UNIVERSIDADE SÃO JUDAS TADEU

Bianca Alves Ribeiro - RA:822224026

Fabrício de Barros Narbon - RA:822227166

Gabriel Farah de Lima - RA:822231424

Luiz Gustavo França de Abreu - RA:823210075

Rafael Rossetto Guitarrari - RA:823158602

Webster Diogenes Rodrigues - RA:8222242764

# ATIVIDADE A3

# GESTÃO E QUALIDADE DE SOFTWARE

São Paulo - SP

2025

Bianca Alves Ribeiro

Fabrício de Barros Narbon

Gabriel Farah de Lima

Luiz Gustavo França de Abreu

Rafael Rossetto Guitarrari

Webster Diogenes Rodrigues

# ATIVIDADE A3

# GESTÃO E QUALIDADE DE SOFTWARE

Trabalho apresentado à Universidade São Judas Tadeu, no curso de graduação de ciências da computação.

Professores: Robson Calvetti e Edyene Cely

São Paulo - SP
2025

## SUMÁRIO

## 1. RESUMO

Este trabalho apresenta o planejamento de testes de software desenvolvido no contexto do projeto Inova Tech, cuja finalidade é otimizar o atendimento hospitalar por meio da automação de filas com base na gravidade dos sintomas dos pacientes. São abordados aspectos essenciais da engenharia de software, como o levantamento de requisitos, a definição de casos de teste, a alocação de recursos, os marcos do projeto, além da gestão e repositório de configuração. A aplicação foi estruturada com foco na eficiência do processo de triagem, considerando requisitos funcionais e não funcionais que asseguram a confiabilidade do sistema. A metodologia empregada baseia-se em práticas ágeis com sprints bem definidos, visando garantir qualidade, rastreabilidade e conformidade com os padrões de desenvolvimento. O planejamento detalhado e a documentação apresentada buscam o compromisso com a qualidade do software e a entrega de valor ao usuário final.

## 2. PLANEJAMENTO DE TESTES DE SOFTWARE

### 2.1. Cronograma de atividades

Cronograma apresentado em tabela:

- Quadro 1

| Atividade | Responsável | Sprint | Status |
|---|---|---|---|
| Levantamento de Requisitos | Rafael Rossetto Guitarrari | Sprint 4 a 5 | Planejado |
| Definição de Casos de Teste | Webster Diogenes Rodrigues | Sprint 4 a 5 | Planejado |
| Execução dos Testes | Bianca Alves Ribeiro | Sprint 4 a 5 | Planejado |
| Correção de Bugs | Gabriel Farah de Lima | Sprint 4 a 5 | Planejado |
| Testes Finais e Validação | Luiz Gustavo França de Abreu, Fabrício de Barros Narbon | Sprint 4 a 5 | Planejado |

O autor (2025)

## 2.2. Alocação de recursos

Tabela com alocação de recursos humanos:

- Quadro 2

| Nome | Função | Carga Horária | Responsabilidades |
|---|---|---|---|
| Rafael Rossetto Guitarrari, Fabrício de Barros Narbon | Desenvolvedor Pleno e Estagiário | 20h/semana | Levantamento e documentação de requisitos |
| Webster Diogenes Rodrigues | Arquiteto | 20h/semana | Definição e execução dos testes |
| Bianca Alves Ribeiro, Gabriel Farah de Lima | Desenvolvedor Junior | 20h/semana | Correção de bugs e suporte técnico |
| Luiz Gustavo França de Abreu | QA | 20h/semana | Correção visual e funcional |

O autor (2025)

## 2.3. Marcos do projeto

- Quadro 3

| Marco | Data Esperada | Descrição |
|---|---|---|
| **Finalização do Levantamento** | **05/04/2025** | **Requisitos documentados e aprovados** |
| **Conclusão dos Casos de Teste** | **10/04/2025** | **Casos revisados e validados** |
| **Primeira Rodada de Testes** | **11/03/2025** | **Testes executados e relatórios gerados** |
| **Correções Concluídas** | **30/04/2025** | **Bugs críticos corrigidos** |
| **Validação Final com Cliente** | **05/05/2025** | **Software validado para produção** |

O autor (2025)

# 3. PLANEJAMENTO DE TESTES DE SOFTWARE

## 3.1. Plano de projeto

### 3.1.1. Planejamento do projeto

O projeto intitulado *Inova Tech*[1] tem como objetivo o desenvolvimento de uma aplicação voltada à organização de filas hospitalares automatizando as fichas de atendimento de acordo com as variáveis que influenciam diretamente a gravidade da situação do paciente.

