



LABORATÓRIO DE SOFTWARE E PROJETOS

PROJETO DE DESENVOLVIMENTO DE SOFTWARE

Autor: Me. Rodrigo Ramos

Revisor: Felipe Oviedo Frosi

INICIAR



introdução

Introdução

Prezado(a) aluno(a), seja bem-vindo(a) à unidade de Projeto de Software. Durante esta unidade iremos estudar os principais elementos que compõem um software em sua etapa de projeto.

Assim como a construção de um empreendimento imobiliário, a construção física de um software não é sua primeira etapa. Do mesmo modo que para construção imobiliária existem a planta e a maquete, a construção de um software é compreendida de diversas etapas.

A primeira etapa é compreensão da ideia geral de um sistema. Geralmente essa etapa é uma conversa com um cliente. Como sendo a primeira etapa, deve ser documentada, permitindo que os próximos passos possam ser realizados. Essa conversa em engenharia de software é denominada entrevista. A partir da entrevista, um projeto de software será transformado em requisitos, casos de uso e todo um conjunto de elementos que favorecem a gestão do projeto de software, bem como o desenvolvimento software em si.

Compreendendo o Sistema - Entrevista

O primeiro momento do desenvolvimento de um sistema é voltado para compreender o que será desenvolvido. Para isso, é realizada uma conversa documental com a pessoa que domina o negócio sob o qual será desenvolvido o sistema. Essa pessoa é denominada stakeholder.

Por muitas vezes chamamos de stakeholder os clientes, o que não está errado, no entanto um stakeholder é todo usuário que tem pleno domínio sobre a regra de negócio em que o sistema será desenvolvido. Um stakeholder pode ser compreendido como parte interessada no projeto, poderá ser um ou diversos, dependendo do escopo e do projeto a ser desenvolvido.

Durante a construção de um sistema, as entrevistas são realizadas com os stakeholders, permitindo que a equipe de desenvolvimento de software possa compreender melhor sobre o que será desenvolvido.

Segundo Sommerville (2011, p. 86), as entrevistas podem ser de dois tipos:

- Entrevistas fechadas, em que o stakeholder responde a um conjunto

predefinido de perguntas.

- Entrevistas abertas, em que não existe uma agenda predefinida. A equipe de engenharia de requisitos explora uma série de questões com os stakeholders do sistema e, assim, desenvolve uma melhor compreensão de suas necessidades.

A entrevista compreende um importante mecanismo de coleta de informações com o stakeholder. Essa etapa deve ser desenvolvida com cuidado, pois impacta nas demais etapas do ciclo de vida de software.

A Entrevista na Gestão de Projetos

Você pode perceber que a entrevista é um importante mecanismo durante o desenvolvimento de um projeto de software. Na sequência você verá um conjunto de mecanismos que permitem transformar a entrevista em um produto (software), utilizando para isso ferramentas de gestão de projeto, principalmente Scrum, integrada com ferramentas da engenharia de software (levantamento de requisitos, casos de uso, UML).

Do ponto de vista do desenvolvimento ágil de software, o agente responsável por realizar a entrevista será o Product Owner (Dono do Produto).

Na metodologia ágil SCRUM, o Product Owner corresponde ao primeiro elemento do ciclo de vida de um projeto de software. Será a partir do conhecimento desse agente sobre o produto que todo o projeto terá continuidade.

