Pontifícia Universidade Católica de Minas Gerais

Curso de Engenharia de Software

Gabriel Henrique Miranda Rodrigues

Plano de Experimento

Comparação entre Test-Driven Development e Desenvolvimento Tradicional na Qualidade e Produtividade de Software

Belo Horizonte
2025

Sumário

1 Identificação Básica

1.1 Título do Experimento

Comparação entre Test-Driven Development (TDD) e Desenvolvimento Tradicional: Efeitos na Qualidade do Código e Produtividade de Desenvolvedores em Ambiente Corporativo.

1.2 Código de Identificação

EXP-TDD-2025-001

1.3 Versão do Documento

Versão	Data	Descrição
1.0	21/11/2025	Versão inicial do plano

Tabela 1: Histórico de versões

1.4 Datas

Data de criação: 21/11/2025
Última atualização: 03/12/2025

1.5 Autor

Gabriel Henrique Miranda Rodrigues
Área: Engenharia de Software
E-mail: gabriel.rodrigues.1447565@sga.pucminas.br

1.6 Responsável Principal

Gabriel Henrique Miranda Rodrigues, sob orientação do Prof. [Nome do Orientador].

1.7 Projeto Relacionado

Este experimento faz parte do Trabalho de Conclusão de Curso (TCC) do curso de Engenharia de Software da PUC Minas, com foco em práticas de desenvolvimento e qualidade de software.

2 Contexto e Problema

2.1 Descrição do Problema

No desenvolvimento de software, equipes frequentemente enfrentam o desafio de equilibrar velocidade e qualidade. Defeitos descobertos tardiamente geram retrabalho e atrasos. O Test-Driven Development (TDD) propõe escrever testes antes do código, prometendo melhorar a qualidade. Porém, existe a percepção de que isso torna o desenvolvimento mais lento.

Este experimento busca comparar TDD e desenvolvimento tradicional em termos de defeitos e tempo de desenvolvimento.

2.2 Contexto Organizacional e Técnico

O experimento será realizado em uma empresa de desenvolvimento que utiliza C# e .NET Core. A empresa possui desenvolvedores de diferentes níveis de experiência. O ambiente inclui Visual Studio, Git e frameworks de teste como xUnit.

2.3 Trabalhos Prévios

Erdogmus et al. (2005) observaram que TDD produziu código com menos defeitos em experimento com estudantes. Rafique e Misic (2013) fizeram revisão sistemática indicando melhoria na qualidade, mas resultados inconclusivos sobre produtividade. Fucci et al. (2017) questionaram se os benefícios vêm do TDD ou simplesmente de escrever testes.

Esses resultados variados justificam novos estudos em contextos específicos.

2.4 Referencial Teórico

O TDD foi popularizado por Kent Beck como parte do Extreme Programming. Baseia-se no ciclo Red-Green-Refactor:

Red: Escrever um teste que falha.

Green: Escrever código mínimo para o teste passar.

Refactor: Melhorar o código mantendo os testes passando.

No desenvolvimento tradicional, o código é escrito primeiro e os testes depois, servindo principalmente como verificação.

3 Objetivos e Questões

3.1 Objetivo Geral

Analisar o Test-Driven Development com o propósito de comparar seus efeitos em relação ao desenvolvimento tradicional, com respeito à qualidade do código e produtividade, do ponto de vista de desenvolvedores de software, no contexto de implementação de funcionalidades em C# .NET Core em ambiente corporativo.

3.2 Objetivos Específicos

O1: Comparar a quantidade de defeitos no código produzido com TDD versus desenvolvimento tradicional.

O2: Comparar o tempo necessário para completar tarefas usando cada abordagem.

O3: Avaliar a cobertura de testes alcançada em cada abordagem.

O4: Identificar a percepção dos desenvolvedores sobre cada técnica.

3.3 Questões de Pesquisa

Q1: O código desenvolvido com TDD apresenta menos defeitos?

Q2: Existe diferença no tempo de desenvolvimento entre as abordagens?

Q3: A cobertura de testes é maior com TDD?

Q4: Qual a percepção dos desenvolvedores sobre cada técnica?

3.4 Métricas

Questão	**Métrica**	**Unidade**	**Fonte**
Q1	Defeitos detectados	Quantidade	Testes de aceitação
Q2	Tempo de conclusão	Minutos	Registro manual
Q3	Cobertura de código	Percentual	Ferramenta de cobertura
Q4	Percepção dos desenvolvedores	Escala Likert	Questionário

Tabela 2: Métricas associadas às questões

4 Escopo e Contexto do Experimento

4.1 Escopo Funcional

Incluído: Implementação de funções de validação e manipulação de dados em C# (validação de CPF, e-mail, cálculos), criação de testes unitários, refatoração básica.