### 3.1.2. Escopo

O escopo deste projeto contempla:

- Cadastro da ficha do paciente;
- Gerenciamento da fila por prioridade a depender de sintomas e idade;
- Criação de senha única por paciente; Interface para visualização da fila.

### 3.1.3. Recursos

#### 3.1.3.1. Recursos humanos: 2 Desenvolvedores Júnior, 1 Desenvolvedor Pleno, 1 Arquiteto, 1 QA, e 1 Estagiário;

#### 3.1.3.2. Recursos tecnológicos:

- Linguagens: Java;
- Banco de dados: MySQL;
- Ferramentas: GitHub e NetBeans.

[1] O projeto Inova Tech é a iniciativa desenvolvida no ano de 2025 que serve como base para o presente trabalho, envolvendo atividades práticas relacionadas à engenharia de software e desenvolvimento de sistemas.

3.1.3.3. Infraestrutura: Computadores pessoais com acesso à internet e ferramentas instaladas.

#### 3.1.4. Estimativas de projeto

A estimativa de duração do projeto é de 5 Sprints:

- Sprint 1: Levantamento de Requisitos e Prototipação;
- Sprint 2-3: Desenvolvimento do Back-end;
- Sprint 3-4: Desenvolvimento do Front-end;
- Sprint 4: Integração e Testes Unitários;
- Sprint 5: Testes Funcionais, Correções e Documentação Final.

Horas previstas: Aproximadamente 250 totais distribuídos entre os membros.

Custo estimado: R$70.200,00.

Riscos identificados: Alterações frequentes de requisitos, indisponibilidade de recursos humanos, falhas em integrações.

## 3.2. Plano de testes

### 3.2.1. Introdução

A finalidade do plano de testes é descrever todo o fluxo de testes implementados, isto inclui objetivo, escopo, requisitos a serem testados, estratégias cronograma de atividades, entre outros.

### 3.2.2. Escopo

O objetivo do software é auxiliar no gerenciamento de pacientes em meio a um cenário de triagem hospitalar, utilizando algoritmos de ordenação para potencializar a performance da triagem. Fornecendo ao usuário um gerenciamento inteligente, o software se encarrega de calcular prioridade e gravidade, além de ordenar a fila para o atendimento.

### 3.2.3. Objetivo

O Objetivo do Plano de Teste de Iteração é reunir todas as informações necessárias ao planejamento e ao controle do esforço de teste referente a uma iteração específica. Ele descreve a abordagem dada ao teste do software e é o plano de nível superior gerado e usado pelos gerentes para coordenar o esforço de teste.

Este Plano de Teste referente ao *InovaTech* suporta os seguintes objetivos:

- Elucidar o testador quanto as ferramentas necessárias para a realização dos testes cabíveis;
- Identificar erros e assegurar suas correções tanto no sistema quanto na sua documentação;
- Identificar os requisitos para o testador realizar os devidos testes com a finalidade de melhorias;
- Mostrar inúmeras formas aos testadores para a utilização dos testes;
- Identificar os resultados dos testes
- Explicar os recursos necessários para a execução dos testes;
- Adquirir estimativa dos esforços de teste;
- Verificar possíveis riscos por meios dos testes.

#### 3.2.4. Requisitos a serem testado

I. Requisitos funcionais a serem testados com o Plano de Teste

- Cálculo da gravidade dos sintomas
- Gerenciamento de atendimento
- Senha única a cada paciente
- Registro de ficha médica
- Armazenar a ficha médica em um banco de dados persistente
- Editar ficha do paciente
- Cálculo de prioridade de atendimento
- Entrada única de paciente por CPF
- Exibir fila de pacientes ordenada
- Visualização resumida da ficha de cada usuário
- A fila deverá ser ordenada por um algoritmo de ordenação

II. Requisitos não funcionais a serem testados com o Plano de Teste

- Distinguir a prioridade de atendimento por cores
- Interface intuitiva para os usuários

#### 3.2.5. Estratégias, tipos de testes e ferramentas a serem utilizadas

A estratégia de testes adotada é baseada na combinação de testes automatizados e testes manuais, visando garantir a qualidade, a estabilidade e o correto funcionamento do sistema. O processo de testes está alinhado às práticas de integração contínua, permitindo que eventuais erros sejam identificados e corrigidos de forma ágil durante o ciclo de desenvolvimento.

