Processo de Teste de Software

Gerência de Projetos e Sistemas
Superintendência Central de Tecnologia da Informação
Secretaria de Estado de Desenvolvimento Econômico e Inovação.

Histórico

Versão	Data	Alteração no Documento	Autor
0.1	24/04/2019	Elaboração do Documento	Gilmar F. Arantes
0.2	26/04/2019	Revisão da Versão 0.1	Paulo Marcos S. Rodrigues

Sumário

1 Introdução

Este documento apresenta o Processo de Teste de Software (PTTS) da Gerência de Projetos e Sistemas (GPS) da Superintendência Central de Tecnologia da Informação (SCTI) da Secretaria de Desenvolvimento Econômico e Inovação (SEDI) do Governo do Estado de Goiás.

Este Processo foi desenvolvido pela Equipe de Qualidade da GPS, que é responsável pela garantia da qualidade dos produtos de software desenvolvidos por esta gerência.

O PTTS foi concebido para ser um guia de teste ágil, o que significa ser aderente às práticas ágeis, sobretudo às da Metodologia Scrum[1], nominada somente como SCRUM no restante deste documento. Esta decisão foi tomada em virtude do atual quadro de:

- Reestruturação da Superintendência, da Gerência.
- Readaptação do Processo de Desenvolvimento e Manutenção de Software da GPS (PDMS), para uma maior aderência à SCRUM.
- Crescimento das demandas à GPS em contraste à recorrente redução de recursos, sobretudo de recursos humanos.
- Necessidade de diminuir do tempo de entrega e finalmente
- Aumentar a qualidade dos produtos entregues.

As atividades do PTTS estão distribuídas em quatro etapas, conforme apresentado na Figura 1 e serão executadas em cada um dos eventos da SCRUM: *Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective*. A responsabilidade pela execução destas atividades é de toda a equipe, considerando que na SCRUM não existe o papel de testador, este papel é assumido por toda a equipe, que possui um compromisso com a qualidade do produto a ser entregue ao cliente.

1https://www.guru99.com/scrum-testing-beginner-guide.html

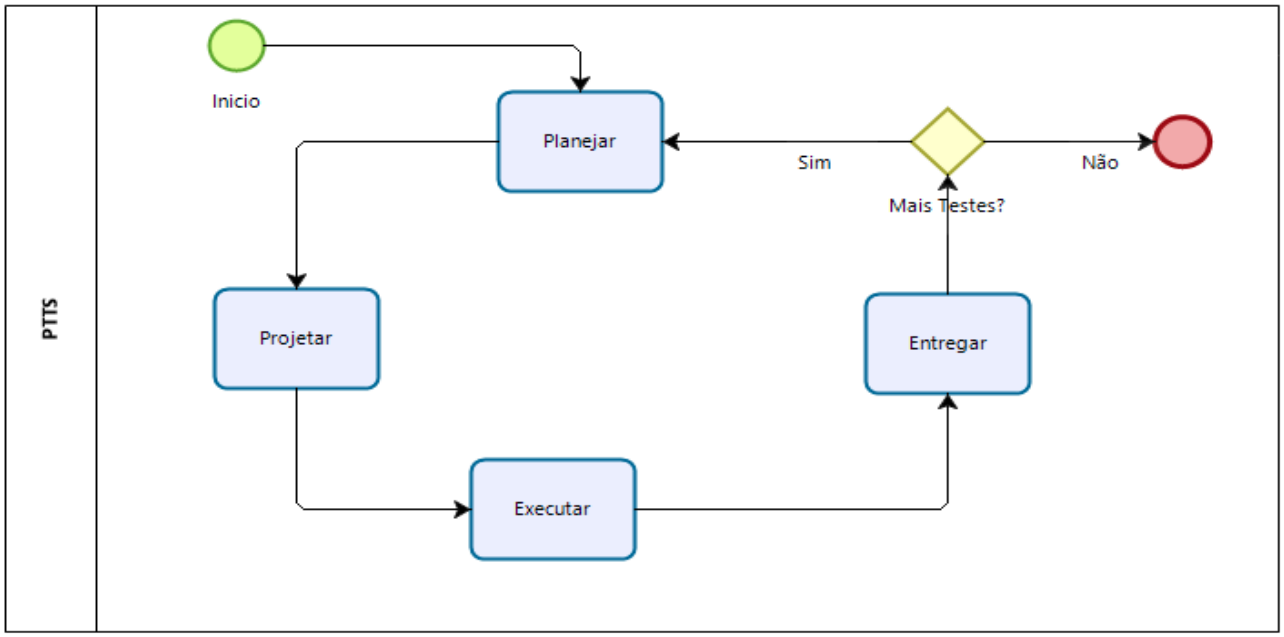

Figura 1: Etapas do PTTS.

1.1 Papéis

É importante a descrição dos papéis da SCRUM neste processo, tendo em vista que a equipe responsável tanto pelo projeto de desenvolvimento, quanto pelo projeto de teste é a mesma *Scrum Team*. A descrição dos papéis e suas respectivas responsabilidades são descritas a seguir:

- **Product Owner** – representa o conhecedor do produto e das necessidades do usuário final. Neste processo tem a função de conhecer e defender os interesses do cliente para que o produto seja entregue com a qualidade esperada. O Product Owner é o responsável pela definição dos critérios de aceitação.
- **Scrum Master –** É o responsável pelo andamento de cada projeto de teste desenvolvido a partir deste processo, resolvendo as dificuldades que surgirem;
- **Time Scrum –** São responsáveis por planejar e aplicar os testes no sistema. Quando os mesmos encontrarem ocorrências durante a análise, eles registraram e apresentaram os resultados no final das atividades. Estas atividades estarão sob coordenação de, pelo menos, um profissional da área de qualidade/teste, que fará parte da *Scrum Team*. Este profissional será o responsável por manter acesa a chama da qualidade entre os componentes da equipe.

Vale ressaltar que ao longo deste processo vão aparecer alguns papéis diferentes daqueles definidos pela SCRUM. Isto será particular às atividades de manutenção em sistemas legados, ou seja, aqueles que não foram desenvolvidos antes da implantação do atual Processo de Desenvolvimento e Manutenção de Software (PDMS). E, estes sistemas possuem equipes fixas, responsáveis pela sua manutenção.

