FACULDADE ALFA DE UMUARAMA

CÉSAR AUGUSTO SURIANO BRASIL

Comunicação gestual entre pulseira e computador

Umuarama

2017

CÉSAR AUGUSTO SURIANO BRASIL

Comunicação gestual entre pulseira e computador

Projeto de pesquisa apresentado como requisito para aprovação na disciplina de Metodologia de Projetos na Faculdade Alfa de Umuarama.

Orientador: Prof. NOME DO PROFESSOR

Umuarama

2017

**SUMÁRIO**

[1) DADOS DE IDENTIFICAÇÃO DO PROJETO 4](#_Toc496916473)

[2) TEMA 4](#_Toc496916474)

[3) DELIMITAÇÃO DO TEMA 4](#_Toc496916475)

[4) FORMULAÇÃO DO PROBLEMA 4](#_Toc496916476)

[5) JUSTIFICATIVA 4](#_Toc496916477)

[6) OBJETIVOS 5](#_Toc496916478)

[6.1 OBJETIVO GERAL 5](#_Toc496916479)

[6.2 OBJETIVOS ESPECÍFICOS 6](#_Toc496916480)

[7) EMBASAMENTO TEÓRICO 6](#_Toc496916481)

[7.1 FUNDAMENTAÇÃO TEÓRICA 6](#_Toc496916482)

[7.1.1 Medologias Ágeis 6](#_Toc496916483)

[7.1.2 SCRUM 9](#_Toc496916484)

[7.1.3 Lean 11](#_Toc496916485)

[7.1.4 Kanban 13](#_Toc496916486)

[7.1.5 Extreme Programming (XP) 14](#_Toc496916487)

[7.2 REVISÃO BIBLIOGRÁFICA 16](#_Toc496916488)

[8) METODOLOGIA 16](#_Toc496916489)

[8.1 MÉTODO DE ABORDAGEM 16](#_Toc496916490)

[8.2 TÉCNICAS DE PESQUISA 17](#_Toc496916491)

[9) CRONOGRAMA 17](#_Toc496916492)

[10) REFERÊNCIAS 17](#_Toc496916493)

# DADOS DE IDENTIFICAÇÃO DO PROJETO

Aluno Autor: César Augusto Suriano Brasil

Professor Orientador: Nome do Professor

Linha de Pesquisa: Metodologias de Desenvolvimento de Software

# TEMA

Melhoria na usabilidade de dispositivos eletronicos através da comunicação por gestos.

# DELIMITAÇÃO DO TEMA

Desenvolvimento de um software que viabilize a comunicação entre uma pulseira e um computador através de gestos do usuário

# FORMULAÇÃO DO PROBLEMA

A comunicação através de gestos facilita a usabilidade dos usuários em comparação aos métodos tradicionais como mouse e teclado?

# JUSTIFICATIVA

A comunicação por gestos é um dos maiores símbolos de tecnologia futurista que se pode imaginar.

Porém, mesmo com métodos acessíveis para esse tipo de interação, nos dias atuais a comunicação por gestos não está sendo tão adotada em comparação com métodos já bem estabelecidos, como a utilização de teclado e mouse.

Por este motivo, foi definido como tema do projeto a comparação da usabilidade gestual com a convencional, para um maior aprofundamento nas qualidades e defeitos da interação gestual.

# OBJETIVOS

## OBJETIVO GERAL

* Analisar e compreender a usabilidade de diferentes usuários através de gestos.

## OBJETIVOS ESPECÍFICOS

* Estudar os métodos atuais de interação dos usuários
* Estudar as técnicas de metodologias ágeis disponíveis e as vantagens propostas dessas práticas.
* Incorporar técnicas ágeis no fluxo de trabalho tradicional.
* Propor a aplicação da metodologia híbrida em uma empresa do setor privado de desenvolvimento de software.