Os testes serão organizados em diferentes níveis e categorias, incluindo desde testes de unidades isoladas até testes de funcionalidades completas e integradas.

#### 3.2.6. Níveis de teste

- Teste de unidade: o teste de unidade será aplicado para validar o funcionamento correto de unidades isoladas do código, como funções, métodos ou classes. O

objetivo é garantir que cada componente individual opere de acordo com o esperado, de forma independente dos demais módulos do sistema.

- Teste de Integração: O teste de integração será aplicado para garantir que os diferentes componentes do software se integrem de maneira eficaz e que as interações entre eles funcionem conforme o esperado.
- Teste de Sistema: O teste de sistema será executado para avaliar o software em sua totalidade, Documentação de um Produto de Software – Autora do Modelo: Profa. Dra. Ana Paula Gonçalves Serra 60 garantindo que todas as funcionalidades estejam integradas de forma adequada e que o sistema atenda aos requisitos definidos.
- Teste de Aceitação**:** O teste de aceitação será executado para verificar se o software atende aos requisitos e expectativas do cliente ou usuário final. Este teste tem como objetivo garantir que o sistema esteja pronto para uso, cumprindo os critérios de aceitação previamente definidos.
- Teste Funcional (Manual): Será realizado para validar, manualmente, o comportamento do sistema do ponto de vista do usuário, verificando se as funcionalidades atendem aos requisitos especificados.

### 3.2.7. Tipos de Teste:

- Teste de Funcionalidade: O teste de funcionalidade se concentrará em validar se o software atende aos requisitos funcionais estabelecidos. Isso inclui verificar se os recursos, como o cálculo de gravidade e prioridade, gerenciamento de atendimento, senha única, registro de ficha médica, editar ficha do paciente, entrada única de paciente por CPF, exibir fila de pacientes ordenada, visualização resumida da ficha médica e algoritmo de ordenação funcionam corretamente.
- Teste de Usabilidade: O teste de usabilidade será importante para avaliar a facilidade de uso do software e a experiência do usuário. Garante que o software

seja intuitivo e que os usuários possam interagir com ele sem a necessidade de um manual ou tutorial.

- Teste de Desempenho: O teste de desempenho será necessário para verificar o tempo de resposta do software.

A tabela a seguir apresenta itens e elementos do produto que foram identificados como alvo dos testes.

| **Tipo de Teste** | **Itens/Elementos a serem testados** |
|---|---|
| Teste de Banco de Dados | - Validar se ficha médica do paciente pode ser cadastrada, consultada e removida do sistema;<br>- Verificar se os valores atribuídos aos sintomas podem ser cadastrados, consultados e removidos do sistema. |
| Teste de Função | - Verificar se registros estão sendo realizados corretamente;<br>- Verificar se a entrada única com CPF está sendo executada com êxito;<br>- Verificar funcionalidade de edição da ficha médica;<br>- Verificar se o cálculo de prioridade e gravidade está sendo realizado corretamente;<br>- Verificar se todas as entradas de dados fornecidas são válidas;<br>- Validar se o gerenciamento de atendimento está coerente. |
| Testes de Interface do Usuário | - Verificar se as informações estão de forma clara e intuitiva para todos os clientes;<br>- Verificar se os ícones no menu estão funcionando e indicando qual o usuário está;<br>- Verificar as nomenclaturas e palavras estão de acordo com a língua portuguesa;<br>- Verificar se a fila está sendo exibida corretamente. |

#### 3.2.8. Ferramentas Utilizadas:

- **JUnit:**

Framework utilizado para criação e execução de testes unitários, validando métodos e funções da lógica de negócio.

- **Mockito:**

Ferramenta para criação de mocks de objetos e simulação de comportamentos em testes unitários, permitindo isolar componentes durante a validação.

- **H2 Database:**

Banco de dados em memória utilizado durante os testes de integração, evitando impacto no banco de dados real e permitindo testes rápidos e seguros.

