Instituto Federal de Educação Ciência e Tecnologia de São Paulo - Campus Birigui

Rangel Vieira Perico

TECHNOSIGHT

Birigui
2022

Sumário

1. Introdução

A forma com que uma empresa trabalha é bem importante para o ambiente de trabalho e envolve muitos detalhes que geralmente não pensamos no dia a dia, mas que definem a taxa de sucesso de nossos projetos. Eles que dizem como eventuais problemas podem ser corrigidos, as datas de entrega das funcionalidades, o que deve ser entregue, pontos de retorno em casos de falha do software e muitos outros aspectos, que facilitam a vida dos desenvolvedores e trazem satisfação dos interessados quanto aos produtos da organização.

Visto toda a matéria passada durante o semestre, vários conceitos foram abordados e várias sugestões de modelos foram mostradas, portanto, apresentarei posteriormente, o que considerei como os tópicos mais sensíveis que devem ser tratados desde o começo de uma empresa, mas que não necessariamente abordam tudo, afinal, prezei o máximo possível entre um balanço de qualidade e velocidade de entrega, evitando sobrecarga de funções, que acarreta em queda de qualidade e produtividade.

2. Como trabalhamos

A Technosight preza pela qualidade e velocidade de entrega de seus produtos e para que isto ocorra, montamos um cronograma integrado a nossa metodologia escolhida, no caso, o SCRUM, uma metodologia ágil de rápida implementação mas de grande valia para a organização estrutural da organização.

2.1 Backlog do Produto

Essa etapa protagonizada ao *Product Owner* define as informações iniciais do projeto, lidando diretamente com as necessidades da parte interessada, por isso, tem grande impacto nas chances de sucesso do software como um todo.

2.1.1 Levantamento de requisitos

Essa é uma das fases iniciais do projeto e visa demarcar o escopo do mesmo, dizendo o que deverá ser feito durante o percurso de desenvolvimento, sejam adições, remoções e modificações em geral.

Por tratar de uma variável delicada, no caso pessoas, é bem importante ter a ideia correta do que deve ser desenvolvido e os detalhes devem ser muito bem explicitados, dos quais posteriormente serão mostrados à equipe como épicos, estórias e tarefas.

Para questões de documentação e facilidade de consulta dos mesmos, uma boa prática seria a gravação de reuniões, porém, os áudios podem facilmente se tornar arquivos grandes e difíceis de se administrar. Imagine uma sessão com duração de 30 minutos, consumiria um certo tempo para encontrar o tópico desejado, porém imagine que essas sessões podem ocorrer durante mais de um dia da semana. Para evitar todo esse emaranhado, será interessante a utilização de ferramentas de transcrição de áudio, que transformam, com auxílio de inteligência artificial, falas em texto. É muito mais simples de procurar em um arquivo de texto um tópico, pois existem ferramentas de busca de palavras chave em arquivos texto, tornando esse processo muito mais rápido e direto, além de diminuírem a chance de alguém que está ouvindo um áudio não conseguir entender o que foi dito. Claro, o áudio original ainda estará disponível, mas junto a ele haverá a transcrição para poupar o tempo de busca.

Será feito o levantamento dos requisitos de usuário, sistema, adiados, funcionais e não funcionais, dos quais serão documentados e categorizados de acordo com a seguinte tabela:

Tabela 1 - Categorias de requisitos

Usuário	Sistema	Adiados
São ideias de alto nível, descrevendo uma tarefa sem entrar em detalhes técnicos	Descrição técnica de como o requisito de usuário será feito (tópico 2.1.3)	Aqueles que não serão feitos no momento, mas que podem agregar valor ao produto

Fonte: Os autores

2.1.2 Análise de Riscos

Os riscos estão em todos os projetos, não importa qual seja, dos quais podem ser mensurados antecipadamente para evitar longas reuniões e discussões de como resolver um, quando pegam o time desprevenido.

Pensando nisso, existem várias formas de se levantar e categorizar os problemas que podem ocorrer durante o desenvolvimento de um projeto, sejam eles internos ou externos.