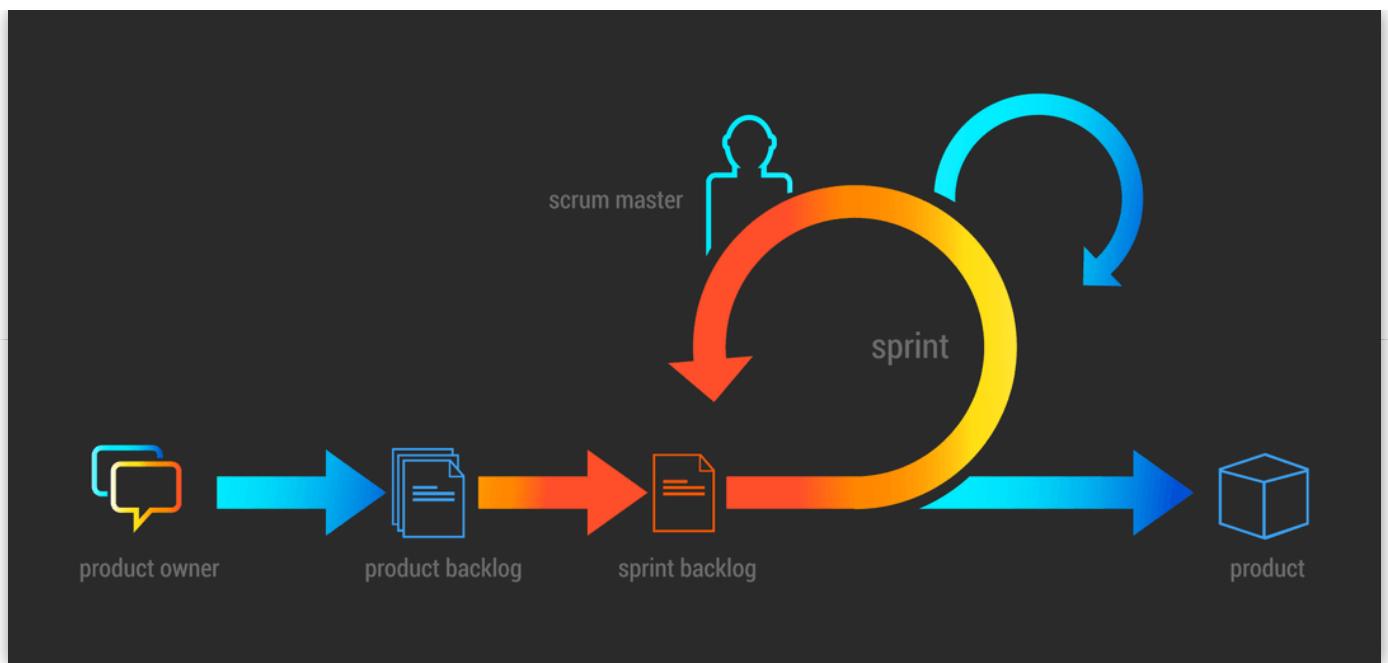


Figura 1.1 - Fluxo da Metodologia SCRUM

Fonte: Bakhtiar Zein / 123RF.

#PraCegoVer: Imagem reflete o fluxo da metodologia SCRUM, começando o fluxo com o product owner, responsável por definir as funcionalidades do produto e priorizar os itens do product backlog. O fluxo segue para o próximo passo, definição do product backlog, representa uma lista de funcionalidades desejadas para o produto. O fluxo vai para o próximo passo que é definido como sprint backlog, representada por uma lista de atividades que precisam ser realizadas durante a sprint. O fluxo entra em um ciclo repetitivo de sprint que a cada ciclo vai recebendo incrementos e conta com a gestão do scrum master até que as entregas aconteçam através da saída dos produtos.

Durante a aplicação do Scrum, a entrevista irá se transformar em Product Backlog, uma lista de ferramentas desejadas que fará com que o fluxo desta metodologia ágil funcione. Por isso, se torna importante a documentação da entrevista nesse momento, explanada a seguir.