# EMBASAMENTO TEÓRICO

## FUNDAMENTAÇÃO TEÓRICA

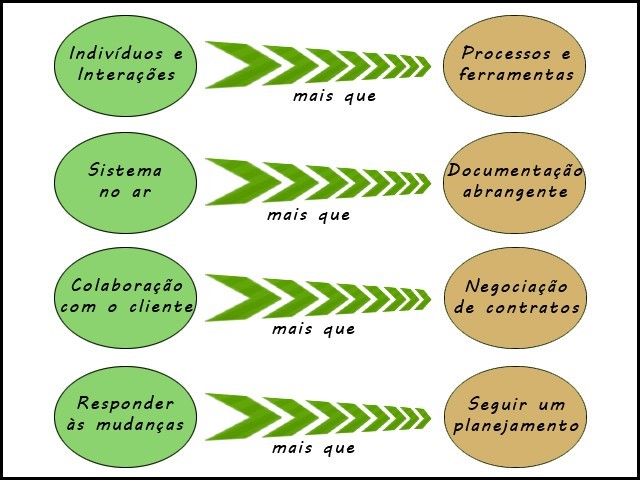
### Medologias Ágeis

STELLMAN e GREENE (2015) definem Ágil como um conjunto de métodos e metodologias que auxiliam um time a pensar de forma mais efetiva, a trabalhar de forma mais eficiente e a tomar decisões melhor. Esses métodos e metodologias se tratam não apenas do espectro da codificação do software, como também da análise das necessidades e do software, da arquitetura, do gerenciamento e da melhoria de processos definidos em processos tradicionais de desenvolvimento.

Junto dessas práticas ágeis, STELLMAN e GREENE ainda enfatizam que um ponto importante e crucial para que times ágeis tenham sucesso é a mudança de mentalidade da equipe. As pessoas envolvidas precisam compreender e viver as mudanças propostas, acreditar que estão agindo de forma a buscar uma melhoria substancial de resultados da empresa.

Por isso, no Manifesto para Desenvolvimento Ágil de *Software*, BECK et at (2001) definiram os valores que são necessários para uma equipe ágil conforme a Figura 1. Os valores tradicionais representados à direita são importantes e podem fazer parte dos processos ágeis, porém os valores à esquerda representam aqueles que são mais valorizados e buscados.

Indivíduos e interações mais que processos e ferramentas: As pessoas são a parte mais importante do desenvolvimento de sistemas. É nelas e no trabalho que desempenham que as metodologias ágeis focam. Processos e ferramentas estruturados são bons e podem auxiliar no gerenciamento dessas pessoas, mas não devem ser o centro das atenções dentro de um projeto, pois senão poderá resultar em um fluxo engessado, ineficiente e que não faz sentido para partes envolvidas.



*Figura 1: Valores do Desenvolvimento Ágil*

Sistema no ar mais que documentação abrangente: Ao contrário do que se acredita, as metodologias ágeis não pregam que não deve haver documentação nos projetos ágeis. Deve haver a documentação que for essencial para que a equipe e o cliente possam se comunicar. Porém, a quantidade e a profundidade desses documentos não são determinadas por processos tradicionais de desenvolvimento e análise, mas são negociados conforme a necessidade do projeto sendo desenvolvido. Ter um sistema em funcionamento agrega muito mais valor ao cliente do que intermináveis páginas de documentação que não serão utilizadas ou lidas.

Colaboração com o cliente mais que negociação de contratos: O cliente é aquele que melhor compreende quais os problemas que devem ser resolvidos, quais necessidades devem ser atendidas e quais as melhorias que podem ser implantadas no seu ambiente de trabalho. Trazê-lo para mais perto do time de desenvolvimento auxilia a equipe a não perder o foco do que deve ser entregue e do que é realmente importante para o produto final.

Responder às mudanças mais que seguir um planejamento: As necessidades do cliente evoluem conforme seu negócio prospera, e é comum que os requisitos de um sistema se alterem conforme o projeto acontece. Ser capaz de entender as necessidades em transformação do cliente durante o período de implementação do sistema é importante para o sucesso do projeto. Insistir em desenvolver um sistema baseado em um planejamento e escopo definidos previamente em contrato, que logo se torna obsoleto, eventualmente irá resultar em um sistema que não atende o cliente.