1.2 Ambiente de Testes

Este processo foi elaborado levando em consideração a infraestrutura disponível para testes na GPS, conforme descrito a seguir:

1. A Ferramenta para documentação dos artefatos e gerenciamento de projetos de teste disponível na GPS é o *Testlink*.
2. A Ferramenta para o gerenciamento do fluxo de demandas é o *Redmine*.
3. A Ferramenta para o gerenciamento do controle de versão de artefatos de teste, sobretudo em relação a artefatos de automatização de testes, é o *GitLab*.
4. O *framework* para automatização de teste utilizado pela GPS é o *Selenium Webdriver*.
5. As aplicações em teste estarão disponíveis em ambiente próprio para teste, em contextos específicos para cada aplicação. O Acesso a este ambiente de teste se dá pelo Portal de Teste.
6. Os dados necessários para o teste deverão estar carregados no banco de dados de teste, acessado pelas aplicações disponíveis nos seus respectivos ambientes de teste.

2 Planejamento

A primeira etapa do PTTS é o planejamento do teste. O planejamento dos testes no modelo ágil é cíclico, sendo concebido em alto nível e refinado à medida da evolução do projeto. As atividades que compõem esta etapa do PTTS serão detalhas nas próximas Subseções. As Atividades constantes desta etapa estão apresentadas visualmente na Figura 2, e descritas em detalhes a partir da Subseção 2.1.

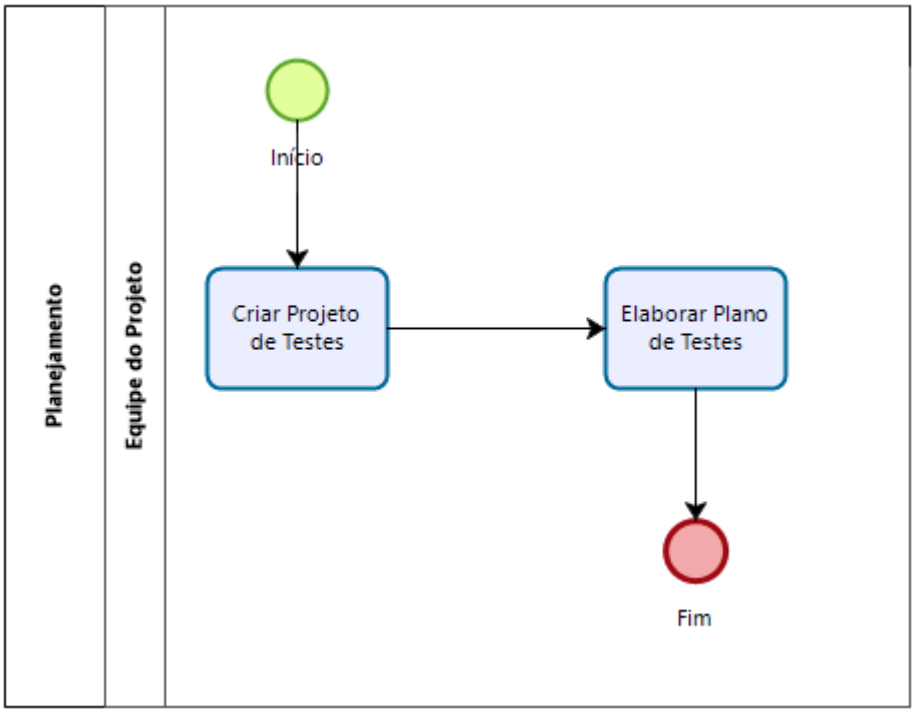

Figura 2 – Etapa de Planejamento

2.1 Projeto de Teste

A primeira das atividades de planejamento de teste é a criação/atualização de um projeto de teste. Um projeto de testes é único para um software, sendo atualizado a cada iteração de desenvolvimento ou a cada manutenção empreendida sobre o mesmo.

2.1.1 O Que:

O projeto de teste é a reunião do conjunto de artefatos em que são registradas todas as atividades relativas ao teste de um determinado software.

2.1.2 Quem:

1. **Para um Novo Projeto de Desenvolvimento**, o responsável pela criação deste Projeto de Teste é o profissional da equipe de qualidade participante da *Scrum Team*.

2. **Para um Novo Projeto de Manutenção**[2], o responsável é o analista a quem a demanda de manutenção foi atribuída.

É importante observar que as responsabilidades do analista em relação às atividades de teste, relativamente aos projetos de manutenção, podem ser compartilhadas entre os demais componentes da equipe da qualidade, de forma a compartilhar o contingente de demandas, ao mesmo tempo em que se dissemina o conhecimento a respeito do software.

2.1.3 Quando:

1. **Para um Novo Projeto de Desenvolvimento –** Após a elaboração, pelo *Scrum Master*, do *Product Backlog*, que ocorre logo após a reunião de *kick-off* do projeto.
2. **Para um Novo Projeto de Manutenção** – Caso o projeto de teste para o software em manutenção não exista, deverá ser criado imediatamente após o analista especificar a solução proposta ao problema apresentado na demanda por manutenção no software.

2.1.4 Como:

1. **Relatar atividade no Redmine.**

   1.1. **Para um Novo Projeto de Desenvolvimento** – O *Scrum Master* deverá relatar uma atividade no *Redmine*, solicitando a criação do Projeto de Testes e atribuí-la ao responsável pela sua execução.

   1.2. **Para um Novo Projeto de Manutenção**, o analista responsável pelo seu atendimento deverá relatar esta atividade e atribuir a si mesmo ou a algum dos componentes da equipe de qualidade que esteja disponível para execução desta atividade.

2. O Projeto de Teste é criado com a utilização da ferramenta *Testlink*, cujo manual de utilização está disponível no servidor de controle de versão, no seguinte link.

2.1.5 Prazo:

Esta atividade deverá ter o prazo máximo de 4 horas para sua elaboração.

2.1.6 Entradas:

1. Atividade relatada no *Redmine*;
2. *Product Backlog* (Para novo Projeto).
3. Especificação da Manutenção (Para projeto de manutenção).

2.1.7 Saídas:

1. Projeto de Testes devidamente criado no *Testlink*.

2 O termo Manutenção, no escopo do PTTS, refere-se a qualquer tipo de intervenção no sistema após sua entrega no ambiente de produção. Como por exemplo: correção, alteração, adição de nova funcionalidade, etc.

2. Atividade do *Redmine* devidamente encerrada.

2.1.8 Fluxo:

Segue para a atividade de Elaboração do Plano de Testes.