- **Teste Manual:**

Realizado diretamente na interface do software (.exe), validando os fluxos operacionais de forma interativa, como um usuário final utilizaria.

### 3.3. Recursos a serem empregados

#### 3.3.1. Equipe**:**

3 Desenvolvedores: Sendo 2 de nível Júnior que serão responsáveis por apoiar o desenvolvimento, criar testes unitários e desenvolver o frontEnd.

1 QA de nível Júnior: Responsável pela execução de testes manuais, automação e verificação de defeitos.

1 Estagiário: Responsável em apoiar no desenvolvimento, testes e documentações

1 Arquiteto de Soluções nível Pleno: Responsável por desenvolver toda a arquitetura do software, isso inclui: Infraestrutura, relacionamento de dados, aplicações, ferramentas, framework e linguagem a ser utilizada.

E por fim, 1 desenvolver de nível Pleno que será responsável por criar a lógica do gerenciamento de atendimento e a parte mais complexa do desenvolvimento como o

algoritmo de ordenação e interação com o banco de dados. Também será responsável por apoiar na arquitetura do software

#### 3.3.2. Equipamentos:

4 Notebooks (AMD Ryzen7-5700U, 16GB RAM, SSD 512GB)

2 Notebooks (AMD Ryzen 5-5625U, 8GB RAM, HD 512GB)

#### 3.3.3. Infraestrutura:

Banco de dados H2 (em memória): Utilizado exclusivamente durante o processo de testes automatizados, principalmente para testes de integração, evitando impacto no banco de dados real e garantindo testes mais rápidos e isolados.

Banco de dados PostgreSQL (produção): Utilizado para armazenar os dados reais da aplicação em ambiente de produção. Será o banco de dados principal do sistema, garantindo persistência confiável e suporte a queries complexas.

#### 3.3.4. Ferramentas e softwares:

Ferramentas de Teste: JUnit, Mockito e H2.
Versionamento: Git

### 3.4. Cronograma das atividades

O cronograma de atividades foi elaborado com base na metodologia ágil, utilizando o conceito de *sprints,* quinzenais para organizar e distribuir as tarefas de teste

ao longo do tempo. Essa abordagem favorece rápida identificação de falhas e facilita o acompanhamento contínuo da evolução do processo de testes.

A tabela a seguir apresenta a distribuição das atividades por sprint e seus respectivos responsáveis:

| ATIVIDADE | RESPONSÁVEL | PERÍODO |
|---|---|---|
| Elaboração dos casos de teste | QA e Desenvolvedores | Sprint 4 |
| Execução de testes unitários | Desenvolvedores | Sprint 4 |
| Execução de testes de integração | Desenvolvedores | Sprint 4 |
| Execução de testes manuais (funcionais) | QA | Sprint 5 |
| Registro e correção de defeitos | TODOS | Sprint 5 |
| Validação final e homologação | QA e Arquiteto | Sprint 5 |

### 3.5. Definição dos marcos do projeto

Os marcos do projeto representam eventos significativos que indicam o progresso das etapas de teste e auxiliam no acompanhamento dos prazos e da qualidade do processo. Esses marcos são pontos de verificação importantes para garantir que cada fase tenha sido concluída de forma satisfatória antes do avanço para as próximas etapas.

A seguir, são apresentados os principais marcos definidos para o processo de testes deste projeto, juntamente com o sprint em que estão previstos:

- Finalização da escrita dos casos de teste (Sprint 4): Marca a conclusão da elaboração dos cenários de teste, baseados nos requisitos definidos. Essa entrega permitirá o início imediato da execução dos testes.
- Conclusão dos testes unitários (Sprint 4): Indica que todos os testes de unidade foram implementados e executados com sucesso, validando o comportamento de componentes individuais do sistema.
- Conclusão dos testes de integração com banco de dados (Sprint 4): Representa a verificação da comunicação entre os componentes principais da aplicação e o banco de dados, assegurando que o fluxo de dados ocorra conforme o esperado.
- Início dos testes manuais funcionais (Sprint 5): Marca o começo da execução dos testes funcionais realizados pelo QA, voltados à verificação do sistema como um todo, com foco na experiência do usuário e nos fluxos de uso reais.
- Correção dos defeitos críticos identificados (Sprint 5): Refere-se à fase em que os principais erros detectados durante os testes serão analisados, classificados e corrigidos pela equipe de desenvolvimento.
- Homologação do sistema e liberação para produção (Sprint 5): Último marco, que representa a validação final do sistema em ambiente de homologação. Após a aprovação, o sistema será considerado pronto para ser implantado em produção.