Tabela 2 - Categorização de riscos

Riscos internos	Riscos externos
1. Defeito em equipamentos, 2. Falta de conhecimento tecnológico, 3. Estouro de prazo por conta de erros no desenvolvimento, 4. Erro ao definir a complexidade do sistema, 5. Alterações no escopo do projeto, 6. Etc…	1. Fenômenos naturais, 2. Crises políticas, 3. Crises econômicas, 4. Doenças, 5. Alteração em legislações, 6. Alteração na compra e venda do mercado, 7. Etc…

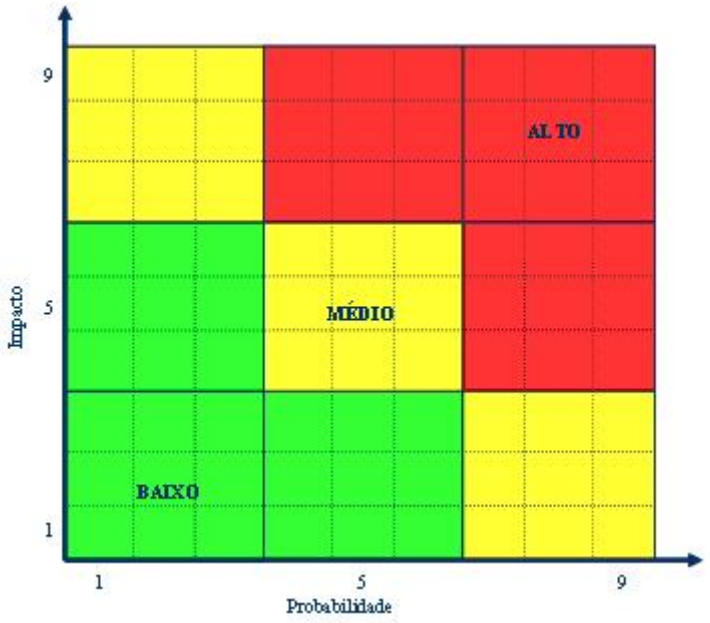
Fonte: ZITKO, Paula[1]

Portanto, uma vez que haja a catalogação dos riscos e seja feita a devida separação por categoria, sendo risco interno ou externo, deve-se fazer uma descrição, encontrar as fontes do problema e dizer como ele pode afetar o projeto.

Uma vez que todos os riscos estejam levantados, é interessante a criação de uma matriz de risco, da qual indica os danos que cada um pode trazer ao projeto e as chances de que ele possa ocorrer:

[1] ZITKO, Paula. Disciplina: Introdução ao Gerenciamento de Projetos – IGPE5 Curso: Tecnologia em Sistemas para Internet. Disponível em: <https://ead.bri.ifsp.edu.br/moodle/pluginfile.php/90705/mod_resource/content/1/AULA2.3%20-Riscos.pdf>. Acesso em: 23 maio 2022.

Figura 1 - Matriz de risco



Fonte: ZITKO, Paula[2]

Após isto, um planejamento de resposta é criado, dizendo quais os passos necessários a serem tomados a fim de tentar amenizar o mesmo, porém, isto abre margem para efeitos colaterais (novos problemas) e riscos residuais (aqueles que não se tem controle e ainda sim continuam), portanto, deve-se fazer um levantamento destes, caso ocorram.

2.1.3 Casos de uso

Serão descrições dos requisitos funcionais, indicando a forma com que o sistema em geral se comunica, podendo ter diagramas gráficos ou apenas textuais, mas que tragam facilidade de interpretação aos desenvolvedores. No caso da utilização dos casos de uso do tipo caixa, o tipo recomendado será o **estilo caixa preta**, desta forma, não fica dependente de tecnologias e abre um leque maior de possibilidades para a implementação.

[2] ZITKO, Paula. Disciplina: Introdução ao Gerenciamento de Projetos – IGPE5 Curso: Tecnologia em Sistemas para Internet. Disponível em: <https://ead.bri.ifsp.edu.br/moodle/pluginfile.php/90705/mod_resource/content/1/AULA2.3%20-Riscos.pdf>. Acesso em: 23 maio 2022.

Tabela 3 - Aspectos funcionais e não funcionais do sistema

Funcionais	Não funcionais
Aspectos técnicos do sistema descritos em estilo caixa preta. Login, Cálculos, Mostrar informações ao cliente, Comunicação entre módulos, Etc…	O que se espera do software, mas que não altera sua funcionalidade diretamente. Eficiência, Portabilidade, Organizacionais, Éticos, Legais, Etc…

Fonte: Os autores

Seguindo os padrões definidos, temos os cenários de sucesso, pré condições, fluxo básico e alternativo, dentre outros. Desta forma, ao se definir como os requisitos devem se comunicar, é importante fazer a conexão entre eles e decidir as condições para que a tarefa seja um sucesso, levando em consideração o estilo anteriormente definido.