Além destes valores, foram também definidos 12 princípios no Manifesto Ágil:

* Nossa maior prioridade é satisfazer o cliente por meio da entrega contínua e adiantada de software com valor agregado.
* Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.
* Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
* Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
* Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
* Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
* O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
* Software funcionando é a medida primária de progresso.
* Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
* Contínua atenção à excelência técnica e bom design aumenta a agilidade.
* Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
* As melhores arquiteturas, requisitos e designs emergem de equipes auto organizáveis.
* Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Esses valores e princípios se tornaram a essência da mentalidade ágil, e são a base das metodologias: Scrum, Lean, Kanban, Extreme Programming.

### SCRUM

SUTHERLAND e SCHWABER (2016) definem Scrum como um framework com o qual pessoas podem resolver problemas adaptativos complexos, enquanto entrega produtiva e criativamente produtos do mais alto valor possível.

É um conjunto de regras, práticas, papéis e artefatos que tem sido usado para gerenciar o desenvolvimento de sistemas complexos desde os anos 90, que são leves e simples de entender, porém difíceis de dominar.

É um framework leve e simples pois consiste em regras bem definidas que podem ser aplicadas em vários tipos de ambientes e estágios de desenvolvimento. É difícil de dominar pois se baseia no controle empírico dos processos, de forma que os conhecimentos são obtidos e as decisões são realizadas com base em experiências vividas e eventos observados pelo time envolvido.

O Scrum tem 3 pilares de sustentação que o guiam:

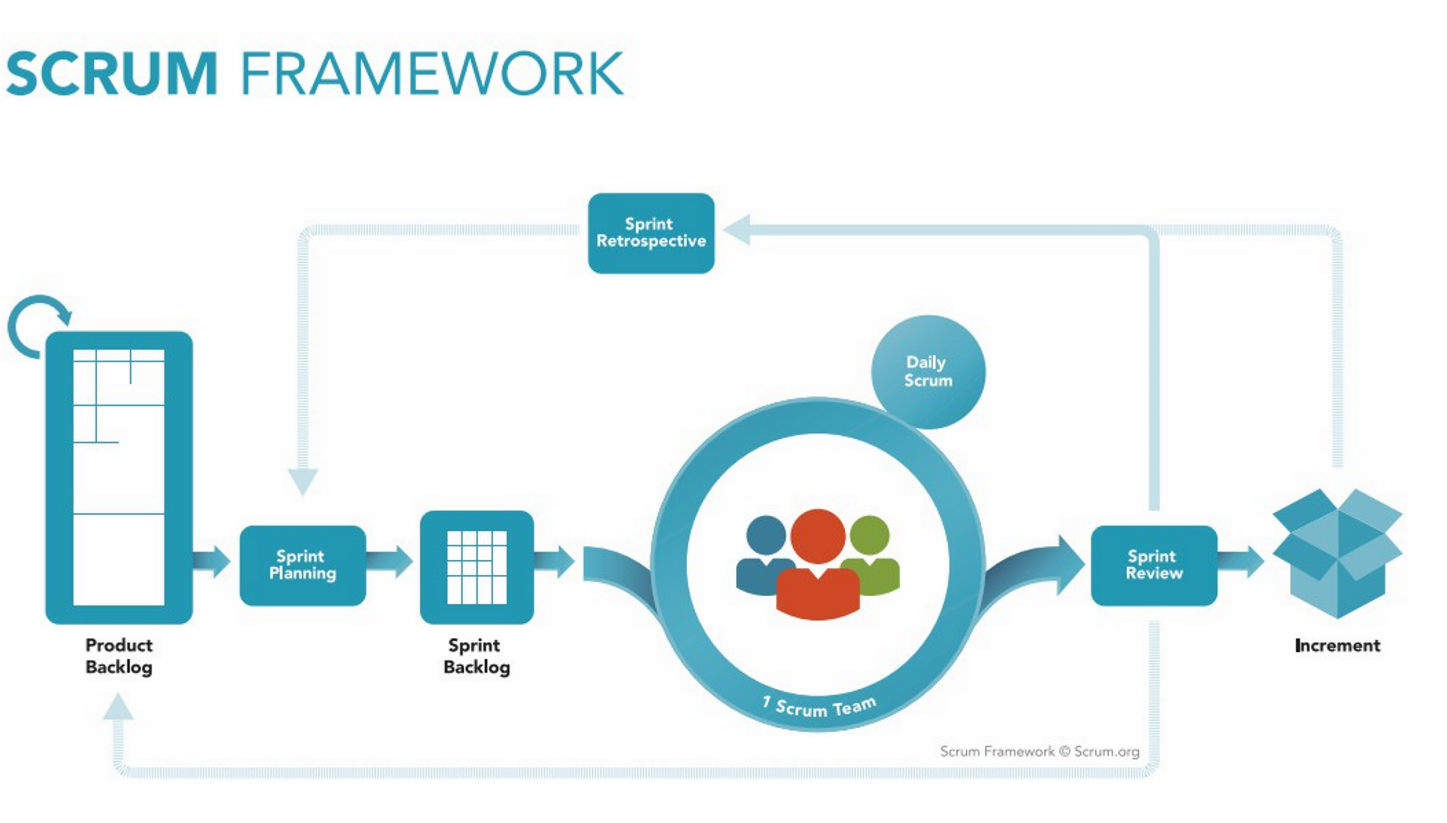
* Transparência: Todas as etapas e responsabilidades do processo devem ser visíveis por todos os envolvidos pelo produto. Todos devem ter os mesmos entendimentos dos conceitos, metas e entregas do projeto.
* Inspeção: Deve haver inspeções durante as etapas do processo Scrum para que não ocorram desvios das metas estabelecidas, porém essas inspeções não devem ser tão frequentes a ponto de prejudicar o andamento do time.
* Adaptação: Se durante o andamento do projeto forem identificados aspectos que façam o time se desviar dos objetivos desejados, ou que prejudicam a entrega do produto ao cliente, então o processo deve ser adaptado nas próximas iterações, de forma a minimizar futuros desvios.

Para que o Scrum seja utilizado com sucesso em um projeto é necessário que, além de práticas e regras, o time Scrum tenha algumas características que permeiam os pilares básicos para promover a confiança do time, que são: comprometimento com a entrega, coragem para questionar, foco nas metas, abertura para mudanças e respeito pelos envolvidos.

O time Scrum é bem definido em 3 papéis diferentes, no qual cada um tem responsabilidades próprias:

* *Product Owner:*é o 'dono do produto', e é sua responsabilidade maximizar o valor do produto a ser desenvolvido e o trabalho do time de desenvolvedores. Deve ser uma única pessoa responsável a qual determina o *Product Backlog*. O *Product Backlog* consiste em na lista de atividades do backlog (na qual cada atividade é chamada de estória), na qual o *Product Owner* deve ordenar e priorizar as atiividades, de forma a otimizar o trabalho dos desenvolvedores. O *Product Backlog* deve ser visível, transparente e claro para todos. A forma com que este trabalho é realizado varia de acordo com a necessidade de cada projeto, times e pessoas envolvidas.
* Time de desenvolvimento: é um pequeno grupo de profissionais que compõem o time de desenvolvimento Scrum. Deve ser um time pequeno e multidisciplinar, que será o responsável em desenvolver e entregar os incrementos do produto. São auto organizados, e não devem sofrer interferências externas nas decisões que tomam.
* *Scrum Master*: é o responsável por garantir que o time entendeu o processo Scrum, suas regras, e o que é esperado de cada papel. É um auxiliador, que deve garantir que os *stakeholders* tenham também entendido o propósito do Scrum, participem do processo conforme necessário, e que as cerimônias estabelecidas pelo Scrum ocorram nos tempos determinados. Estabelece e realiza as melhorias no processo quando são identificadas.