### 3.6. Caso de teste

Os casos de teste representam cenários específicos validados com o objetivo de garantir que o sistema atenda aos requisitos definidos, tanto funcionais quanto não funcionais. Esta etapa visa assegurar a qualidade do software por meio da execução sistemática de entradas, ações e verificações esperadas, permitindo detectar falhas, inconsistências ou comportamentos inesperados ainda durante o processo de desenvolvimento.

| ID | Descrição do Caso de Teste | Entrada | Resultado Esperado |
|---|---|---|---|
| CT-01 | Testar registro de ficha médica | Nome, Data de Nascimento, CPF, Nível de Dor, Temperatura, Sexo, Gestante, Lista de sintomas | Cadastro salvo com sucesso |
| CT-02 | Testar tentativa de registro com CPF duplicado | CPF já existente | Mensagem de erro informando duplicidade |
| CT-03 | Testar classificação emergencial | Qualquer sintoma com gravidade definida como gravíssimo ou conjunto de sintomas que resultem em 1000 ou mais pts | Status do paciente igual a “EMERGENCIA” |
| CT-04 | Testar classificação muito urgente | Qualquer sintoma com gravidade a partir de grave ou um conjunto de sintomas que totalizem score de 400 até 999 | Status do paciente igual a “MUITO_URGENTE” |
| CT-05 | Testar classificação urgente | Qualquer sintoma com gravidade a partir de normal ou um conjunto de sintomas que totalizem score de 100 até 399 | Status do paciente igual a “URGENTE” |
| CT-06 | Testar classificação pouco urgente | Qualquer sintoma com gravidade a partir de leve ou um conjunto de sintomas que totalizem score de 20 até 99 | Status do paciente igual a “POUCO URGENTE” |

| | | | |
|---|---|---|---|
| CT-07 | Testar classificação não urgente | Qualquer sintoma com gravidade a partir de muito leve ou um conjunto de sintomas que totalizem score de 7 até 19 | Status do paciente igual a “NÃO URGENTE” |
| CT-08 | Testar ordenação de fila 5 pacientes com status diferentes | Status: amarelo, azul, vermelho, verde e laranja | Ordem correta: vermelho > laranja > amarelo > verde > azul |
| CT-09 | Testar ordenação de fila 3 pacientes com status iguais a emergente | 1. Paciente 1: score = 1000<br>2. Paciente 2: score = 2500<br>3. Paciente 3: score = 1600 | Ordem de atendimento correta: Paciente 2 > Paciente 3 > Paciente 1 |
| CT-10 | Testar persistência no banco de dados | Dados completos | Dados persistidos e recuperados corretamente |
| CT-11 | Testar comboBox e checkBox | Sintomas cadastrados | Funcionalidade e retorno dos sintomas corretos |
| CT-12 | Testar edição da ficha de cada paciente | Uma ficha de um paciente | Edição realizada com sucesso |
| CT-13 | Testar erro ao salvar ficha com campos obrigatórios vazios | Nome, CPF, data de nascimento, gravida, sexo e nível da dor ausentes | Mensagem de erro informando campo obrigatório |

| | | | |
|---|---|---|---|
| CT-14 | Testar erro ao remover campos obrigatórios na edição de ficha | Remover nome, CPF, data de nascimento, gravida, sexo ou nível da dor | Mensagem de erro informando campo obrigatório |
| CT-15 | Testar usabilidade da interface na abertura da ficha médica | Abrir formulário de ficha médica | Campos dispostos de forma intuitiva e de fácil preenchimento |
| CT-16 | Testar navegação entre telas | Usuário acessa diferentes seções do sistema | Navegação fluida e sem erros de redirecionamento |
| CT-17 | Testar campo CPF com caracteres inválidos | "ABC12345678" | Mensagem de erro de formatação |
| CT-18 | Testar tempo de resposta para classificação de paciente | Cadastro de ficha com 10 sintomas | Processamento concluído em menos de 1 segundo |