2.1.4 Projeto de dados

Esta seção trata sobre padronizações e técnicas utilizadas quanto a organização dos dados armazenados e manipulados durante e após a criação do software.

Tais padronizações são importantes para o andamento do projeto e futuras manutenções, afinal, quanto mais bagunçado fica um código fonte, menor o rendimento dos desenvolvedores, o que ocasiona atrasos e possíveis refatorações totais que custam e demoram muito até ficarem prontas, dependendo da complexidade do sistema.

2.1.4.1 Diagrama entidade-relacionamento

Assim que os requisitos estiverem definidos, vamos para a parte de modelagem de banco de dados, utilizando o diagrama entidade-relacional, que dita tabelas, chaves, campos e outros.

Nesta parte do projeto é interessante definir um padrão específico que deverá ser utilizado por todos quando os elementos forem nomeados, do qual como sugestão poderia ser utilizado da seguinte forma:

Tabela 4 - Sugestões para padronização de nomes em banco de dados

Campo	Exemplo padrão
Chave primária	NomeDaTabela_PK
Chave estrangeira	NomeDaTabela_NomeDaTabelaEstrangeira_FK_nn
Check	NomeDaTabela_CK_nn
Chave única	NomeDaTabela_UK_nn

Fonte: SANTOS, Helen[3]

Ao final de cada nome, existe o nn, que se refere ao número de ocorrência. Em outras palavras, 01, 02, 03 e etc.

2.1.4.2 Dicionário de dados

Por mais que a consulta em bancos de dados tenha tornado este item um pouco defasado, tem algumas coisas que o mesmo não supre, como funções no código fonte.

Se tratando de dados, há um grande volume, e seu gerenciamento é crucial para o desenvolvimento, sendo assim, é importante existirem documentos que dizem quais e como utilizar dos recursos já disponíveis dentro de um código, evitando assim a tão temida duplicidade.

Sendo assim, uma forma simples, mas que pode fazer toda a diferença, de documentar seria utilizando de tabelas, dizendo o componente em questão, com quem se comunica, sua descrição e as variáveis contidas no mesmo. Se necessário, uma outra tabela pode ser criada explicando os campos, deixando-os mais legíveis.

Tabela 5 - Exemplo de dicionário de dados

Nome da função	Descrição	Campos	Métodos
setupLogin	Faz o login de um novo usuário	username password	set get

Fonte: Os autores

Outro tópico a ser abordado é o padrão de escrita destes, uma vez que existam muitos como, nomeExemplo, NomeExemplo, nome_exemplo e diversos outros, isto considerando apenas variáveis, visto que, as estruturas de

[3] SANTOS, Helen. Projeto de Software. Disponível em: <https://docs.google.com/presentation/d/1_4DEr6vwFU6redoT8yYc2jrQDgFk-fWB/edit#slide=id.p1>. Acesso em 23 maio 2022.

dados possuem outros padrões. Pensando nisso, a seguinte tabela foi proposta, a fim de padronizar a forma com que algumas coisas são descritas.

Tabela 6 - Exemplo de padronizações em nomes

Tipo	Exemplo
Variáveis	nomeExemplo
Classes	NomeClasse
Funções	nomeFuncao
Constantes	CONSTANTE_EXEMPLO

Fonte: Os autores

2.1.5 Projeto de Interface

Assim como padronizações de variáveis e estruturas são importantes, ter padrões gráficos é tão importante quanto, servindo para evitar que um mesmo botão esteja em diferentes lugares dependendo de qual tela estivermos. Também é interessante para evitar mudanças incongruentes na paleta de cores do sistema.

Primeiramente, um levantamento de quais telas deverão existir no sistema deve ser feita, onde, caso uma tela não tenha sido previamente pensada, deve seguir os padrões propostos para as outras, a fim de evitar confusões. Exemplo:

Tabela 7 - Exemplo de levantamento de telas

Tela	Função
Login	Servirá para o usuário se conectar em sua conta e conseguir acessar nossos serviços
Cadastrar	Caso o usuário não possuir conta, poderá se cadastrar por esta página

Fonte: Os autores

Outra coisa a se fazer será verificar quais serão os botões que terão em cada tela e dizer a sua função nesta. Também definindo uma posição padrão para o mesmo. Exemplo:

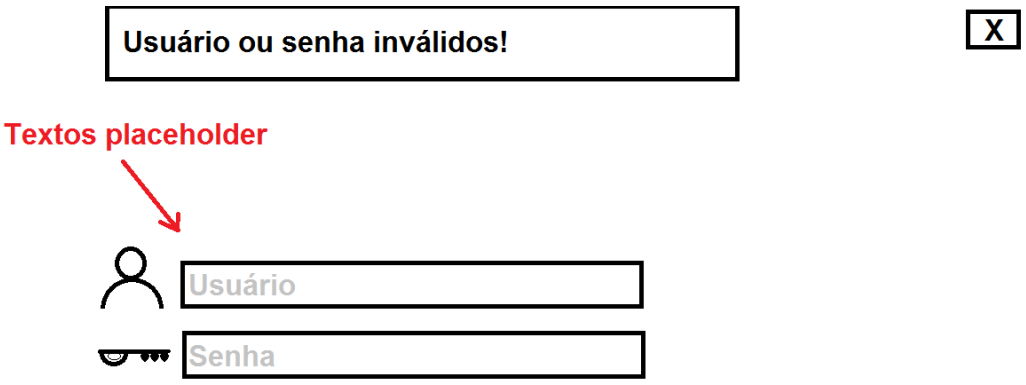
Tabela 8 - Descrição de alguns elementos da tela de login

Elemento	Ação	Tela	Posição
Botão sair	Fecha a tela atual e retrocede à página principal	Login	Topo direito
Caixa de texto nome	Usuário digita seu nome neste elemento	Login	Centralizado
Caixa de texto senha	Usuário digita sua senha neste elemento	Login	Centralizado, abaixo da caixa de texto nome
Caixa de texto status	Indica caso algum campo esteja faltando ou se o login for inválido	Login	Topo da tela, centralizado

Fonte: Os autores

Assim que os elementos estiverem devidamente catalogados, exemplos da tela, assim como sua paleta de cores, devem ser feitos para indicar como será implementado no sistema futuramente. Se forem necessárias observações quanto a alguma informação na tela, estas podem ser feitas utilizando uma flecha apontando para o elemento, junto a um texto, ambos de cor vermelha.

Figura 2 - Exemplo de tela de login



Fonte: Os autores

No exemplo acima, a paleta poderia ser descrita como:

Tabela 9 - Descrições das paletas de cores das telas

Tela	**Paleta**
Login	Preto, Branco, Cinza

Fonte: Os autores

2.1.6 Cálculo das métricas de pontos por função

Assim que um projeto está sendo finalizado, existem diversas informações relevantes que podem servir para futuros softwares, portanto, ter anotado informações como custo, quantidade de pessoas no projeto, tempo necessário para conclusão e outros, são de grande valia.

Com estas informações em mãos, é possível calcular vários dados referentes ao projeto e presumir futuramente, quanto tempo um projeto levará para ser concluído. Para tanto, deve-se primeiro catalogar o sistema em arquivos lógicos internos, arquivos de interface externa, entradas externas, saídas externas e consultas externas, então, definir qual seu nível de complexidade e depois, em uma planilha, colocar um tabela preenchendo os seguintes campos:

Tabela 10 - Tabela de métrica de pontos por função

Componente Lógico	Complexidade Funcional			Total Complexidade	Total Tipo Componente
Arquivo Lógico Interno – ALI		Simples	X 7 =		
		Média	X 10 =		
		Complexa	X 15 =		
Arquivo de Interface Externa – AIE		Simples	X 5 =		
		Média	X 7 =		
		Complexa	X 10 =		
Entradas Externas – EE		Simples	X 3 =		
		Média	X 4 =		
		Complexa	X 6 =		
Saídas Externas – SE		Simples	X 4 =		
		Média	X 5 =		
		Complexa	X 7 =		
Consultas Externas – CE		Simples	X 3 =		
		Média	X 4 =		
		Complexa	X 6 =		
Total PF Bruto					

Fonte: SANTOS, Helen[4]

Para definir o nível de complexidade de cada documento, é necessária a referência de algumas tabelas que podem ser consultadas neste link[5].