Excluído: Integração com banco de dados, interfaces gráficas, APIs externas, deploy.

4.2 Contexto do Estudo

Empresa de pequeno a médio porte com desenvolvedores profissionais de nível júnior a sênior. As tarefas serão exercícios isolados com impacto em projetos reais.

4.3 Premissas

Participantes possuem conhecimento básico de C# e .NET Core.
Ambiente de desenvolvimento estará configurado e funcional.
Participantes terão disponibilidade sem interrupções.
Todos receberão treinamento sobre TDD antes do experimento.

4.4 Restrições

Número de participantes limitado aos disponíveis na empresa.
Tempo total não pode exceder um turno de trabalho (4 horas).
Tarefas devem ser simples para conclusão no tempo disponível.
Uso apenas de ferramentas já disponíveis.

4.5 Limitações Previstas

Experimento em uma única empresa limita generalização dos resultados.
Amostra de conveniência pode não representar todos os desenvolvedores.
Tarefas simplificadas não capturam toda complexidade de projetos reais.

5 Stakeholders e Impacto Esperado

5.1 Stakeholders Principais

Desenvolvedores participantes.
Gestores e líderes técnicos da empresa.
Orientador acadêmico.

5.2 Interesses e Expectativas

Desenvolvedores: aprender sobre TDD e entender seus benefícios.
Gestores: evidências para decisão sobre adoção do TDD.
Orientador: experimento com rigor metodológico.

5.3 Impactos Potenciais

Participantes dedicarão 3 a 4 horas, temporariamente reduzindo disponibilidade para outras atividades. Como benefício, receberão treinamento em TDD. Resultados poderão influenciar práticas de desenvolvimento na empresa.

6 Riscos e Critérios de Sucesso

6.1 Riscos de Alto Nível

Número insuficiente de participantes.
Desistências durante o experimento.
Problemas técnicos no ambiente.
Participantes não seguirem o protocolo corretamente.
Mudança de prioridades na empresa.

6.2 Critérios de Sucesso

Pelo menos 80% dos participantes completarem todas as tarefas.
Dados suficientes para responder às questões de pesquisa.
Protocolo seguido sem desvios significativos.

6.3 Critérios de Parada

Menos de 4 participantes confirmados.
Problemas técnicos críticos não resolvidos.
Empresa retirar apoio ao estudo.
Piloto revelar problemas graves no protocolo.

7 Modelo Conceitual e Hipóteses

7.1 Modelo Conceitual

O modelo baseia-se na ideia de que a técnica de desenvolvimento influencia qualidade e tempo. O TDD teoricamente melhora a qualidade ao forçar o desenvolvedor a pensar nos requisitos antes de codificar, permitindo feedback rápido. Porém, pode aumentar o tempo inicial por exigir testes antes do código funcional.

O desenvolvimento tradicional pode ser mais rápido inicialmente, mas potencialmente resulta em mais defeitos porque os testes são escritos depois.

7.2 Hipóteses Formais

Para defeitos (Q1):

$H_{0,1}$: Não existe diferença na quantidade de defeitos entre TDD e tradicional.

$H_{1,1}$: TDD apresenta menos defeitos que tradicional.

Para tempo (Q2):

$H_{0,2}$: Não existe diferença no tempo de desenvolvimento.

$H_{1,2}$: Existe diferença no tempo de desenvolvimento.

Para cobertura (Q3):

$H_{0,3}$: Não existe diferença na cobertura de testes.

$H_{1,3}$: A cobertura é maior com TDD.

7.3 Nível de Significância

Será adotado $\alpha = 0,05$ para todos os testes. O poder estatístico desejado é 0,80. Com base em estudos anteriores, estima-se que 4 a 10 participantes sejam adequados para um desenho pareado.

8 Variáveis, Fatores e Tratamentos

8.1 Objetos de Estudo

Implementações de funcionalidades em C# .NET Core produzidas pelos participantes, como validadores de dados e funções de cálculo.

8.2 Participantes

Desenvolvedores da empresa, entre 4 e 10 profissionais, com diferentes níveis de experiência.

8.3 Variável Independente

Técnica de desenvolvimento com dois níveis:

Nível 1: Test-Driven Development (TDD)

Nível 2: Desenvolvimento Tradicional

8.4 Tratamentos

Tratamento A (TDD): Participante segue o ciclo Red-Green-Refactor. Escreve teste que falha, implementa código para passar, refatora.