2.2 Plano de Testes

Independentemente de se tratar de um **Novo Projeto** ou do **Projeto de uma Manutenção**, o Plano de Testes deverá ser criado. No primeiro caso, para cada *Sprint*, no segundo, para a manutenção, propriamente dita.

2.2.1 O que:

Diferentemente dos modelos tradicionais, em um modelo ágil, um plano de teste é escrito e atualizado a cada iteração (*Sprint*), de forma que não existe um plano de testes para o sistema, como um todo.

Este plano de teste ágil inclui os tipos de testes que devem ser executados, considerando o escopo da *Sprint*, como por exemplo: requisitos de dados de teste, infraestrutura, resultados dos testes, etc. Um Típico plano de testes ágil inclui:

- Escopo do teste.
- Novas funcionalidades que serão testadas.
- Níveis e tipos de testes definidos a partir da complexidade dos requisitos da *Sprint*.
- Definição de quais testes serão automatizados e quais serão executados manualmente.
- Necessidade de execução dos testes da *Sprint* anterior.
- Tempo para execução dos testes da *Sprint*.
- Testes de carga e performance (Opcional, pois o software pode não demandar estes tipos de teste).
- Considerações acerca da infraestrutura necessária para a execução dos testes.
- Plano de mitigação de riscos.

2.2.2 Quem:

1. **Para um Novo Projeto de Desenvolvimento,** a responsabilidade pela elaboração do Plano de Testes é de toda a equipe do projeto. Sendo eleito um para a documentação do mesmo. Este eleito será escolhido pelo *Scrum Master*, preferencialmente o que estiver com menor *backlog* pessoal.
2. **Para um Novo Projeto de Manutenção**, esta responsabilidade é do analista a quem a demanda de manutenção foi atribuída.

2.2.3 Quando:

1. **Para um Novo Projeto de Desenvolvimento**, a criação do Plano de Teste ocorre logo após a *Sprint Planning*, quando já se conhece o escopo da *Sprint*.
2. **Para um Novo Projeto de Manutenção**, imediatamente após a criação/atualização do projeto de testes da demanda.

2.2.4 Como:

1. **Para um Novo Projeto de Desenvolvimento**, um dos componentes da *Scrum Team*, definido durante a *Sprint Planning*, deverá relatar uma atividade no *Redmine*, relativamente à elaboração do Plano de Testes da *Sprint*.

   1.1. Um novo Plano de Testes será criado no *Testlink*, identificando a qual *Sprint* se refere e quais os cenários de testes farão parte do escopo do plano.

2. **Para um Novo Projeto de Manutenção**, o analista deverá:

   2.1. Se a manutenção é recorrente e se já existe um plano de testes para a mesma.

      2.1.1. Se sim, atualizar o plano relativamente ao novo trabalho de manutenção efetuado.

   2.2. Se não, criar um plano de teste para validar o trabalho de manutenção efetuado.

2.2.5 Prazo:

Esta atividade deverá ter duração de no máximo 8 (oito) horas.

2.2.6 Entradas:

1. Atividade relatada no *Redmine*.
2. *Sprint backlog* (Novo Projeto).
3. Especificação da Manutenção (Projeto de Manutenção).

2.2.7 Saídas:

1. Plano de Testes criado/atualizado no *Testlink*.
2. Atividade do *Redmine* devidamente encerrada.

2.2.8 Fluxo:

Segue-se para a etapa de Projetar os Testes.

3 Projeto

Esta etapa do projeto de teste contempla as atividades necessárias para a criação do conjunto de casos de teste necessários para a cobertura dos cenários identificados e descritos no Plano de Testes de cada *Sprint* de um novo projeto de desenvolvimento ou Para um Novo Projeto de Desenvolvimento de manutenção, conforme pode se observar na Figura 3. Além da elaboração dos dados de teste, definição de roteiros para execução dos testes e implementação de *scripts* de teste, quando a forma de execução for automatizada. O detalhamento das atividades componentes desta etapa pode ser encontrado a partir da Seção 3.1.

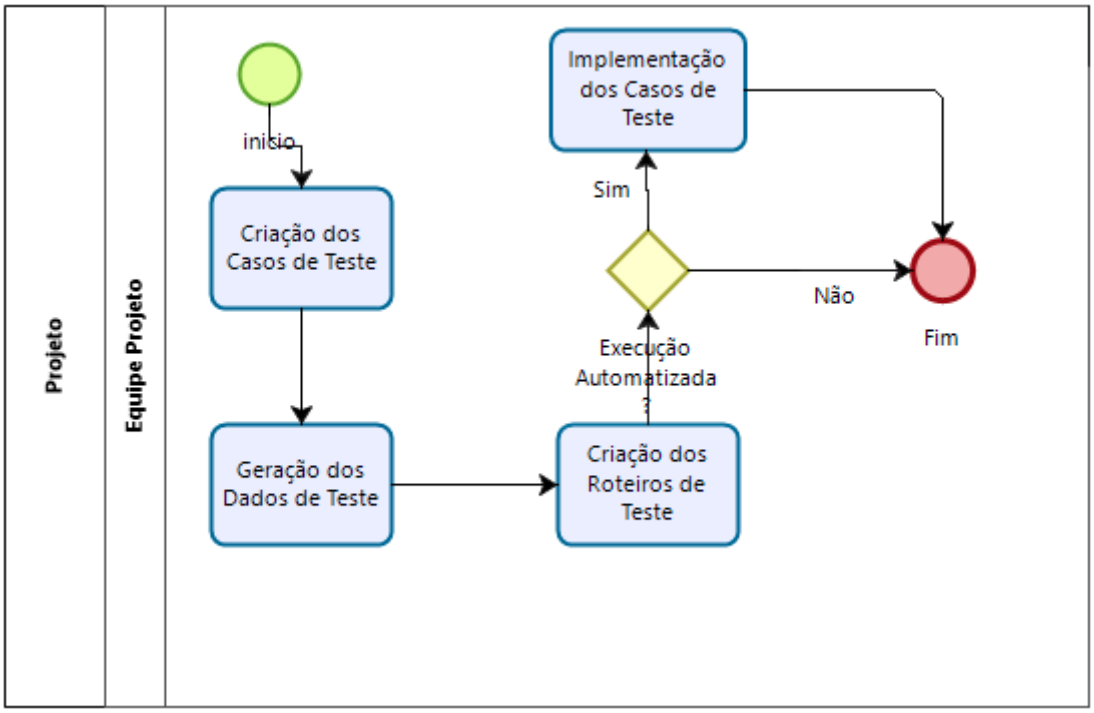

Figura 3 – Etapa de Projeto

3.1 Criação dos Casos de Testes

Os casos de teste são criados para a cobertura dos cenários de testes descritos no Plano de Teste, independentemente se é um novo projeto de desenvolvimento, ou um projeto de manutenção de um software existente. Estes cenários são relativos às funcionalidades que o sistema deve implementar. Estas funcionalidades são especificadas na forma de estórias de usuário, conforme definido no PDMS.