### 3.7. Roteiro de Testes

O roteiro de testes tem como objetivo organizar e detalhar a execução dos casos de teste previamente definidos, proporcionando uma visão clara das ações a serem realizadas. Cada caso de teste será executado conforme sua natureza (funcional, validação de regras de negócio, usabilidade ou desempenho), levando em conta o fluxo natural do sistema

A tabela abaixo resume a sequência e a priorização da execução:

| **ID do Caso** | **Objetivo Principal** | **Tipo de Teste** | **Responsável** | **Observação** |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| CT-01 | Cadastro inicial de ficha médica | Funcional | QA | Verifica inserção de dados válidos |
| CT-13 | Validação de campos obrigatórios | Validação | QA | Cenário negativo de cadastro |
| CT-02 | CPF duplicado | Validação | QA | Garante unicidade |
| CT-03 – CT-07 | Classificação por gravidade | Regras de negócio | Dev | Teste unitário por faixa de score |
| CT-08 – CT-09 | Ordenação da fila de atendimento | Regras de negócio | Dev | Testes de lógica de ordenação |
| CT-17 | Validação de CPF inválido | Validação | QA | Teste de formatação e restrição |
| CT-10 | Persistência no banco de dados | Funcional | QA | Teste de integração com BD |
| CT-11 | Verificação de checkBox e comboBox | Funcional | QA | Teste de interface e retorno |
| CT-12 | Edição de ficha | Funcional | QA | Inclui cenários positivos e negativos |
| CT-14 | Campos obrigatórios na edição | Validação | QA | Similar ao CT-13 no fluxo de edição |
| CT-15 | Usabilidade do formulário | Usabilidade | QA | Avaliação subjetiva e intuitividade |
| CT-16 | Navegação entre telas | Funcional | QA | Garantia de fluxo correto |

| | | | | |
|---|---|---|---|---|
| CT-18 | Desempenho da classificação | Desempenho | QA | Medição de tempo de resposta |

## 4. GESTÃO DE CONFIGURAÇÃO DE SOFTWARE

A gestão de configuração de software (GCS) tem como objetivo controlar e registrar as mudanças realizadas durante o desenvolvimento do sistema *InovaTech*, garantindo a integridade e a rastreabilidade de todos os artefatos produzidos.

O processo de GCS da InovaTech:

- Controle de Versões: Todas as alterações no código-fonte, documentos e ativos relacionados ao projeto são controladas por meio de um sistema de versionamento (Git).

- Identificação de Configuração: Cada item de configuração — como arquivos de código, documentação técnica e arquivos de testes — possui um identificador único e é devidamente descrito no controle de versão.

- Auditorias e Revisões: São realizadas revisões a cada final de sprint, garantindo que as versões estejam coerentes com os requisitos e as metas estabelecidas.

- Gerência de Mudanças: As solicitações de mudança são documentadas e avaliadas por toda a equipe. Somente mudanças aprovadas são implementadas, reduzindo riscos de inconsistências.

- Baseline de Entregas: Cada sprint concluída é considerada um ponto de verificação (baseline), sendo armazenada de forma segura no repositório remoto para eventual retorno ou recuperação.

## 5. REPOSITÓRIO DE GESTÃO DE CONFIGURAÇÃO DE SOFTWARE

O repositório de configuração do projeto *InovaTech* está hospedado no *GitHub*, acessível apenas à equipe de desenvolvimento mediante autenticação.

**Plataforma utilizada:**

- *GitHub* (https://github.com/Webster-Rodrigues/HospitalScreening).

## 6. BIBLIOGRAFIA

GONÇALVES, Priscila de Fátima; BARRETO, Jeanine dos Santos; ZENKER, Aline Maciel; FAGUNDES, Rubem. Testes de software e gerência de configuração. Soluções Educacionais Integradas, 2019. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595029361/cfi/1!/4/4@0.00:56.3. Acesso em: 10 mai. 2025.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de software: uma abordagem profissional. 8. ed. Porto Alegre: Bookman, 2016. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788580555349/cfi/3!/4/2@100:0.00. Acesso em: 10 mai. 2025.

SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011. Disponível em: https://bv4.digitalpages.com.br/?term=engenharia%2520de%2520software&searchpage=1&filtro=todos&from=busca&page=_14&section=0#/legacy/276. Acesso em: 10 mai. 2025.