CENTRO ESTADUAL DE EDUCAÇÃO TECNOLOGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE SÃO PAULO

LUCAS EDUARDO ORMOND DE SOUSA

**DESENVOLVIMENTO DE SOFTWARE PARA GESTÃO DE PROJETOS EM SCRUM**

SÃO PAULO/SP

2022

LUCAS EDUARDO ORMOND DE SOUSA

**DESENVOLVIMENTO DE SOFTWARE PARA GESTÃO DE PROJETOS EM SCRUM**

Monografia apresentada a FATEC – São Paulo como parte dos requisitos para obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Aristides Novelli Filho.

SÃO PAULO/SP

2022

|  |
| --- |
|  |

LUCAS EDUARDO ORMOND DE SOUSA

**DESENVOLVIMENTO DE SOFTWARE PARA GESTÃO DE PROJETOS EM SCRUM**

Monografia apresentada a FATEC – São Paulo como parte dos requisitos para obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

|  |
| --- |
|  |

Prof. Dr. Aristides Novelli Filho

FATEC – São Paulo

Dedico este trabalho à minha família, que sempre me apoiou e esteve ao meu lado nos momentos mais difíceis, incentivando e renovando as minhas energias.

**AGRADECIMENTOS**

À Deus, por me guiar durante a minha jornada.

À minha família por todo o incentivo e apoio incondicional durante os meus estudos.

Aos meus amigos por todo o incentivo, e pela contribuição direta ou indiretamente à minha graduação.

Ao Prof. Dr. Aristides por ter aceitado ser meu orientador, e por todo o apoio durante o processo de elaboração do TCC.

À minha empresa atual, pela oportunidade de experiência prática no ciclo de desenvolvimento de software.

À Prof. M. Edmea pelas sugestões.

À instituição FATEC – SP juntamente com o CPS pela oportunidade de aprendizado e estudos.

A todos os outros professores da FATEC – SP pela experiência e conhecimento compartilhado durante o processo.

“Sentir é criar. Sentir é pensar sem ideias, e por isso sentir é compreender, visto que o Universo não tem ideias.”

(Fernando Pessoa, 1966)

**RESUMO**

Diferente do modelo em cascata, a metodologia ágil Scrum se baseia em diversos ciclos de atividades. Em cada ciclo, o projeto pode passar por alterações e ser aperfeiçoado de acordo com a interpretação obtida referente ao contexto e situação atual do projeto. O propósito deste trabalho foi analisar conceitos da engenharia de software juntamente com conceitos da metodologia ágil Scrum, para então iniciar o desenvolvimento de um sistema que permita auxiliar e gerir projetos que adotam essa metodologia em seu dia a dia. Foram abordadas as etapas necessárias que um software deve passar para ter êxito em seu processo de desenvolvimento, como especificação dos requisitos funcionais e não-funcionais do sistema, modelagem de software (diagramas de caso de uso, diagramas de classe, diagramas de evento, diagramas de atividade e diagramas de sequência), dentre outros tópicos importantes. Também foram abordados os tópicos principais da Scrum como os papéis da equipe (Product Owner e Scrum Master), Product Backlog, Sprint, dentre outras coisas.

**Palavras-chave:** Engenharia de software; metodologia; Scrum; desenvolvimento; projeto.

**LISTA DE FIGURAS**

FIGURA 1 – Modelo em cascata 18

FIGURA 2 – Modelo de prototipagem rápida 18

FIGURA 3 – Modelo em espiral 20

FIGURA 4 – Classificação de requisitos não funcionais 22

**LISTA DE QUADROS**

QUADRO 1 – Requisitos funcionais de um sistema mantedor de informações sobre pacientes em tratamento de saúde mental. 20

QUADRO 2 – Requisitos não funcionais de um sistema mantedor de informações sobre pacientes em tratamento de saúde mental. 23

**LISTA DE TABELAS**

**LISTA DE ABREVIATURAS E SIGLAS**

**SUMÁRIO**

[INTRODUÇÃO 14](#_Toc114428030)

[1 ENGENHARIA DE SOFTWARE 16](#_Toc114428031)