3.1.1 O Que:

Esta atividade objetiva a criação dos casos de testes suficientes para a cobertura dos cenários descritos no Plano de Teste da *Sprint* do novo projeto ou do projeto de manutenção de um software existente.

3.1.2 Quem:

1. **Para um Novo Projeto de Desenvolvimento**, o Responsável pela criação dos casos de testes é um dos componentes da *Scrum Team*, eleito para esta finalidade na *Sprint Planning*. Salvo em situação muito específica, mas o elaborador dos casos de testes não deve ser a mesma pessoa que redigiu as estórias de usuário.
2. **Para um Novo Projeto de Manutenção**, o responsável é o analista a quem a demanda da manutenção foi atribuída.

3.1.3 Quando:

1. **Para um Novo Projeto de Desenvolvimento**, sempre que uma nova estória de usuário estiver disponível no repositório do controle de versão da documentação do projeto, no *GitLab*.
2. **Para um Novo Projeto de Manutenção**, a partir do momento em que o analista a especificação da manutenção no repositório de controle de versão do projeto.

3.1.4 Como:

1. **Para um Novo Projeto de Desenvolvimento**, quando o responsável por redigir a estória de usuário a disponibilizar no repositório do controle de versão, ele relata duas atividades no *Redmine* e as atribui ao responsável pela elaboração dos casos de testes, sendo:

   1.1. A primeira atividade solicitando a revisão do documento de especificação da estória de usuário e

   1.2. A segunda solicitando a criação dos casos de testes suficientes para o teste da estória.

2. O responsável pela criação dos casos de teste revisa o documento de especificação de estória de usuário. Se encontrar alguma não conformidade, notifica o responsável pela sua edição para providenciar a adequação. Se não encontrar nenhuma não conformidade, passa para a criação dos casos de teste.
3. **Para um Novo Projeto de Manutenção**, os passos são os mesmos, apenas altera-se o artefato. Em vez de estória de usuário será a especificação da manutenção.

Os casos de testes são criados utilizando a ferramenta Testlink. E devem conter:

1. Um Identificador único;
2. Uma descrição;
3. Valores de entrada;
4. Passo a passo para sua execução;

5. Resultados esperados
6. Forma de execução (manual ou automatizada).

3.1.5 Prazo:

O tempo de execução desta atividade, para cada estória de usuário/especificação de manutenção, não pode ser superior a 8 (oito) horas.

3.1.6 Entradas:

1. Atividades relatada no Redmine;
2. Especificação de estória de usuário (Novo Projeto).
3. Especificação da manutenção (Projeto de Manutenção).

3.1.7 Saídas:

Conjunto de casos de teste devidamente criados e documentos na ferramenta *Testlink*.

3.1.8 Fluxo:

Segue-se para a atividade de criação dos dados de teste.

3.2 Geração dos Dados de Testes

A geração dos dados de teste é uma atividade cuja forma de execução não se altera independentemente se é para o teste de uma nova Sprint de um novo projeto ou Para um Novo Projeto de Desenvolvimento de manutenção de software.

3.2.1 O Que:

Esta atividade diz respeito à geração dos dados de teste que serão utilizados como valores de entradas pelos casos de testes, descritos na Seção 3.1, deste documento.

3.2.2 Quem:

O responsável por esta atividade é a pessoa componente da *Scrum Team*, que exerce a função de analista de dados do projeto. No caso de um novo projeto, esta pessoa deve estar identificada na *Sprint Planning*. No caso de um projeto de manutenção, esta atividade é atribuída ao AD (administrador de dados) do projeto.

3.2.3 Quando:

Esta atividade deve ser executada em paralelo à Atividade 3.1.

3.2.4 Como:

Esta Atividade é efetuada na forma de elaboração de scripts SQL para a obtenção de dados junto ao banco de teste.

3.2.5 Prazo:

Esta Atividade deverá ter a mesma duração da Atividade 3.1.

3.2.6 Entradas:

Cenários de Teste, cujos dados para cobertura necessitam ser gerados.

3.2.7 Saídas:

Scripts para a geração dos dados de testes elaborados e disponibilizados no repositório do controle de versão dos artefatos de testes de projeto.

3.2.8 Fluxo:

Segue-se para a atividade 3.3.

3.3 Criação dos Roteiros de Testes

3.3.1 O Que:

Esta tarefa consiste na elaboração da ordem de execução dos testes da *Sprint*/manutenção, além da definição de configurações, cargas de dados como pré-requisitos para a execução dos testes.

3.3.2 Quem:

Membro da *Scrum Team* ou equipe do projeto, responsável pela elaboração dos casos de teste.

3.3.3 Quando:

À medida que o conjunto de casos de teste, com seus respectivos dados de teste, para casa estória de usuário ou manutenção estejam concluídos.

3.3.4 Como:

Definido a prioridade e as pré-condições dos casos de teste na Ferramenta *Testlink*.

3.3.5 Prazo

O tempo de duração desta atividade não pode ser superior a 2 (duas) horas.

3.3.6 Entradas:

Conjunto de casos de teste para cada estória de usuário / manutenção.

3.3.7 Saídas:

Conjunto de casos de teste priorizados e com sua pré-condições definidas.

3.3.8 Fluxo:

Se a forma de execução dos casos de teste possui alguma automatização, segue-se para o fluxo 3.4. Se não, segue-se para o fluxo 3.5.

3.4 Implementação dos Casos de Testes

3.4.1 O Que:

Implementar os casos de testes é uma atividade executada somente quando a forma de execução de um determinado caso de teste foi definida como automática. E, consiste na conversão da especificação do caso de teste em código executável, escrito utilizando a mesma linguagem de programação da implementação do projeto.