Tratamento B (Tradicional): Participante implementa funcionalidade completa primeiro, depois escreve testes unitários.

8.5 Variáveis Dependentes

Número de defeitos: quantidade de falhas detectadas pelos testes de aceitação.
Tempo de conclusão: tempo total em minutos.
Cobertura de testes: percentual de código coberto pelos testes unitários.

8.6 Variáveis de Controle

Complexidade das tarefas (equivalente para todas).
Ambiente de desenvolvimento (mesma configuração).
Tempo disponível (igual para todos).
Especificação das tarefas (mesmas instruções).

8.7 Variáveis de Confusão

Experiência prévia com TDD.
Experiência geral em programação.
Familiaridade com o domínio das tarefas.
Essas serão coletadas via questionário e consideradas na análise.

9 Desenho Experimental

9.1 Tipo de Desenho

Desenho within-subjects (medidas repetidas) com contrabalanceamento. Cada participante executa tarefas em ambas as condições, servindo como seu próprio controle. Isso reduz variabilidade e aumenta poder estatístico.

9.2 Randomização

Participantes serão divididos aleatoriamente em dois grupos:
Grupo 1: Primeira tarefa com TDD, segunda com tradicional.
Grupo 2: Primeira tarefa com tradicional, segunda com TDD.
A randomização será por sorteio computadorizado.

9.3 Contrabalanceamento

O contrabalanceamento evita que efeitos de aprendizagem ou fadiga favoreçam uma técnica. Serão usadas duas tarefas diferentes de complexidade equivalente (Tarefa X e Tarefa Y).

9.4 Grupos e Sessões

Dois grupos de contrabalanceamento. Cada participante realiza duas sessões, uma com cada técnica. Experimento em um único dia com intervalo entre sessões.

10 População, Sujeitos e Amostragem

10.1 População-Alvo

Desenvolvedores de software que trabalham com C# e .NET Core em empresas brasileiras.

10.2 Critérios de Inclusão

Ser desenvolvedor da empresa onde o experimento será conduzido.
Ter pelo menos 6 meses de experiência com C# ou .NET.
Concordar voluntariamente em participar.
Estar disponível para todas as etapas do experimento.

10.3 Critérios de Exclusão

Estagiários com menos de 6 meses de experiência.
Profissionais que não utilizam C# regularmente.
Pessoas envolvidas na elaboração do experimento.

10.4 Tamanho da Amostra

Entre 4 e 10 participantes, divididos igualmente entre os grupos de contrabalanceamento. Esse número é adequado para o desenho pareado considerando estudos similares na literatura.

10.5 Método de Seleção

Seleção por conveniência através de convite aos desenvolvedores da empresa. Participação voluntária sem qualquer pressão.

10.6 Treinamento

Todos receberão treinamento de 30 minutos sobre TDD antes do experimento, incluindo explicação do ciclo Red-Green-Refactor e demonstração prática em C#. Isso garante conhecimento mínimo da técnica.

11 Instrumentação e Protocolo Operacional

11.1 Instrumentos de Coleta

Questionário de caracterização: dados sobre experiência e formação dos participantes.

Formulário de registro de tempo: planilha para anotar início e fim de cada tarefa.

Suíte de testes de aceitação: testes predefinidos para verificar corretude das implementações.

Ferramenta de cobertura: para medir percentual de código coberto (Coverlet ou dotCover).

Questionário pós-experimento: percepções sobre as técnicas usando escala Likert.

11.2 Materiais de Suporte

Guia rápido de TDD (resumo de uma página).

Especificação das tarefas com requisitos e exemplos.

Instruções do experimento passo a passo.

Projeto base .NET Core pré-configurado.

Slides de treinamento sobre TDD.

11.3 Procedimento Experimental

Recepção e apresentação do experimento (10 min).

Assinatura do termo de consentimento (5 min).

Questionário de caracterização (10 min).

Treinamento sobre TDD (30 min).

Distribuição aleatória nos grupos (5 min).

Primeira sessão de desenvolvimento (45 min).

Intervalo (15 min).

Segunda sessão de desenvolvimento (45 min).

Questionário pós-experimento (10 min).

Encerramento (5 min).

Tempo total: aproximadamente 3 horas.

11.4 Plano de Piloto

Piloto com 2 a 3 desenvolvedores para validar clareza das instruções, adequação do tempo e funcionamento dos instrumentos. Ajustes poderão ser feitos com base no feedback antes do experimento principal.