A forma com que esses papéis interagem e trabalham são definidos no processo Scrum por cerimônias e artefatos conforme a Figura 2. As cerimônias e artefatos serão estudados individualmente para que seja possível identificar práticas ágeis do SCRUM que possam ser incorporados ao processo tradicional de desenvolvimento.



*Figura 2: Framework SCRUM (SUTHERLAND & SCHWABER, 2016)*

### Lean

Conforme dito por STELLMAN e GREENE (2015), o Lean não é uma metodologia, mas sim uma mentalidade com valores e princípios próprios. Desta forma, ela pode ser aplicada em times que aplicam tanto processos tradicionais quanto processos ágeis, como o Scrum.

A origem do Lean vem da indústria automotiva japonesa, das fabricas da marca Toyota e do “Sistema de Produção Toyota”, ou Produção Lean. Conforme apontado por STOBER e HANSMANN (2010), há mais de 20 anos atrás as indústrias automotivas da Europa foram surpreendidas pela produção barata e de qualidade proveniente de fábricas da Asia que rapidamente superaram a manufatura do resto do mundo. As fábricas da Toyota adotaram a ideia que difunde a melhoria contínua de seus processos, o '*Kaizen*'. *Kaizen* é um termo japonês para difunfir a melhoria de atos e representa a dedicação à melhoria contínua em todos os aspectos da vida. E além dessa ideia, adotou uma série de práticas que tornaram as suas indústrias mundialmente competitivas. A essas práticas chamados de Produção Lean.

As principais práticas da Produção Lean são:

* Procurar a causa raiz da falha.
* Focar no valor do cliente e evitar qualquer desperdício.
* Descentralizar as responsabilidades.
* Concentre-se em trabalhar em equipe e em colaboração, em vez de dividir o trabalho.
* Melhoria contínua (Kaizen).
* Flexibilidade para reagir às mudanças nos requisitos do cliente
* Padronização dos processos.
* Planejamento e planejamento antecipado.
* Ferramentas simples e pragmáticas.

Em cima dessas práticas Lean, Mary and Tom Poppendieck (STOBER & HANSMANN, 2010) deram forma ao Desenvolvimento de Software Lean. Essa abordagem compreende as seguintes práticas:

* Elimine o desperdício.
* Concentre-se em aprender.
* Construa processos de qualidade.
* Postergue compromissos.
* Entregue rápido.
* Respeite as pessoas.
* Otimize o todo.

Essas práticas serão estudadas e detalhadas para que possam ser incorporados ao processo tradicional de desenvolvimento.

### Kanban

STELLMAN e GREENE (2015) definem Kanban como um processo de melhoria de processo utilizado por equipes ágeis. Não é uma metodologia de desenvolvimento, e não é do seu intento o uso exclusivo do Kanban em uma equipe ágil. O Kanban é desenhado para ser uma ferramenta de apoio às equipes que praticam outras metodologias ágeis, como Scrum, Lean ou Extreme Programming. ANDERSON (2010) descreveu que o Kanban não é uma metodologia de ciclo de vida de desenvolvimento de software ou uma abordagem de gerenciamento de projetos. Ele requer que algum processo já esteja sendo utilizado para que o Kanban possa ser aplicado para mudar incrementalmente o processo em andamento.

O foco do Kanban é auxiliar o time a melhorar a forma com que constroem um *software*, pois um time que utiliza o Kanban tem uma visualização melhor de quais ações são necessárias para se construir um software, como cada um interage com o resto da empresa, quais são os pontos de desperdício ou gargalo, e como melhorar o processo removendo a causa raiz desses pontos.