O teste automatizado se divide em:

1. **Teste Unitário**: É o nível de teste em que se objetiva a captura de defeitos em nível individual, ou seja, em apenas uma unidade funcional, independentemente da sua interação com os demais elementos do projeto. Esta unidade funcional pode ser concebida com uma classe ou um método, dependendo do nível de complexidade envolvida.
2. **Teste de Integração**: É um nível de teste complementar ao unitário. Sabendo que a unidade não apresenta inconsistências em seu escopo individual, é necessário pois, aferir se este comportamento se repete quando se relaciona/comunica com os demais componentes do sistema (outras classes, serviços, banco de dados, etc.).
3. **Teste de Sistema**: é o nível de teste em que se objetiva aferir o comportamento do software como um todo. Mesmo que este todo seja apenas o resultado de uma *Sprint*. Este teste deve ser executado a cada nova iteração. Este nível de teste também é chamado de Teste Funcional Automatizado.

3.4.2 Quem:

1. **Teste Unitário e Teste de Integração -** a implementação dos casos de teste em nível unitário e integração é responsabilidade do membro da *Scrum Team* responsável pela implementação do projeto de desenvolvimento. Este responsável foi identificado na *Sprint Planning*.
2. **Teste de Sistema**:
   2.1. **Para um Novo Projeto de Desenvolvimento**: Componente da *Scrum Team* que exerce o papel de analista de teste e que foi escolhido para esta tarefa da *Sprint Planning*.

2.2. **Para um Novo Projeto de Desenvolvimento**: Membro da equipe da qualidade que exerce o papel de analista de teste e que foi escolhido para esta tarefa em comum acordo entre o líder da equipe do projeto e o líder da equipe de qualidade.

3.4.3 Quando:

É pré-condição para a implementação de teste automatizado que a funcionalidade alvo dos testes esteja entregue, ou seja, conste de uma *build* do projeto disponível no ambiente de teste.

3.4.4 Como:

A implementação do teste automatizado é feita da seguinte forma:

1. **Teste Unitário e Integração**:

   1.1. Os níveis de Teste Unitário e Integração devem ser implementados pelo desenvolvedor em um único projeto, junto com o projeto de desenvolvimento. Estes níveis de teste só serão exigidos para novos projetos. **Para sistemas legados, somente o Teste de Sistema será exigido**.

   1.2. Devem existir casos de testes unitários para todas as operações de negócio de uma classe, que não envolva a interação com outras classes do projeto.

   1.3. Devem existir casos de testes de integração para todas as operações de negócio de uma classe, que envolva a interação com outras classes/componentes do projeto.

   1.4. Devem existir casos de testes de integração para todas as operações CRUD (*Create*, *Restore*, *Update*, *Delete* – Criação, Consulta, Alteração e Exclusão) de todas as classes que persistirão objetos nos mecanismos de persistência definidos para o projeto.

   1.5. Todos estes casos de testes devem estar documentados na Ferramenta *Testlink*.

2. **Teste de Sistema**:

   2.1. Este nível de teste é exigido tanto para novos projetos, quanto para os legados. Sua implementação consiste da criação de um novo projeto de teste automatizado para o software, a partir do modelo de arquitetura de teste de referência disponível nos repositórios de controle de versão para cada uma das plataformas de desenvolvimento utilizadas na GPS, Java, C# e PHP.

   2.2. Execução manual do teste, gravando o passo a passo desta execução. Esta gravação é efetuada usando recursos do *framework* de testes *Selenium*.

   2.3. O *Script* de gravação é exportado para a linguagem de programação do projeto, adicionado ao projeto de teste, com as devidas adaptações que se fizerem necessárias;

   2.4. Quando todos os testes forem implementados, uma nova versão do projeto de teste é disponibilizada no servidor de integração contínua.

3. Cada projeto de teste deverá implementar a integração entre o *Selenium Webdriver* e o *Testlink*, de forma que a cada execução de um caso de teste, de forma automatizada, o próprio *Selenium* registre no *Testlink* o resultado obtido.

3.4.5 Prazo:

O tempo de duração desta Atividade não pode ser superior ao tempo despendido na implementação da funcionalidade.

3.4.6 Entradas:

Casos de teste, cuja forma de execução é automatizada.

3.4.7 Saídas:

Nova versão do projeto de testes automatizado disponível no servidor de integração contínua.

3.4.8 Fluxo:

Seguir para a etapa de Execução dos Testes.

4 Execução

Nesta etapa estão concentradas todas as atividades necessárias à execução dos testes, tanto daqueles definidos cada uma das *Sprints* de um novo projeto, quanto daqueles definidos Para um Novo Projeto de Desenvolvimento de manutenção de um software existente. A representação visual das atividades componentes desta etapa do processo, pode ser observada na Figura 4. O detalhamento destas atividades está disponível a partir da Subseção 4.1.