[1.1 Definição 16](#_Toc114428032)

[1.2 Ciclo de vida do software 17](#_Toc114428033)

[1.2.1 Modelo cascata 17](#_Toc114428034)

[1.2.2 Modelo de prototipagem rápida 18](#_Toc114428035)

[1.2.3 Modelo espiral 19](#_Toc114428036)

[1.3 Requisitos de sistema 20](#_Toc114428037)

[1.3.1 Requisitos funcionais 21](#_Toc114428038)

[1.3.2 Requisitos não funcionais 22](#_Toc114428039)

[1.4 Modelagem de software 23](#_Toc114428040)

# INTRODUÇÃO

**Contextualização**

O software está presente em diversas áreas, e por esse motivo sua existência é de extrema importância no mundo moderno. Grande parte de infraestruturas, serviços, indústrias, manufatura, dentre outras coisas, já funcionam de forma totalmente informatizada. Devido a esse fator, a engenharia de software se mostra fundamental para o funcionamento da sociedade (Sommerville, 2011).

Há diversos tipos de sistemas de softwares, e cada um possui a sua própria particularidade. Por esse motivo, apesar dessas aplicações precisarem de engenharia de software, não existem métodos ou técnicas universais a serem utilizadas. Portanto, é necessário observar a necessidade de cada sistema para abordá-lo de forma adequada (Id., 2011).

É importante ressaltar que a engenharia de software engloba diversos processos, não se limitando somente a desenvolver um simples programa de computador. Segundo Sommerville (2011, p. 3) “[...] quando falamos de engenharia de software, não se trata apenas do programa em si, mas de toda a documentação associada e dados de configurações necessários para fazer esse programa operar corretamente.”.

Tendo em vista esses conceitos e a importância da engenharia de software, ao desenvolver um sistema que consiga auxiliar na gestão de projetos em Scrum, ou qualquer outro sistema, é necessário levar em conta toda a estrutura e modelagem necessária para que ele consiga funcionar de maneira correta e consiga atender todas as expectativas.

**Objetivos Gerais**

Pesquisar sobre as etapas de desenvolvimento de software, modelagem de software, além de também entender a estrutura, funcionamento e organização da metodologia ágil Scrum.

**Objetivos Específicos**

Aplicar de forma prática todos os conhecimentos pesquisados, para modelar e estruturar um software que permita configurar e auxiliar projetos que adotam a metodologia ágil Scrum durante o seu ciclo de vida.

**Justificativa**

A proposta na presente pesquisa vai servir para aprofundar os conhecimentos pesquisados referentes a engenharia de software juntamente com a metodologia Scrum. Há ainda a aplicação prática desses conhecimentos, no qual será modelado um software capaz de auxiliar na gestão de projetos em Scrum, permitindo assim uma visão desses conceitos que vai além do estudo da parte teórica.

# 1 ENGENHARIA DE SOFTWARE

Este capítulo apresenta detalhes sobre os fundamentos da engenharia de software. Serão abordados tópicos sobre modelagem de software, envolvendo diagrama de classes, diagrama de caso de uso, diagrama de atividades e diagrama de eventos. Além disso, o capítulo também irá contemplar conceitos como especificação de requisitos funcionais e não-funcionais, ciclo de vida de software, dentre outros assuntos.

## 1.1 Definição

Segundo Stephen Schach (2009, p. 4) “[...] a engenharia de software é uma disciplina cujo objetivo é produzir software isento de falhas, entregue dentro do prazo e orçamento previstos, e que atenda às necessidades do cliente. Além disso, o software deve ser fácil de ser modificado quando as necessidades do usuário mudarem.”.

“Engenharia de Software trata da aplicação de abordagens sistemáticas, disciplinadas e quantificáveis para desenvolver, operar, manter e evoluir software. Ou seja, Engenharia de Software é a área da Computação que se preocupa em propor e aplicar princípios de engenharia na construção de software.” (Marco Túlio, 2020, p. 2).

“Engenharia de software é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado” (Sommerville, 2011, p. 5).