2.2 Sprint Planning

Esse é o evento onde todas as partes da equipe se juntam (PO, Scrum Master e desenvolvedores), e começam a discutir o que será feito durante o período da sprint. Neste momento são definidas as tarefas que são feitas, o prazo que cada uma deve levar e outras informações do tipo. É muito importante que épicos e estórias estejam muito bem definidas, uma vez que

[4] SANTOS, Helen. O tempo e o Processo de Desenvolvimento de Software. Disponível em: <https://docs.google.com/presentation/d/1UUSM1D2lJ0B4IsSoHRUE1b5hOcH9zwVn/edit#slide=id.p56>. Acesso em: 23 maio 2022.

[5] SANTOS, Helen. O tempo e o Processo de Desenvolvimento de Software. Disponível em: <https://docs.google.com/presentation/d/1UUSM1D2lJ0B4IsSoHRUE1b5hOcH9zwVn/edit#slide=id.p56>. Acesso em: 23 maio 2022.

esta é uma parte bem sensível e pode afetar não apenas os prazos como também a qualidade do produto.

2.2.1 Análise das métricas por pontos de função

Esta etapa servirá para substituir o SCRUM Poker, que tem a função de estimar tempo para as tarefas, entretanto, se baseia em aproximações e “chutes” ditos pelos próprios desenvolvedores, entretanto, isto abre margem para valores cada vez mais distantes da realidade, não tornando esta uma forma muito convincente, principalmente para a parte interessada no projeto.

Os cálculos feitos anteriormente, serão consultados a fim de definir as tarefas possíveis de serem feitas naquele sprint.

2.3 Sprint Daily Meeting

Este é um evento que ocorre diariamente, tem duração de cerca de 15 minutos e serve para os colaboradores dizerem o que será feito durante o dia, o que foi feito anteriormente e assim por diante. São assuntos rápidos mas que trazem mais conhecimento por parte da equipe sobre o que cada um está desenvolvendo.

2.4 Sprint Review

Nesta fase a parte interessada e os desenvolvedores participam, onde será mostrado tudo que foi feito durante a sprint, o que foi concluído e afins. Os *stackholders* verificarão e darão feedback quanto ao progresso e dirão se tudo está ocorrendo conforme o que foi planejado.

Como é uma fase mais específica e não há muito além de receber críticas do que está sendo feito, a prioridade seria estar gravando o que está sendo discutido e estar separando pontos de interesse, por exemplo, alterações necessárias propostas pelos clientes, que futuramente estarão em outras sprints para serem implementadas.

2.5 Sprint Retro

Este evento ocorre quase ao fim da sprint e serve para verificar se tudo ocorreu como o planejado, onde podem acontecer eventuais reestruturações do plano para desenvolvimento do software, onde será de responsabilidade do PO verificar.

2.5.1 Monitoramento de qualidade

Junto ao TDD (Test Driven Development) aplicado durante toda a confecção do código, este momento seria onde os devs discutiriam e testam o código a fim de encontrar não apenas problemas de implementação como funcionalidades que podem vir a dar problemas (correção preventiva) como supostos problemas de implantação, que podem ter se diferenciado do que os clientes esperavam de seu produto. Para isto que a implementação do SCRUM é interessante, quando estes forem descobertos, é possível de os aplicar como patches/correções em futuras versões do software, assim garantindo a satisfação da parte interessada e também garantindo a integridade e facilidade de manutenção do código em questão, que pode ser refatorado a fim de facilitar a legibilidade e a integração de novos componentes que serão incluídos no decorrer do desenvolvimento do projeto.

2.5.2 Levantamento de dados para as métricas

Como a sprint costuma ser bem rápida, é interessante fazer este processo ao seu fim, assim tendo os dados que serão utilizados na próxima sprint, não dando este trabalho para o PO, que já tem muitas outras responsabilidades.

A ideia é colocar tudo isto em um documento separado contando qual foi a tarefa, o que foi feito, quanto tempo isto levou e quantas pessoas estavam envolvidas nesta atividade.

2.5.3 Atualização da documentação do projeto

Possíveis implementações ou correções apontadas durante a sprint review devem ser documentadas e a atualização da documentação deve ser feita para evitar que algo fique de fora. Essas correções serão integradas a documentação que já está no github.

3. Conclusão

Essa foi uma experiência interessante, visto que, ao se criar uma empresa, mesmo que fictícia, nos mostra como essa área é ampla e os milhares de cuidados que devemos ter ao iniciar uma. Por exemplo, neste trabalho me foquei muito mais na parte administrativa do projeto, deixando um pouco de lado as partes de implementação, testes e afins, que por si já geram outras áreas de muito estudo e várias técnicas diferentes para que funcionem.