12 Plano de Análise de Dados

12.1 Estratégia Geral

Comparação das métricas entre TDD e tradicional considerando o desenho pareado. Estatísticas descritivas serão calculadas primeiro, seguidas de testes de hipótese.

12.2 Métodos Estatísticos

Estatísticas descritivas: média, mediana, desvio padrão.

Teste de normalidade: Shapiro-Wilk para escolher entre testes paramétricos e não paramétricos.

Teste t pareado: para dados com distribuição normal.

Teste de Wilcoxon: para dados sem distribuição normal.

Tamanho do efeito: d de Cohen ou r para avaliar magnitude das diferenças.

12.3 Dados Faltantes e Outliers

Participantes que não completarem ambas as tarefas serão excluídos da análise pareada.

12.4 Análise Qualitativa

Respostas abertas do questionário serão analisadas por categorização temática, identificando padrões nas percepções dos participantes.

13 Avaliação de Validade

13.1 Validade de Conclusão

Ameaças: baixo poder estatístico, violação de suposições dos testes.

Mitigação: desenho pareado, verificação de suposições, cálculo de tamanho do efeito.

13.2 Validade Interna

Ameaças: efeito de aprendizagem, fadiga, diferenças entre tarefas.

Mitigação: contrabalanceamento, intervalo entre sessões, validação de equivalência das tarefas no piloto.

13.3 Validade de Construto

Ameaças: métricas podem não capturar todos aspectos de qualidade e produtividade.

Mitigação: múltiplas métricas complementares, definições operacionais claras, dados qualitativos.

13.4 Validade Externa

Ameaças: amostra de uma única empresa, tarefas simplificadas.

Mitigação: caracterização detalhada do contexto, discussão explícita das limitações.

13.5 Resumo

Tipo	**Ameaça**	**Mitigação**
Conclusão	Baixo poder	Desenho pareado
Interna	Aprendizagem	Contrabalanceamento
Construto	Métricas limitadas	Múltiplas métricas
Externa	Contexto específico	Caracterização detalhada

Tabela 3: Ameaças à validade

14 Ética, Privacidade e Conformidade

14.1 Questões Éticas

Participação voluntária sem pressão.

Resultados não serão usados para avaliar desempenho profissional.

Experimento durante horário de trabalho com autorização da empresa.

Participantes receberão treinamento como benefício.

14.2 Consentimento Informado

Cada participante receberá informações sobre objetivos, procedimentos, tempo estimado, riscos e benefícios. O consentimento será registrado através do Termo de Consentimento Livre e Esclarecido.

14.3 Privacidade

Dados coletados: nome, e-mail, cargo, experiência, formação.

Proteção: cada participante receberá código identificador (P01, P02, etc.). Dados armazenados em local seguro. Nenhum dado individual será publicado de forma identificável.

Dados mantidos por 5 anos após conclusão do estudo.

14.4 Aprovações Necessárias

Autorização da empresa.

Aprovação do orientador acadêmico.

Verificação de necessidade de submissão ao Comitê de Ética da PUC Minas.

15 Recursos, Infraestrutura e Orçamento

15.1 Recursos Humanos

Gabriel Henrique Miranda Rodrigues: pesquisador, execução, coleta e análise.

Orientador: supervisão acadêmica e revisão metodológica.

Contato na empresa: facilitação de acesso e apoio logístico.

15.2 Infraestrutura Técnica

Computadores com Visual Studio ou VS Code.

.NET Core SDK.

Framework de testes (xUnit).

Ferramenta de cobertura de código.

Repositório Git.

Sala para os participantes.

Projetor para treinamento.

15.3 Materiais

Formulários (TCLE, questionários).

Projeto base configurado.

Especificações das tarefas.

Material de treinamento.

Suíte de testes de aceitação.

15.4 Orçamento

Custos mínimos: impressão de materiais (R$ 50 a R$ 100). Demais recursos já disponíveis na empresa e universidade.

16 Cronograma e Riscos Operacionais

16.1 Macrocronograma

Entrega	Prazo
Entrega 1: Identificação Básica, Contexto e Problema	21/11/2025
Entrega 2: Escopo, Objetivo, Stakeholders, Riscos de alto nível, Premissas e Critérios de Sucesso	25/11/2025
Entrega 3: Modelo conceitual e hipóteses; Variáveis, fatores, tratamentos e objetos de estudo; Desenho experimental	28/11/2025
Entrega 4: População, sujeitos e amostragem; Instrumentação e protocolo operacional; Plano de análise de dados	02/12/2025
Entrega 5: Avaliação de validade (ameaças e mitigação)	05/12/2025
Entrega Final: Apresentação e entrega final do planejamento de experimento (em Markdown)	12/12/2025

Tabela 4: Cronograma de Entregas da Disciplina

16.2 Dependências

Instrumentos dependem da aprovação do plano.