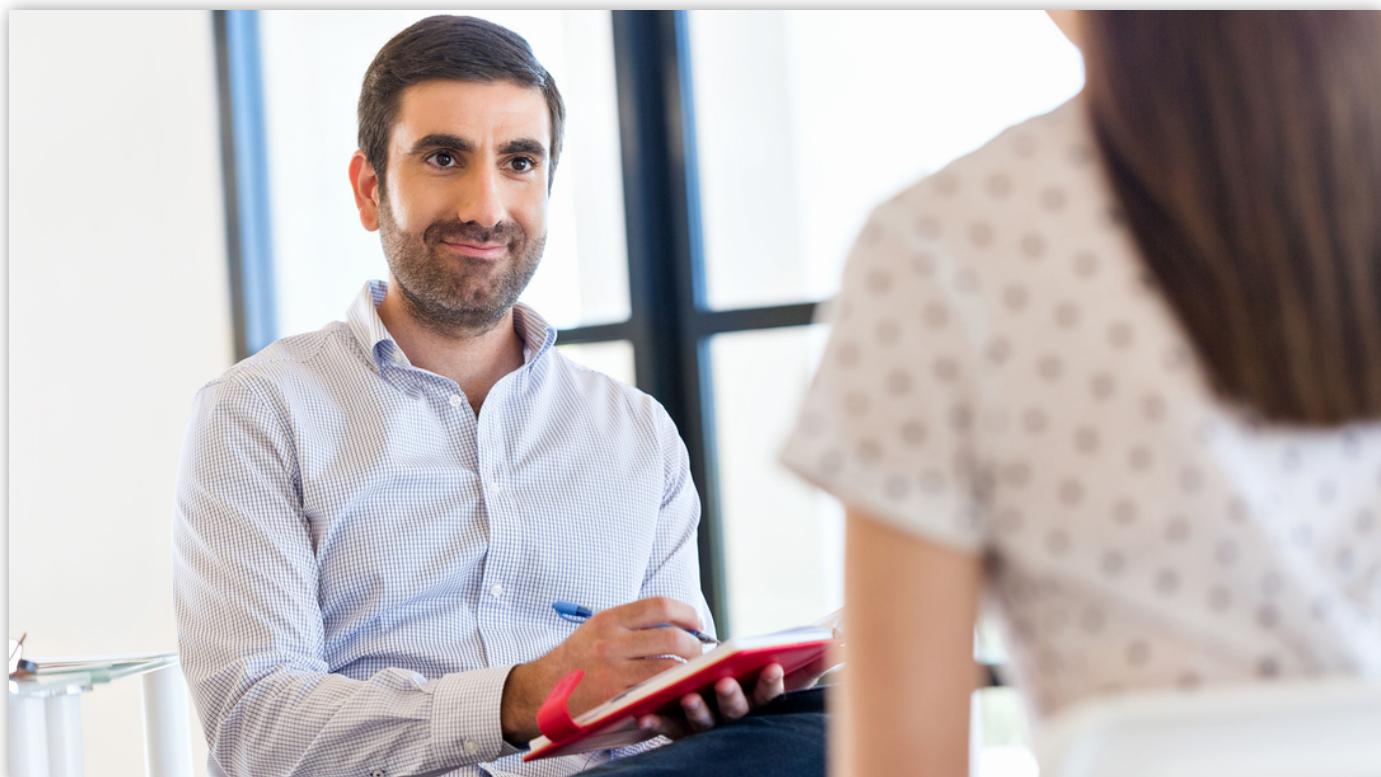
Documentação de uma Entrevista

Durante a realização de uma entrevista, é necessário desenvolver a documentação necessária. A documentação de uma entrevista pode

acontecer de maneira simultânea ao seu acontecimento ou após ser finalizada.

Quando acontece de maneira simultânea, estamos falando que ao mesmo tempo que a entrevista acontece é elaborado um documento, tradicionalmente esse documento é denominado briefing.

O briefing é um documento que reúne informações gerais do que o cliente deseja ter no sistema e que foram discutidas durante a entrevista. A partir do briefing, outros documentos de software podem ser elaborados.



*Figura 1.2 - Entrevista
Fonte: Sergey Nivens / 123RF.*

#PraCegoVer: Imagem reflete um entrevistador em plena entrevista para montagem de um briefing, podendo ser realizada com qualquer um dos interessados ou envolvidos. A entrevista na imagem é registrada através de formulários em papel, mas pode ser através de áudio ou vídeo.