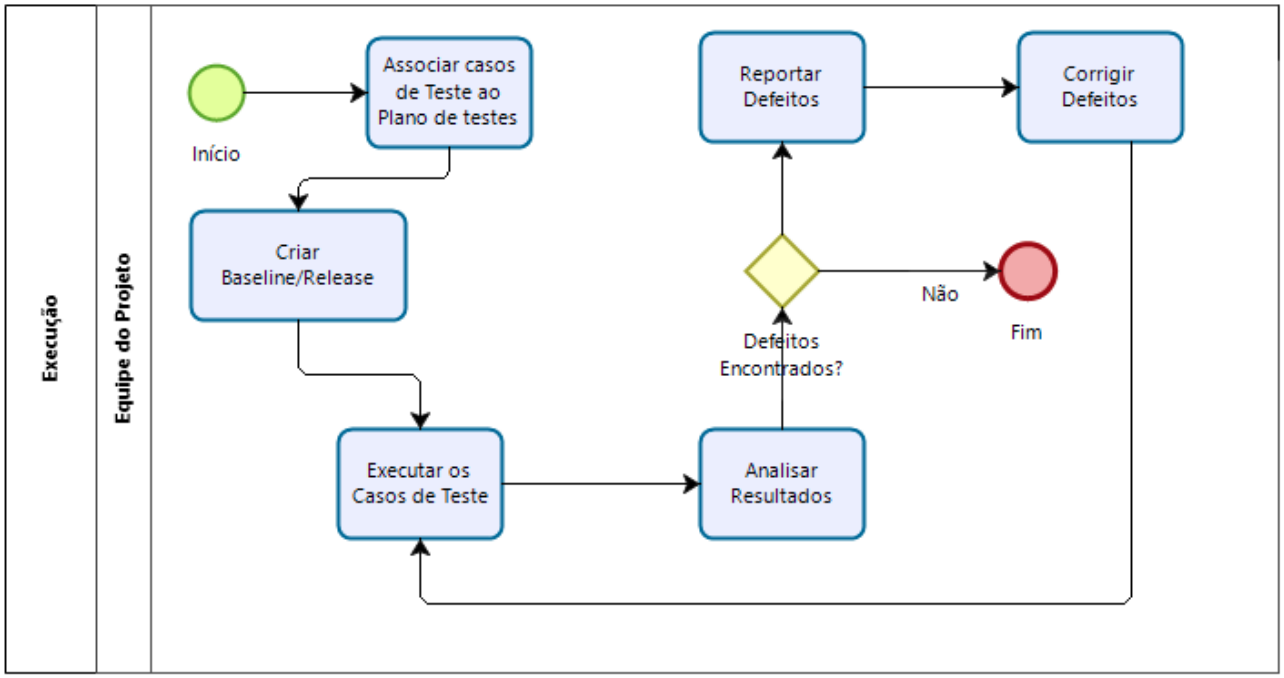

Figura 4 – Etapa de Execução.

4.1 Associar os Casos de Teste ao Plano de Teste

Considerando o uso da ferramenta *Testlink* para o gerenciamento do projeto de teste, esta é uma atividade necessária para a liberação dos casos de teste para execução.

4.1.1 O Que:

Os casos de teste para serem executados devem estar associadas a:

- Um Plano de Testes.
- Uma *Build* do projeto. Ver atividade 4.2.

E esta associação tem o objetivo de definir o escopo da execução dos testes. Este escopo pode incluir:

1. Se somente aqueles definidos para uma determinada *Sprint*, no caso de um novo projeto, ou se serão incluídos outros casos de testes existentes para funcionalidades que eventualmente tenha sofrido alguma alteração na nova *Sprint*.
2. Somente os casos de teste definidos Para um Novo Projeto de Manutenção, ou se serão incluídos outros casos de teste para aferir a não adição de algum novo defeito a partir da manutenção. Estes outros casos de teste, são casos de teste existentes referentes a manutenções anteriores.

4.1.2 Quem:

1. **Para um Novo Projeto de Desenvolvimento**, esta responsabilidade é atribuída àquele membro da *Scrum Team* eleito para a criação dos casos de teste, pois este membro possui o conhecimento do escopo do teste, sendo capaz da seleção correta dos casos de teste para a execução.
2. **Para um Novo Projeto de Manutenção**, esta responsabilidade é do Analista responsável pela especificação da solução proposta.

4.1.3 Quando:

Sempre que uma nova versão do projeto estiver disponível para teste. Esta nova versão estará incrementada com os produtos gerados durante uma nova *Sprint* de um novo projeto ou com a alteração imposta pela manutenção em um projeto existente.

Esta disponibilização deve ser comunicada pelo membro da *Scrum Team* ou equipe do projeto ao responsável pela execução dos testes, da *Sprint*/Manutenção.

4.1.4 Como:

Utilizando o *Testlink* para associar os casos de teste ao Plano de Teste, da *Sprint* ou Manutenção.

4.1.5 Prazo:

O Prazo máximo para a execução desta atividade é de 1 (uma) hora.

4.1.6 Entradas:

Nova versão testável do sistema.

4.1.7 Saídas:

Conjunto de casos de teste devidamente associados ao Plano de Testes correspondente.

4.1.8 Fluxo:

Segue-se para a atividade 4.2.

4.2 Criar uma *Baseline/Release* para a Execução dos Testes.

O *Testlink* exige que para a execução dos testes seja criada uma *Baseline*/*Release*. Isto se dá pela necessidade de se manter a sincronização entre o projeto de testes e o projeto de desenvolvimento, de forma que esta nova *Baseline*/*Release*, deve ser nominada/numerada, da mesma forma que a versão do projeto de desenvolvimento.

4.2.1 O Que:

Criar uma nova versão do projeto de teste correspondente à versão do projeto de desenvolvimento, para a execução dos casos de teste.

4.2.2 Quem:

Os mesmos responsáveis pela execução da Atividade, descrita na Subseção 4.1, deste Documento.

4.2.3 Quando:

Quando uma nova versão testável do software estiver disponível no ambiente de teste.

4.2.4 Como:

Utilizando o *Testlink* para a criação da nova *Release/Baseline* do projeto de testes.

4.2.5 Prazo:

O Prazo máximo para a execução desta atividade é de 1/2 (meia) hora.

4.2.6 Entradas:

Nova versão testável do sistema.

4.2.7 Saídas:

Nova *Baseline/Release* do projeto de testes criada no *Testlink*.

4.2.8 Fluxo:

Segue-se para o Fluxo 4.3.

4.3 Executar os Casos de Teste

A execução dos casos de testes vai ser efetivada conforme consta do Plano de Testes do Projeto. A forma de execução é manual ou automatizada.

4.3.1 O Que:

Execução dos casos de teste definidos para a cobertura dos cenários definidos para a *Sprint* ou para a Manutenção.

4.3.2 Quem:

1. **Execução Manual** – Membro da *Scrum Team* / Equipe Projeto, previamente definido, através da atribuição dos casos de teste para execução, na Ferramenta *Testlink*.
2. **Execução Automatizada**:
    1. **Teste Unitário e Integração** – Execução durante a *Build* do Projeto, normalmente disparada pelo *Jenkins*.
    2. **Teste Funcional Automatizado** – Execução, também disparada pelo *Jenkins*, neste caso, após o *deploy* da versão no Ambiente de Testes.

4.3.3 Quando:

Sempre que uma versão testável do Projeto seja disponibilizada para teste.

4.3.4 Como:

1. Na **Execução Manual**, o testador executa o sistema seguindo o roteiro para a execução dos casos de teste. Para cada não conformidade encontrada, deve ser registrada uma ocorrência, baseada com evidências, quem podem ser *prints* de tela, por exemplo.
2. A **Execução Automatizada** é disparada pelo Servidor de Integração Contínua, conforme descrito na Subseção 4.3.2.