Os princípios fundamentais do Kanban são:

* Comece da forma que você está: observe o seu processo de trabalho atual, e considere-o como um conjunto de passos mutáveis e repetitivos. Identificar as regras e passos atuais dos times de desenvolvimento auxilia a estabelecer o fluxo de trabalho sendo aplicado e a enxergar pontos que podem ser melhorados.
* Concorde em buscar mudanças incrementais e evolutivas: o principal motivo de utilizar o Kanban é poder visualizar os pontos de melhoria do processo atual. Como as outras práticas ágeis, é necessário que as pessoas envolvidas estejam de acordo a pensar ágil e promover mudanças.
* Inicialmente, respeite os papéis, as responsabilidades e os cargos já existentes: As mudanças ocorrerão de forma gradativa e evolutiva, conforme ciclos de desenvolvimento forem realizados na empresa. Não é necessário implantar nenhuma alteração prévia ao Kanban nos times.

A partir desses princípios, ANDERSON (2010) determinou as 5 propriedades principais do Kanban para que um processo tenha sucesso:

* Visualizar o fluxo de trabalho.
* Limitar o trabalho em progresso (do inglês *Work in Progress*, WIP).
* Medir e gerenciar o fluxo.
* Use modelos para identificar oportunidades de melhorias.

A forma com que esses princípios são aplicados em um time de desenvolvimento para promover a melhoria contínua será estudada para que possa ser aplicado na metodologia proposta por este trabalho.

### Extreme Programming (XP)

Segundo BECK e ANDRES (2005), o XP é uma filosofia de desenvolvimento de software baseado nos valores de comunicação, feedback, simplicidade, coragem e respeito que propõe uma lista de práticas de desenvolvimento de software e um conjunto de princípios e técnicas complementares para lidar com problemas que possam ocorrer durante o processo. É um processo leve, escalável e adaptável focado na resolução de problemas e restrições do desenvolvimento de software.

O XP se difere de outras metodologias ágeis e tradicionais por algumas características descritas por BECK e ANDRES. Entre elas as mais importantes são:

* Seus ciclos de desenvolvimento são curtos e resultam em feedbacks contínuos, concretos e antecipados do que está sendo desenvolvido.
* Sua abordagem de planejamento incremental constrói um plano geral que evolui ao longo da vida do projeto.
* Possuir flexibilidade para agendar o desenvolvimento de funcionalidades, dando capacidade ao projeto de atender às solicitações de mudanças de escopo.
* A utilização de testes automatizados escritos por programadores, clientes e testadores para monitorar o progresso do desenvolvimento, que permite que o sistema evolua e esteja apto a capturar defeitos antecipadamente.

São várias as práticas e os princípios propostos pela metodologia XP, entre eles (BECK & ANDRES, 2005) (SCHMIDT, 2016):

* Sentar junto: estimular a comunicação entre as pessoas envolvidas com todos os sentidos envolvidos.
* Time inteiro: o time deve ter a consciência coletiva de responsabilidade e propriedade de código.
* Pair Programming: a programação de atividades em pares causa: dupla se mantém focada na atividade, as dúvidas e ideias são discutidas e esclarecidas rapidamente, auxílio rápido quando um dos programadores fica 'travado' evitando frustrações e desperdício de tempo.
* Integração contínua: programadores desenvolvem e testam atividades rapidamente, permitindo a integração contínua do código desenvolvido.
* Desenvolvimento orientado a testes: desenvolver primeiro testes que quebram, para após desenvolver o código da funcionalidade desejada. Essa prática auxilia a identificar problemas como código mal desenvolvido, alto acoplamento e baixa coesão de código; além de aumentar a confiança do sistema.
* Testes automatizados: a criação e manutenção de testes automatizados proporcionam um feedback contínuo do funcionamento do sistema em diferentes níveis de abstração de informações.
* Padrão de código: definição de regras e convenções de codificação aplicadas aos programadores de uma equipe. Estimular em estilo comum de programação auxilia a melhorar a legibilidade e a manutenção de um código de software.
* Revisão de Código: todo código finalizado é revisado por outro programador para a obtenção de feedback da atividade e garantir maior qualidade do código.
* Refatoração: reestruturar código já existente para melhorar a qualidade do código para aumentar a manutenibilidade e extensibilidade do sistema.