Diante das citações acima, conclui-se que a engenharia de software é uma disciplina extremamente ampla e que aborda diversos temas referente ao desenvolvimento de sistemas. Existe uma preocupação desde as etapas iniciais até a sua manutenção, sendo que o software deve cumprir com o que foi especificado, e estar livre de falhas que possam o comprometer. Em suma, a engenharia de software utiliza-se de técnicas e abordagens que permitem que o produto final, no caso o software, consiga atender às necessidades do cliente, considerando funcionalidade, prazo e orçamento previsto.

## 1.2 Ciclo de vida do software

O modelo de ciclo de vida do software é uma descrição das etapas que devem ser seguidas ao desenvolver um sistema. Por ser mais fácil realizar uma sequência de tarefas menores do que uma grande tarefa, o ciclo de vida é subdividido em uma série de etapas menores que são denominadas fases. Dependendo do modelo de ciclo de vida a ser adotado, o número de fases pode variar para mais ou para menos (Schach, 2009).

Diante dos conceitos apresentados, será possível observar que a codificação é feita somente após uma análise detalhada do escopo do software e requisitos do cliente. Todo o projeto é dividido em fases, e cada fase do projeto possui a sua própria responsabilidade. As fases podem sofrer variações dependendo do modelo a ser adotado.

### 1.2.1 Modelo cascata

Foi o primeiro modelo de ciclo de vida de software a ser publicado. No modelo em cascata, as fases são executadas em sequência. O estágio seguinte não deve ser iniciado até que a fase anterior seja concluída. Nesse modelo, as iterações podem ser dispendiosas envolver retrabalho. Por isso, somente é recomendável seu uso quando os requisitos são bem compreendidos e que não venham a ser alterados no futuro. É baseado nas seguintes etapas (Sommerville, 2011):

- Análise e definição dos requisitos. Obtido através da consulta aos usuários;

- Projeto de sistema e software, no qual os requisitos são alocados tanto para hardware quanto para software;

- Implementação e teste unitário, no qual o software é desenvolvido em unidades de programa, e cada unidade é testada para que os requisitos sejam atendidos;

- Integração e teste de sistema, onde as unidades individuais são integradas e testadas como um sistema completo;

- Operação e manutenção. O sistema é instalado colocado em uso e sofre manutenções.

A FIGURA 1 apresenta as principais fases e fluxo do funcionamento do modelo em cascata.

Diagrama

Descrição gerada automaticamente

FIGURA 1 – Modelo em cascata.

Fonte: Sommerville, 2011, pág. 20.

Conclui-se que o modelo em cascata é um modelo extremamente linear, e isso acaba por dificultar eventuais alterações nos requisitos inicialmente estabelecidos. Portanto, caso o modelo seja adotado pela equipe de desenvolvimento, é necessário definir da forma mais completa e correta possível as necessidades do software.

### 1.2.2 Modelo de prototipagem rápida

É baseado na entrega de um protótipo, ou seja, um modelo operacional que consiste em uma pequena parte funcional do software. Essa parte funcional não faz validações dos dados de entrada ou atualizações na base de dados, ela atua somente como um exemplo de funcionamento do produto final (Schach, 2009).

A FIGURA 2 apresenta as principais fases e fluxo do funcionamento do modelo de prototipagem rápida:

Diagrama

Descrição gerada automaticamente

FIGURA 2 – Modelo de prototipagem rápida.

Fonte: Stephen Schach, 2009, pág. 53.

A primeira etapa é criar um protótipo de forma rápida e deixar que o cliente interaja e avalie esse protótipo. Após a avaliação do cliente, os desenvolvedores podem ter a certeza de que o sistema irá atender todas as necessidades do cliente (Schach, 2009).

Devido as interações e validações postas pelo usuário, existe uma grande chance do documento de especificações resultantes esteja correto. Isso contribui para que as possíveis iterações não sejam mais necessárias. O fato de uma versão preliminar do software estar funcionando corretamente, acaba reduzindo as chances de uma correção durante ou após a implementação (Schach, 2009).