4.3.5 Prazo:

O Prazo para a execução do teste deve ser definido na *Sprint Planning*.

4.3.6 Entradas:

Nova versão testável da aplicação.

4.3.7 Saídas:

Relatórios de Resultados de Testes.

4.3.8 Fluxo:

O Fluxo segue para a Atividade 4.4.

4.4 Análise dos Resultados

4.4.1 O Que:

Nessa atividade os resultados da execução dos testes são comparados com os resultados esperados. Aqueles que não estiverem em conformidade deverão reportados em detalhes para facilitar a identificação dos defeitos pela equipe de desenvolvimento.

Fazem parte das atividades de análise de resultados:

- Revisão dos resultados de não conformidade (avaliar evidências de testes e avaliar e priorizar a lista de *bugs*).
- Formalizar defeitos encontrados (coletar, detalhar e classificar os defeitos por nível de severidade e priorização),

4.4.2 Quem:

Responsável pela execução dos testes, junto com algum membro da equipe previamente selecionado.

4.4.3 Quando:

Imediatamente após o encerramento da execução de um ciclo de testes.

4.4.4 Como:

Cada execução gera um relatório de resultados. Uma cópia deste relatório deve ser disponibilizada no repositório de controle de versão no Gitlab.

A partir da análise dos resultados obtidos em relação aos esperados, é que a tomada de decisão em relação à liberação ou não da versão da Aplicação. Esta decisão será tomada levando em consideração, tanto a criticalidade quanto o impacto dos defeitos potencialmente encontrados.

4.4.5 Prazo:

O prazo para execução desta atividade não deve ser superior a 2 (duas) horas.

4.4.6 Entradas:

Relatórios de execução do ciclo de testes.

4.4.7 Saídas:

Tomada de decisão em relação à entregue da versão conforme o resultado dos testes ou se haverá necessidade de correção em virtude de defeitos encontrados.

4.4.8 Fluxo:

O fluxo segue para a Atividade 4.4, quando defeitos são encontrados durante a execução do teste. Segue para a Etapa 5, quando defeitos não são encontrados.

4.5 Reportar Defeitos

Esta atividade é executada sempre que algum defeito for encontrado durante a execução dos testes.

4.5.1 O Que:

Comunicar à equipe os defeitos encontrados durante o ciclo de execução dos testes.

4.5.2 Quem:

Responsável pela execução dos testes.

4.5.3 Quando:

Imediatamente após o encerramento de um ciclo de testes.

4.5.4 Como:

Relatando uma atividade no *Redmine*, informando o defeito encontrado, junto com as evidências para sua comprovação. Esta atividade deve ser atribuída ao responsável pela implementação da funcionalidade em que o defeito foi encontrado.

Deve-se relatar uma atividade para cada defeito encontrado.

4.5.5 Prazo:

O prazo para esta atividade é de no máximo 2 horas.

4.5.6 Entradas:

Relatório de resultado da execução dos testes.

4.5.7 Saídas:

Atividade(s) relatada(s) no Redmine solicitando a correção do(s) defeito(s) encontrado(s).

4.5.8 Fluxo:

Segue-se para a Atividade 4.5.

4.6 Corrigir Defeitos

4.6.1 O Que:

Alteração no código-fonte da aplicação para a correção de defeito(s) encontrado(s) durante a execução dos testes.

4.6.2 Quem:

Responsável pela implementação da funcionalidade.

4.6.3 Quando:

A partir da notificação da existência de defeito(s) para ser(em) corrigido(s).

4.6.4 Como:

Através de edição do código-fonte da aplicação, alterando o trecho de código onde o(s) defeito(s) foi(ram) encontrado(s).

4.6.5 Prazo:

O Prazo para correção deste defeito deve ser decidido pela equipe, de forma que não comprometa o prazo de duração planejado para a *Sprint*.

4.6.6 Entradas:

Atividade(s) no Redmine reportando a presença de defeitos na versão da aplicação disponibilizada para testes.

4.6.7 Saídas:

Nova versão da aplicação em que contempla a correção do(s) defeito(s) reportado(s).

4.6.8 Fluxo:

Volta-se novamente à Atividade 4.3.

5 Entrega

Nesta Etapa são analisados indicadores extraídos durante a execução dos testes para o trabalho realizado de forma quantitativa e qualitativa. O resultado dessa análise permitirá registrar as lições aprendidas durante a execução dos testes. A representação visual das atividades componentes desta etapa pode ser observada na Figura 5. A descrição destas atividades acontece a partir da Subseção 5.1.

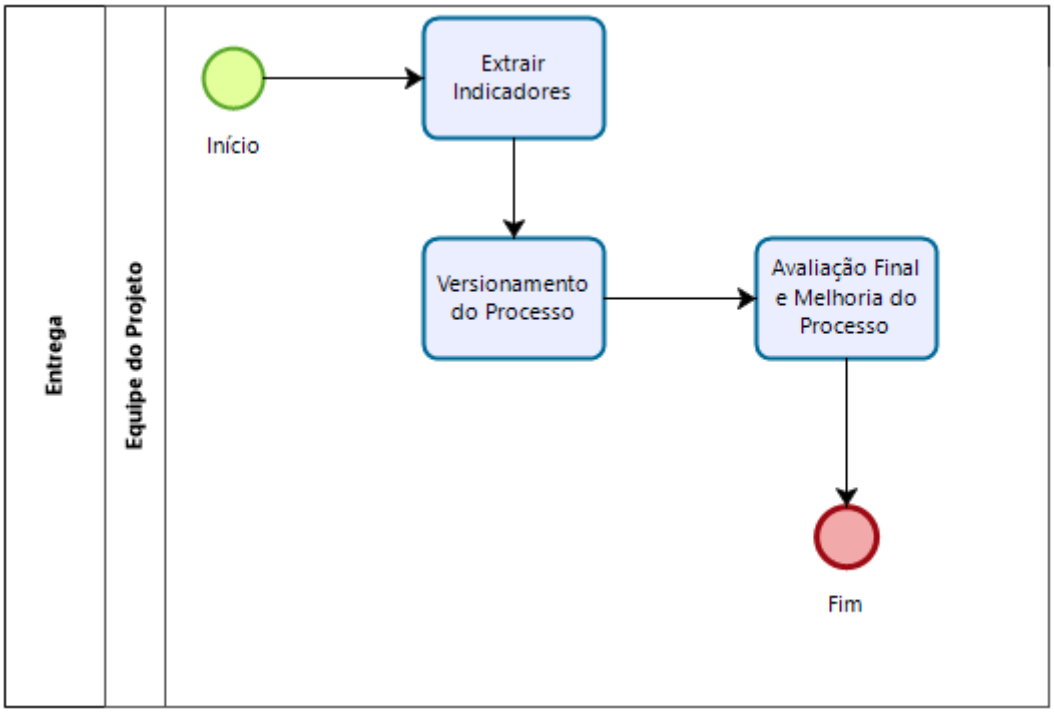

Figura 5 – Etapa de Entrega.