Acredito que por conta de não ter experiência utilizando estas ferramentas, posso ter cometido erros cronológicos, colocando tarefas em momentos embaralhados, mesmo tomando certo cuidado com as ordens.

Outro ponto que me preocupa são os cronogramas, que conhecendo o SCRUM, costumam ser apertados, geralmente sendo no período de um mês. Tendo isto em vista, todos os processos que estão contidos nos tópicos iniciais, relativos ao projeto em si, podem ser muito pesados, ainda mais quando, o product owner costuma ser uma única pessoa, nunca um time, o que pode trazer problemas relativos ao tempo de projetar o que estará contido em cada sprint.

Um último detalhe, por conta de o SCRUM ser um pouco abstrato, nem sempre todas as suas sprints ficam evidentes, no caso, temos 4 principais, porém, não há nenhuma referente ao estágio de desenvolvimento, por isto, não entrei em tantos detalhes do que se fazer neste momento, mas espera-se que pelo menos a documentação se mantenha atualizada, como por exemplo, os arquivos de interface, projeto de dados e afins.

3.1 Visão geral e Artefatos

Durante todo o processo de documentação do projeto, diversos itens são gerados, como por exemplo códigos fonte, documentos de requisitos, documentos legais, arquivos que contêm a interface do sistema prototipada e muitos outros. A frente estarei descrevendo os que identifiquei, deixando de maneira mais organizada o que será gerado. Lembrando que esta tabela não é definitiva, uma vez que mais coisas podem ser integradas enquanto outras podem ser retiradas, tudo dependerá do comportamento dos colaboradores quanto às sugestões que foram anteriormente descritas.

Tabela 11 - Artefatos gerados durante o ciclo de vida do software

Artefato	Descrição	Fonte
Áudio e transcrição	Gravações das reuniões com os clientes	Backlog do Produto
Documento de requisitos	Descreve quais são as funcionalidades que devem constar no software	Backlog do Produto
Documento de riscos	Descreve riscos, matriz de risco, riscos secundário, residuais e planejamento de resposta	Backlog do Produto
Casos de uso	Descreve os fluxos que cada ação do usuário gera, definindo os casos onde tudo ocorre bem	Backlog do Produto

	ou os desvios necessários caso haja incongruências	
Diagrama entidade relacionamento	Descreve o modelo do banco de dados, mostrando tabelas e relacionamento entre as mesmas	Backlog do Produto
Dicionário de dados	Este é em relação ao código fonte, visto que o banco de dados possui seu próprio (que é feito automaticamente em muitos casos)	Backlog do Produto
Projeto de interface	Demonstra a aparência de cada tela, a paleta de cores utilizada e a disposição dos botões na tela, evitando problemas de continuação entre as telas	Backlog do Produto
Arquivos referentes às métricas	Tanto o arquivo geral que possui todos os dados relativos às sprints anteriores, quanto o que descreve a atual. Usados para confeccionar as estimativas futuras	Backlog do Produto
Código fonte	Todos os arquivos criados, em processo de implementação, que descrevem ao computador como fazer determinadas tarefas	Desenvolvimento
Banco de dados	Este se refere quanto à estrutura do banco de dados, não apenas o que está contido de informações no mesmo	Desenvolvimento
Ferramentas utilizadas	Softwares que foram utilizados durante a criação do sistema,	Desenvolvimento

	contendo sua versão, prioritariamente, sendo assim, em eventuais problemas é possível voltar atrás e utilizar esta versão em casos de problema	
Arquivos de teste	Referentes ao TDD, descrevem rotinas que são ou foram utilizadas para se ter noção do que foi ou não testado no software	Desenvolvimento e Sprint Retro