Conclui-se que o modelo de prototipagem rápida tem como pilar principal a entrega de uma versão minimalista do software para então começar todo o processo de desenvolvimento de fato. É feito de tal forma pois é mais fácil alinhar o que está sendo desenvolvido com as expectativas que o usuário possui, reduzindo a chance de eventual retrabalho que pode ser necessário.

### 1.2.3 Modelo espiral

O modelo espiral se baseia em diversas iterações, no qual cada iteração versões mais completas do software são construídas. Esse modelo considera os riscos e a avaliação do cliente em cada iteração. Isso ajuda a mitigar grandes problemas que podem surgir durante a execução do projeto. É definido pelas seguintes quatro atividades (Pressman, 1995):

- Planejamento, onde ocorre a determinação de objetivos, restrições e alternativas;

- Análise dos riscos, no qual ocorre a identificação dos riscos ou resolução dos riscos;

- Engenharia, onde ocorre o desenvolvimento do produto no “nível seguinte”;

- Avaliação feita pelo cliente, onde os resultados da engenharia são avaliados.

A FIGURA 3 apresenta as principais fases e fluxo do funcionamento do modelo em espiral. Nesse modelo, cada fase é representada através de um quadrante na figura. Por ser um modelo iterativo, é comum que uma mesma etapa seja executada mais de uma vez.

Diagrama, Desenho técnico

Descrição gerada automaticamente

FIGURA 3 – Modelo em espiral.

Fonte: Adaptado de Pressman, 1995, p. 39.

Pressman detalha o funcionamento do modelo em espiral da seguinte maneira:

O cliente avalia o trabalho de engenharia (o quadrante de avaliação do cliente) e apresenta sugestões para modificações. Baseada na entrada do cliente, ocorre a fase seguinte de planejamento e análise dos riscos. Em cada arco da espiral, a conclusão da análise dos riscos resulta numa decisão de “prosseguir/não prosseguir”. Se os riscos forem muito grandes, o projeto pode ser encerrado (Pressman, 1995, p. 40).

Conclui-se que o modelo em espiral é um modelo em que os requisitos do software são refinados em cada iteração realizada. Em cada uma dessas iterações, é de extrema importância avaliar os riscos existentes no projeto para obter sucesso no desenvolvimento. É um modelo onde a opinião do cliente é considerada em cada ciclo, ajudando a reduzir eventuais problemas que podem surgir no processo.

## 1.3 Requisitos de sistema

Os requisitos de um sistema são todas as descrições das funcionalidades, dos serviços oferecidos e das restrições referente ao funcionamento do software. Os requisitos dizem a respeito sobre a necessidade e expectativas que o cliente possui relativo funcionamento do software. O processo de verificar essas necessidades e expectativas é chamado de engenharia de requisitos (Sommerville, 2011).

### 1.3.1 Requisitos funcionais

Os requisitos funcionais de um sistema servem para definir o que o sistema deve de fato fazer no que se diz a respeito as funcionalidades e serviços implementados (Marco Túlio, 2020).

Sommerville detalha a definição de requisitos funcionais da seguinte maneira:

Os requisitos funcionais de um sistema descrevem o que ele deve fazer. Eles dependem do tipo de software a ser desenvolvido, de quem são seus possíveis usuários e da abordagem geral adotada pela organização ao escrever os requisitos. Quando expressos como requisitos de usuário, os requisitos funcionais são normalmente descritos de forma abstrata, para serem compreendidos pelos usuários do sistema. No entanto, requisitos de sistema funcionais mais específicos descrevem em detalhes as funções do sistema, suas entradas e saídas, exceções etc. (Sommerville, 2011, p. 59).

O QUADRO 1 apresenta os requisitos funcionais de um sistema utilizado para manter informações sobre os pacientes em tratamento por problemas de saúde mental.

QUADRO 1 – Requisitos funcionais de um sistema mantedor de informações sobre pacientes em tratamento de saúde mental.

|  |  |
| --- | --- |
| 1 | Um usuário deve ser capaz de pesquisar as listas de agendamentos para todas as clínicas. |
| 2 | O sistema deve gerar a cada dia, para cada clínica, a lista dos pacientes para as consultas daquele dia. |
| 3 | Cada membro da equipe que usa o sistema deve ser identificado apenas por seu número de oito dígitos. |

Fonte: Adaptado de Sommerville, 2011, pág. 59.