Piloto depende dos instrumentos e ambiente.

Execução depende de autorização e ajustes do piloto.

Análise depende da conclusão da execução.

16.3 Riscos e Contingências

Risco	Prob.	Contingência
Poucos participantes	Média	Ampliar convite
Problemas técnicos	Baixa	Testar previamente
Desistências	Média	Recrutar extras
Atrasos em aprovações	Média	Iniciar cedo

Tabela 5: Riscos operacionais

17 Governança do Experimento

17.1 Papéis e Responsabilidades

Pesquisador: decisões operacionais, execução, coleta, análise.

Orientador: revisão e aprovação do plano, supervisão metodológica.

Contato na empresa: facilitar acesso, providenciar infraestrutura.

17.2 Acompanhamento

Reunião de aprovação do plano com orientador.

Checkpoint de preparação uma semana antes do piloto.

Reunião pós-piloto para ajustes.

Confirmação final dois dias antes da execução.

17.3 Controle de Mudanças

Mudanças significativas no desenho ou protocolo requerem aprovação do orientador. Mudanças serão registradas no histórico de versões e comunicadas aos envolvidos.

18 Plano de Documentação e Reprodutibilidade

18.1 Repositórios

Estrutura de diretórios:

/experimento-tdd/

  /plano/

  /instrumentos/

  /materiais/

  /dados/

  /analise/

  /relatorio/

18.2 Templates

Termo de Consentimento.

Questionários de caracterização e pós-experimento.

Especificação de tarefas.

Formulário de registro de tempo.

Roteiro do pesquisador.

18.3 Pacote de Replicação

Versão final do plano.

Todos os instrumentos em formato editável.

Materiais de treinamento.

Projeto base e testes de aceitação.

Scripts de análise.

Dados anonimizados (se permitido).

Relatório com lições aprendidas.

19 Plano de Comunicação

19.1 Públicos e Mensagens

Orientador: progresso, decisões metodológicas, resultados.

Gestão da empresa: solicitação de autorização, cronograma, benefícios, resultados agregados.

Potenciais participantes: convite, objetivos, tempo necessário, garantias de anonimato.

Participantes confirmados: detalhes logísticos, data, local.

19.2 Canais e Frequência

E-mail: comunicações formais e documentação.

Reuniões: alinhamentos quinzenais com orientador.

Mensagens: comunicações rápidas com participantes.

19.3 Pontos Obrigatórios

Aprovação do plano pelo orientador.

Obtenção de autorização da empresa.

Confirmação da data de execução.

Mudanças significativas no cronograma ou protocolo.

Conclusão da coleta de dados.

20 Critérios de Prontidão para Execução

20.1 Checklist

20.2 Aprovações Finais

Orientador: confirmar adequação do plano e preparação do pesquisador.

Item	Status
Autorização da empresa	OK
Questionários prontos	OK
Especificações das tarefas finalizadas	OK
Projeto base configurado e testado	OK
Testes de aceitação validados	OK
Material de treinamento preparado	OK
Ambiente testado	OK
Piloto realizado e ajustado	OK
Mínimo de participantes confirmados	OK
Local reservado	OK
Participantes informados	OK

Tabela 6: Checklist de prontidão

Responsável na empresa: autorizar realização na data prevista e liberação dos participantes.

Referências

BECK, K. **Test-Driven Development: By Example**. Boston: Addison-Wesley, 2003.

ERDOGMUS, H.; MORISIO, M.; TORCHIANO, M. On the effectiveness of the test-first approach to programming. **IEEE Transactions on Software Engineering**, v. 31, n. 3, p. 226-237, 2005.

FUCCI, D. et al. A dissection of the test-driven development process: does it really matter to test-first or to test-last? **IEEE Transactions on Software Engineering**, v. 43, n. 7, p. 597-614, 2017.

MUNIR, H.; MOAYYED, M.; PETERSEN, K. Considering rigor and relevance when evaluating test driven development: A systematic review. **Information and Software Technology**, v. 56, n. 4, p. 375-394, 2014.

RAFIQUE, Y.; MISIC, V. B. The effects of test-driven development on external quality and productivity: A meta-analysis. **IEEE Transactions on Software Engineering**, v. 39, n. 6, p. 835-856, 2013.

WOHLIN, C. et al. **Experimentation in Software Engineering**. Berlin: Springer, 2012.