão	Jogador submete um valor durante um leilão (REQ-FUNC-3.2).	Média
EE-05	Construir Edifício	Jogador solicita a construção de uma casa ou hotel (REQ-FUNC-4.1).	Média
EE-06	Vender Edifício	Jogador solicita a venda de uma casa ou hotel	Média

		(REQ-FUNC-4.3).	
EE-07	Propor Transação	Jogador inicia uma negociação de propriedade com outro (REQ-FUNC-3.4).	Média
EE-08	Hipotecar Propriedade	Jogador solicita a hipoteca de uma propriedade (REQ-FUNC-5.2).	Baixa
EE-09	Pagar Hipoteca	Jogador paga para resgatar uma propriedade hipotecada (REQ-FUNC-5.2).	Baixa
EE-10	Pagar Fiança	Jogador opta por pagar para sair da cadeia (REQ-FUNC-5.3).	Baixa
EE-11	Usar Carta "Sair da Cadeia"	Jogador opta por usar a carta para sair da cadeia (REQ-FUNC-5.3).	Baixa

2.3.3. Saídas Externas (SE)

Informações apresentadas ao usuário que envolvem processamento ou cálculo

e cujo estado principal é manter a visão do jogo atualizada.

ID	Nome da SE	Descrição	Complexidade
SE-01	Display Principal do Jogo	Renderização do estado do tabuleiro, posições das peças e informações dos jogadores (REQ-FUNC-2.2, 3.3).	Alta
SE-02	Notificação de Ação/Evento	Mensagens informando o resultado de uma ação (ex: "Você pagou $X de aluguel").	Média
SE-03	Exibição de Carta Sorte/Cofre	Apresenta a instrução de uma carta retirada do monte (REQ-FUNC-2.5).	Baixa
SE-04	Tela de Vencedor	Exibição da mensagem final declarando o vencedor (REQ-FUNC-5.5).	Baixa

2.3.4. Consultas Externas (CE)

Consultas diretas a informações de um ALI, sem processamento complexo.

ID	Nome da CE	Descrição	Complexidade
CE-01	Consultar Título de Posse	Jogador clica em uma propriedade para ver seus detalhes (aluguel, custo de construção).	Baixa
CE-02	Consultar Ativos de Jogador	Exibir um resumo do dinheiro e propriedades de qualquer jogador.	Média

2.3.5. Arquivos de Interface Externa (AIE)

Para este projeto, o número de AIEs é **zero**, pois a aplicação não interage com dados mantidos por outros sistemas.

2.4. Cálculo dos Pontos de Função Não-Ajustados (UFP)

Utilizando os pesos padrão, a contagem de UFP é a seguinte:

Tipo de Função	Qtd Baixa	Qtd Média	Qtd Alta	Peso (B/M/A)	Subtotal
ALI	1	2	0	7 / 10 / 15	(1 * 7) + (2 * 10) = **27**
EE	6	5	0	3 / 4 / 6	(6 * 3) + (5 * 4) = **38**

SE	2	1	1	4 / 5 / 7	(2 * 4) + (1 * 5) + (1 * 7) = **20**
CE	1	1	0	3 / 4 / 6	(1 * 3) + (1 * 4) = **7**
AIE	0	0	0	5 / 7 / 10	**0**
Total UFP					**92**

2.5. Fator de Ajuste de Valor (VAF) e Pontos de Função Finais (PF)

O VAF é calculado com base em 14 Características Gerais do Sistema (CGS). Para esta estimativa inicial, assumimos uma influência média do ambiente, onde a soma dos graus de influência (DI) é aproximadamente **28** (média de 2 para cada uma das 14 características).

- **Fator de Ajuste (VAF):** $0.65+(0.01*\sum DI)=0.65+(0.01*28)=0.65+0.28=0.93$
- **Pontos de Função Finais (PF):** $UFP*VAF=92*0.93\approx86$ PF

2.6. Estimativa de Esforço

Para converter a medida de tamanho (PF) em esforço (horas), adotamos a premissa de que a produtividade da equipe, usando a tecnologia Python/Pygame para este tipo de projeto, é de **10 horas por Ponto de Função**.

- **Esforço Total Estimado (Horas):** $PF*(\text{Horas por PF})=86*10=860$ horas

Esta estimativa paramétrica servirá como uma referência macro para ser comparada com a estimativa bottom-up que será realizada via Planning Poker.

3. Planning Poker

3.1. Metodologia

Esta etapa registra os resultados da sessão de estimativa ágil, utilizando a técnica do Planning Poker. O objetivo foi obter uma estimativa de esforço/complexidade "bottom-up" para cada pacote de trabalho definido na Estrutura Analítica do Projeto (EAP).

- **Unidade de Medida:** Story Points (SP).
- **Escala Utilizada:** Sequência de Fibonacci modificada (0, 1, 2, 3, 5, 8, 13, 20).
- **Item de Referência:** O pacote de trabalho **"1.4.1.3 Implementar Lançamento de Dados e Movimentação"** foi definido consensualmente como a nossa linha de base, com um valor de **3 SP**. Ele envolve lógica clara (gerar números aleatórios, atualizar a posição do jogador) e uma atualização visual simples, representando uma tarefa de complexidade moderada e bem compreendida. Todas as outras estimativas foram feitas em relação a este item.

3.2. Tabela de Estimativas por Pacote de Trabalho

A seguir estão os resultados da estimativa para cada pacote de trabalho relevante da EAP.