Em suma, os requisitos funcionais limitam-se ao que o software deve de fato fazer, podendo serem escritos de forma abstrata ou específica, detalhando as entradas, saídas e eventuais erros que são capazes de ocorrer. Os fatores determinantes desses requisitos levam em conta o tipo de sistema a ser desenvolvido, além dos seus possíveis usuários.

### 1.3.2 Requisitos não funcionais

Os requisitos não funcionais são requisitos que não possuem ligação direta com os serviços específicos prestados pelo software ao cliente. Geralmente estão relacionados a propriedades como confiabilidade, tempo de resposta, segurança, entre outros (Sommerville, 2011).

Requisitos não funcionais tendem a serem mais críticos que requisitos funcionais individuais. Sommerville explica esse fator da seguinte forma:

Os usuários do sistema podem, geralmente, encontrar maneiras de contornar uma função do sistema que realmente não atenda a suas necessidades. No entanto, deixar de atender a um requisito não funcional pode significar a inutilização de todo o sistema. Por exemplo, se um sistema de aeronaves não cumprir seus requisitos de confiabilidade, não será certificado como um sistema seguro para operar; se um sistema de controle embutido não atender aos requisitos de desempenho, as funções de controle não funcionarão corretamente (Sommerville, 2011, p. 60).

O surgimento desses requisitos se dá através das necessidades dos usuários, devido a restrições de orçamento, políticas organizacionais, fatores externos, entre outros (Sommerville, 2011). A FIGURA 4 classifica os requisitos não funcionais existentes.

Diagrama

Descrição gerada automaticamente

FIGURA 4 – Classificação de requisitos não funcionais.

Fonte: Sommerville, 2011, p. 61.

Tais requisitos são definidos da seguinte maneira (Sommerville, 2011):

- Requisitos de produto são requisitos referentes ao comportamento do software, como por exemplo velocidade em requisitos de desempenho, e espaço em disco em requisitos de espaço.

- Requisitos organizacionais são requisitos decorrentes das políticas e procedimentos da organização do cliente e desenvolvedor, como por exemplo como o software deverá ser usado em requisitos operacionais, e a linguagem de programação a ser utilizada em requisitos de desenvolvimento.

- Requisitos externos são requisitos decorrentes de fatores externos ao sistema e seu processo de desenvolvimento, como por exemplo um software operante dentro da lei em requisitos legais, e o que deve ser feito para ser aceito por um regulador como o banco central em requisitos reguladores.

O QUADRO 2 apresenta os requisitos não funcionais de um sistema utilizado para manter informações sobre os pacientes em tratamento por problemas de saúde mental.

QUADRO 2 – Requisitos não funcionais de um sistema mantedor de informações sobre pacientes em tratamento de saúde mental.

|  |  |
| --- | --- |
| **Requisito de produto** | O MHC-PMS deve estar disponível para todas as clínicas durante as horas normais de trabalho (segunda a sexta-feira, 8h30 às 17h30). Períodos de não operação dentro do horário normal de trabalho não podem exceder cinco segundos em um dia. |
| **Requisito organizacional** | Usuários do sistema MHC-PMS devem se autenticar com seus cartões de identificação da autoridade da saúde. |
| **Requisito externo** | O sistema deve implementar as disposições de privacidade dos pacientes, tal como estabelecido no HStan-03-2006-priv. |

Fonte: Adaptado de Sommerville, 2011, pág. 62.

Diante dos conceitos apresentados, é possível concluir que os requisitos não funcionais não estão relacionados de forma direta com o que o software deve realizar. Em suma, os requisitos não funcionais acabam se preocupando com a forma de implementação, restrições de projeto, qualidade, entre outros tópicos. Tais requisitos costumam ser mais críticos que requisitos funcionais, podendo até inviabilizar um sistema por completo.

## 1.4 Modelagem de software