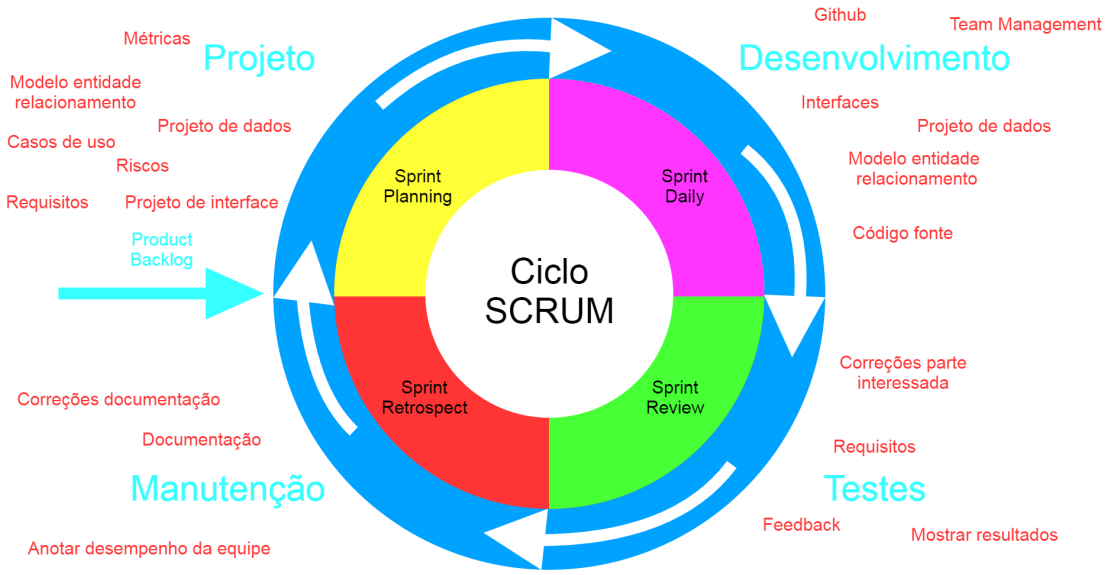
Fonte: Os autores

Para armazenar estes artefatos, o github será de grande valia, visto que ele dá a opção de armazenar todo e qualquer tipo de arquivo. A ideia será separar o que faz parte da implementação do que faz parte do projeto, assim fazendo dois repositórios. Um irá conter toda a documentação do projeto, relativa aos requisitos, interfaces e etc. Enquanto o outro terá o código em si, do qual armazenará a parte do desenvolvimento do mesmo. Desta forma, em ambos os casos é possível ter noção de alterações feitas em ambos, assim como, ter versionamento dois dois. Ao final da sprint, se gera um estado final do código e da documentação, como se fosse uma snapshot, do que foi feito naquele mês e que pode ser consultada ou revertida para ver se eventuais problemas que aconteceram, já existiam nestas versões.

3.2 Diagrama

Abaixo um diagrama que possui alguns passos, em ordem cronológica, pode ser encontrado. Este resume basicamente tudo o que foi visto anteriormente, de forma gráfica, assim facilitando a compreensão de como a documentação e gestão serão feitas na Technosight.

Figura 3 - Diagrama contendo o trajeto feito nas sprints



Fonte: os autores

3.3 O que faltou

- PERT-CPM

Esta é uma ferramenta muito interessante que possibilita estimar atrasos de entrega, assim como as “folgas” entre as tarefas. Por conta de não ter entrado tanto em detalhes quanto a parte do desenvolvimento, acabei por deixar esta de lado por hora.

- O que fazer quando um funcionário sair

Este é um tópico muito importante, visto que as vezes os colaboradores acabam por sair da empresa e querem levar o que produziu, ou até mesmo, podem ter algo contra a mesma e querer de certa forma se vingar por algo que aconteceu, assim acontecendo vazamentos e diversos outros problemas. Porém, como meu foco foi em “como trabalhamos” deixei um pouco de lado, por mais que tenha levantado este tópico neste momento.

- Definir exatamente como será a documentação no estágio de desenvolvimento na sprint

Este foi por conta de o próprio SCRUM não definir exatamente o período nem momento em que a sprint de desenvolvimento ocorrerá, então acabei por dar ênfase nas que já estão integradas oficialmente ao modelo.

- Tratar da viabilidade do projeto

Este é mais referente a tecnologia, onde pessoas possuem diversas ideias, às vezes malucas, onde se será possível fazer ou não, irá depender muito das capacidades da própria empresa no sentido de infraestrutura e mão de obra qualificada.

4. Referências

TURMASM. Padrões de Codificação no Desenvolvimento de Sistemas. **Micreiros**, 31 jul 2017. Disponível em: <https://micreiros.com/padroes-de-codificacao-no-desenvolvimento-de-sistemas/>. Acesso em: 23 maio 2022.

ABREU, André. Metodologia Scrum: o que é e como ela pode ajudar a sua empresa?. **Bossabox**, [s.d]. Disponível em: <https://blog.bossabox.com/especial/metodologia-scrum-o-que-e-e-como-ela-pode-ajudar-sua-empresa/>. Acesso em: 28 jun 2022.