5.1 Extração de Indicadores

5.1.1 O Que:

Extração de indicadores (indicadores quantitativos, de produtividade, confiabilidade, financeiros e de nível de satisfação, individuais e do projeto)

5.1.2 Quando:

1. Após o teste da última Sprint do projeto, em se tratando de um **Novo Projeto**.
2. Após o encerramento do teste, no caso de uma **Manutenção de Software**.

5.1.3 Quem:

Esta tarefa deverá ser executada por algum dos membros da equipe, sob coordenação de um profissional da área da qualidade.

5.1.4 Como:

Esta atividade consiste em:

- Registrar e analisar os indicadores de testes,
- Lista de defeitos,
- Horas planejadas x horas executadas,
- Indicadores de sucesso,
- Lições aprendidas, caminhos críticos e
- Realizar uma divulgação corporativa sobre os resultados obtidos

Todos estes itens deve constar de um artefato chamado ResumoTeste, que deverá estar disponível no repositório de documentação do projeto no *Gitlab*.

5.1.5 Prazo:

Esta atividade deverá durar no máximo 16hs.

5.1.6 Entradas:

1. Projeto de Teste devidamente documentado no *Testlink*.

5.1.7 Saídas:

Artefato descrevendo ResumoTeste, disponível no *Gitlab*.

5.1.8 Fluxo:

Segue para a Atividade 5.2.

5.2 Versionamento do Processo de Testes.

Todos os artefatos gerados em virtude da execução das atividades da GPS são mantidos sob o controle de versão em repositórios específicos no GitLab. Os artefatos de um projeto de testes seguem a mesma dinâmica dos demais.

5.2.1 O Que:

Verificar se todos os artefatos de um determinado projeto de testes estão devidamente disponibilizados no repositório do controle de versão correspondente.

5.2.2 Quando:

Após o encerramento dos ciclos de testes de uma Sprint ou de uma manutenção de software.

5.2.3 Quem:

Esta tarefa deverá ser executada por algum dos membros da equipe, sob coordenação de um profissional da área da qualidade.

5.2.4 Como:

1. No *Testlink*. Esta ferramenta já mantêm um controle de versão sobre todos os artefatos gerados. Contudo, por motivações gerenciais, como rastreabilidade, por exemplo, este controle de versão ocorre automaticamente, considerando a dinâmica de criação de Plano de Testes e *Build/Release*, para cada ciclo de execução de testes.
2. Scripts para geração de dados – Conferir se todos os scripts gerados e utilizados durante o processo de teste estejam disponibilizados no repositório do controle de versão no *Gitlab*.
3. Código-fonte do teste automatizado:

    3.1. Teste Unitário e de Integração – deverão constar da TAG criada por ocasião da entrega da versão do projeto de desenvolvimento.

    3.2. Teste de Sistema – Deverá ser gerada uma nova TAG, identificando que se trata da versão final do projeto de Teste.

5.2.5 Prazo:

Esta atividade deverá ser executada no prazo máximo de 8 (oito) horas.

5.2.6 Entradas:

Atividade relatada no Redmine solicitando a criação de TAG dos artefatos do projeto de teste.

5.2.7 Saídas:

TAG do projeto de teste criada no repositório do controle de versão, no *Gitlab*.

5.2.8 Fluxo:

Segue para a Atividade 5.3.

5.3 Avaliação Final e Melhoria do Processo.

Fazem parte da atividade de encerramento de processo: avaliação final e melhoria do processo (avaliar riscos planejados x concretizados, performance, atualizar plano de melhoria contínua e sinalizar o encerramento do processo).

5.3.1 O Que:

1. Para um Novo Projeto de Desenvolvimento, esta atividade consiste em compilar as observações relatadas durantes os eventos de avaliação de cada uma das *Sprints* do Projeto (*Sprint Review*), pois as atividades de teste fazem parte do escopo desta avaliação.

2. Para um Novo Projeto de Manutenção, consiste de um *feedback* do membro da equipe, responsável pelas atividades de teste, ao coordenador da equipe de qualidade. Este *feedback* deverá reportar o que foi bem-sucedido e o que não foi.

5.3.2 Quando:

Após o encerramento do ciclo de testes de um novo projeto de desenvolvimento ou de manutenção de software.

5.3.3 Quem:

Preferencialmente toda a equipe. Quando não for possível, pelo menos por aqueles componentes da equipe envolvidos nas atividades de teste.

5.3.4 Como:

1. No caso de um **Novo Projeto** de desenvolvimento, esta atividade deverá ocorrer durante uma reunião em que será avaliado o desenvolvimento do projeto, como um todo. Todos os componentes da equipe deverão participar e apresentar suas considerações a respeito do processo de testes.
2. Para um Novo Projeto de Manutenção, esta avaliação poderá ser feita através de uma reunião entre o responsável pelos testes e a equipe de qualidade.

5.3.5 Prazo:

Esta atividade deverá ter um prazo máximo de 1 (uma) hora.

5.3.6 Entradas:

Projeto de Teste encerrado.

5.3.7 Saídas:

Registro de sugestões para a melhoria contínua do processo.

Proposta para a divulgação dos resultados do projeto.

5.3.8 Fluxo:

O Processo se encerra nesta atividade.

6 Considerações Finais

Este documento apresentou o PTTS, um processo de teste concebido dentro da GPS, que incorpora um conjunto de práticas ágeis, particularmente aquelas associadas a metodologia SCRUM.

Vale ressaltar que este processo deverá ser um guia, uma diretriz para a execução das atividades de teste no âmbito da GPS, com o objetivo da melhoria da qualidade dos produtos entregues por esta Gerência.

Os próximos passos é a definição, por parte do corpo diretivo da GPS, das etapas de implantação deste processo.