ID da EAP	Pacote de Trabalho	Story Points (SP)	Justificativa da Estimativa
1.2.3	**Configuração do Ambiente de Dev**	**5**	Envolve mais do que git init. Inclui a definição da estratégia de branches, configuração de .gitignore e garantia

			de que todos os membros consigam rodar o projeto.
1.3.1	**Desenho da Arquitetura de Software**	**8**	Tarefa de alta complexidade conceitual. Definir as classes, responsabilidades e interações corretamente no início é crucial e exige debate e prototipagem.
1.3.2	**Design da UI e Coleta de Ativos**	**3**	Trabalho direto de encontrar/criar imagens e montar um mockup visual simples. Não envolve lógica complexa.
	--- Subtotal Fases Iniciais ---	**16 SP**	
1.4.1.1	Implementar Estrutura Tabuleiro/Propriedades	**5**	Requer a criação de uma estrutura de dados para armazenar o estado de 40 casas, incluindo nomes, preços, aluguéis, etc. É um trabalho de

			digitação e estruturação, mas volumoso.
1.4.1.2	Implementar Classes de Jogador/Peças	**3**	Relativamente simples. Classes para guardar o estado de cada jogador.
1.4.1.3	Implementar Movimentação	**3 (REF)**	Nosso item de referência. Lógica clara de dados e renderização.
1.4.1.4	Implementar Lógica do Ponto de Partida	**1**	A tarefa mais simples: um if que verifica a posição do jogador e adiciona $200.
1.4.1.5	Criar Janela Principal e Renderizar	**5**	Envolve a configuração inicial do Pygame, o carregamento da imagem de fundo e a criação do loop principal do jogo (game loop). É uma peça fundamental da arquitetura.
	--- Subtotal Iteração 1 ---	**17 SP**	

1.4.2.1	Implementar Compra de Propriedade	**5**	Envolve a interação com o jogador, a validação de fundos, a atualização do dono da propriedade (ALI-02) e a transferência de dinheiro (ALI-01).
1.4.2.2	Implementar Pagamento de Aluguel	**8**	Mais complexo do que parece. A lógica precisa calcular o aluguel correto (varia com casas, hotéis, monopólio), identificar o proprietário e gerenciar a transação.
1.4.2.3	Implementar Sistema de Cartas	**5**	Requer carregar as 32 instruções das cartas, embaralhá-las e executar a ação correspondente, que pode ser desde um simples pagamento até "Vá para a Cadeia".
1.4.2.4	Implementar Regra da "Dupla"	**3**	Lógica de estado que precisa controlar o

			número de duplas consecutivas e interagir com a mecânica da cadeia.
1.4.2.5	Implementar Impostos e Taxas	**2**	Lógica simples e direta de débito de valor fixo do jogador.
	--- Subtotal Iteração 2 ---	**23 SP**	
1.4.3.1	Implementar Construção de Casas/Hotéis	**13**	Tarefa de alta complexidade. Requer a validação da posse de um monopólio, aplicação da regra de construção uniforme e gerenciamento do estoque de edifícios do banco.
1.4.3.2	Implementar Lógica de Hipoteca	**8**	Envolve a interação com o banco, o cálculo de juros para resgate e a marcação do estado "hipotecado" da propriedade.
1.4.3.3	Implementar Mecânica da Cadeia	**8**	Lógica de estado complexa. O sistema precisa saber que um

			jogador está preso e, a cada turno dele, oferecer as diferentes opções de saída (pagar, usar carta, tentar a sorte nos dados).
1.4.3.4	Implementar Falência e Fim de Jogo	**5**	Lógica para verificar a condição de falência, gerenciar a transferência de todos os ativos (incluindo propriedades hipotecadas) e declarar o vencedor.
1.4.3.5	Refinamento da Interface	**5**	Embora não seja uma funcionalidade central, polir a UI com feedback claro ao usuário exige um esforço considerável de pequenas implementações.
	--- Subtotal Iteração 3 ---	**39 SP**	
1.5.1	**Testes de Integração e Sistema**	**13**	É um esforço significativo garantir que todas as regras,

			que foram desenvolvidas separadamente, interajam corretamente. Simular uma partida completa revelará muitos bugs de integração.
1.5.2	**Correção de Bugs Finais**	**8**	Com base nos testes, sempre há uma fase de correção concentrada. É um esforço que precisa ser explicitamente planejado.
	--- Subtotal Fase de Transição ---	**21 SP**	

4. Conclusão e Análise Comparativa

O processo de estimativa resultou em duas métricas distintas para o tamanho do projeto:

- **Resultado da APF:** Um esforço total estimado de 860 horas.
- **Resultado do Planning Poker:** Um tamanho total de 116 Story Points.

É fundamental entender que essas métricas não são diretamente conversíveis, mas complementares. A estimativa em horas, derivada da APF, nos forneceu a base para o planejamento de custos e para o orçamento do projeto, representando uma visão macro do esforço. A estimativa em Story Points, por

sua vez, será a unidade de trabalho utilizada no dia a dia para o planejamento das Sprints, medição da velocidade (velocity) da equipe e monitoramento do progresso através do Gráfico de Burndown.

A utilização de ambas as técnicas nos dá maior confiança na nossa compreensão do escopo e do esforço necessário, alinhando a visão gerencial com a visão técnica da equipe de desenvolvimento.