Será estudado neste trabalho a forma com que as práticas e princípios do XP podem ser incorporados em um processo tradicional de desenvolvimento de software.

## REVISÃO BIBLIOGRÁFICA

SOMMERVILLE (2011) e PRESSMAN (2015) definem os processos tradicionais de desenvolvimento de software estudados e utilizados como base da construção da metodologia híbrida de desenvolvimento: Modelo Cascata, Espiral e modelo em V.

# METODOLOGIA

## MÉTODO DE ABORDAGEM

Para este trabalho será considerada a abordagem Dedutivo.

Premissa maior: estabelecer processos de trabalho que sejam viáveis e conciliá-los com as entregas acordadas e aguardadas pelos clientes.

Premissa menor: a utilização de metodologia híbrida de desenvolvimento de software torna o processo de desenvolvimento viável.

Conclusão: estabelecer se a utilização de metodologias híbridas de software torna os processos de trabalho viáveis e dentro de datas acordadas de entrega com o cliente.

## TÉCNICAS DE PESQUISA

As técnicas que serão utilizadas neste trabalho são:

* Pesquisa bibliográfica: Estudo de metodologias tradicionais e ágeis de desenvolvimento de software.
* Pesquisa aplicada: Metodologias híbridas de desenvolvimento de software.
* Estudo de caso: aplicação de metodologia híbrida em uma empresa de software.

# CRONOGRAMA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ATIVIDADES** | **Nov** | **Dez** | **Jan** | **Fev** | **Mar** | **Abr** | **Mai** |
| Elaboração e entrega do projeto de pequisa |  |  |  |  |  |  |  |
| Fundamentação teórica |  |  |  |  |  |  |  |
| Estado da Arte |  |  |  |  |  |  |  |
| Proposta de Modelo Híbrido de Desenvolvimento |  |  |  |  |  |  |  |
| Aplicação de Modelo Híbrido |  |  |  |  |  |  |  |
| Análise dos Resultados |  |  |  |  |  |  |  |
| Desenvolvimento do Artigo |  |  |  |  |  |  |  |
| Correções e Ajustes |  |  |  |  |  |  |  |
| Apresentação |  |  |  |  |  |  |  |

# REFERÊNCIAS

ALAMI, Adam. **Why Do Information Technology Projects Fail?** Procedia

Computer Science 100. Melbourne : 2016. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050916322918> Acesso em 7 mai. 2017.

ANDERSON, David J. **Kanban**: successful evolutionary change for your technology business. Blue Hole Press, 2010.

BECK, Kent et al. **Manifesto for Agile Software Development**, 2001. Disponível em: <http://http://agilemanifesto.org/> Acesso em 07 mai. 2017.

BECK, Kent; ANDRES, Cynthia. **Extreme Programming Explained**: Embrace change. Addison Wesley Professional, 2004.

PRESSMAN, Roger S; MAXIM, Bruce R. **SOFTWARE ENGINEERING**: A

PRACTITIONER'S APPROACH. 8th Edition. New York: McGraw-Hill Edication, 2015.

SCHMIDT, Christoph. **Agile Software Development Teams:** The Impact of Agile Development on Team Performance**.** Progress on IS. Springer International Publishing, 2016.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide**. Scrum Alliance 21, 2016. Disponível em: <http://www.scrumguides.org/> Acesso em 20 mai. 2017.

SOMMERVILLE, IAN. **SOFTWARE ENGINEERING**. 9th Edition. USA: Pearson

Education, Inc., 2011.

STELLMAN, Andrew; GREENE, Jennifer. **Learning Agile**: Understanding Scrum, XP, Lean, and Kanban. 1st Edition. USA: O'Reilly Media, 2015.

STOBER, Thomas; HANSMANN, Uwe. **Agile Software Development:** Best Practices for Large Software Development Projects. Springer Berlin Heidelberg, 2010.