Temos um caso em que o briefing pode ser realizado após a entrevista com o stakeholder, quando ele é empregado no uso de tecnologias que permitam gravar o que foi conversado: áudio ou vídeo.

Vejamos um exemplo de um briefing realizado com um proprietário de uma escola:

"Senhor Valdeci é proprietário de uma faculdade de ensino de computação e necessita de um sistema para fazer a gestão acadêmica dos seus professores e alunos. O principal objetivo do sistema é gerenciar notas e presença dos alunos, respectivamente calcular as horas ministradas pelos professores. Existem 4 cursos de graduação: Ciência da Computação, Engenharia da Computação, Tecnologia em Sistemas para Internet e Design de Jogos Digitais. Os cursos existentes compartilham disciplinas, bem como os docentes. Os alunos se matriculam diretamente nas disciplinas, que podem ser ministradas por um ou vários professores. Como a instituição tem plano de formação os professores também podem ser alunos e cursar disciplinas. Os coordenadores de curso são sempre professores e são responsáveis por supervisionar docentes. O sistema deve permitir com que tanto alunos quanto professores e coordenadores tenham acesso via Web e celular. No entanto, cada tipo de usuário terá um tipo de acesso, alunos poderão fazer apenas a matrícula e consultar nota e frequência".

Até o momento você aprendeu que a entrevista é a maneira formal de chamar a conversa com o cliente, denominado stakeholder. A entrevista é documentada na forma de um Briefing, documento este que impactará em todo o ciclo de vida do software.

Requisitos - Formalização de um Briefing

O Briefing documenta o que foi conversado durante uma entrevista, no entanto muitas vezes é um documento único e precisa de um grau de formalização para o nível de gestão e projeto de software.

Durante o projeto de software, o primeiro documento formal sobre o sistema a ser desenvolvido denomina-se requisito de software. Na engenharia de software, esse processo de criar requisitos é denominado levantamento de requisitos.

Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, colocar um pedido ou encontrar informações. O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado engenharia de requisitos (RE, do inglês requirements engineering). O termo 'requisito' não é usado de forma consistente pela indústria de software. Em alguns casos, o requisito é apenas uma declaração abstrata em alto nível de um serviço que o sistema deve oferecer ou uma restrição a um sistema. No outro extremo, é uma definição detalhada e formal de uma função do sistema (SOMMERVILLE, 2011, p. 57).

Os requisitos de um sistema são divididos em dois tipos: requisitos funcionais e requisitos não funcionais. Os requisitos funcionais estão relacionados a funcionalidade que um sistema deve ter, ou seja, são as funcionalidades que farão parte do sistema. Em um sistema acadêmico são exemplos de caso de uso “cadastrar aluno” ou “o sistema deve permitir fazer o cadastro de alunos”.

Por sua vez, os requisitos não funcionais são aqueles que não se remetem a funcionalidades existentes do sistema. Isso significa que tais requisitos envolvem fatores externos ao sistema. No mesmo exemplo de um sistema bancário, são requisitos não funcionais “o sistema deverá ser desenvolvido com interface Web” ou “o sistema deverá possibilitar sua execução em celulares com sistema operacional Android”.

Saiba mais

Saiba mais

Os requisitos funcionais são um fator determinante para a etapa de projeto de software, bem como tem um papel importante para que o software final seja desenvolvido de acordo com as necessidades do cliente. Todo requisito funcional para ser considerado bom tem como características: unidade, completude, consistência, atomicidade, não ambiguidade, ser verificável, rastreável e ter prioridade. Para que você possa ver um exemplo do desenvolvimento de requisitos.

Fonte: Elaborado pelo autor.

[ACESSAR](#)

Tendo em vista compreender melhor o desenvolvimento dos requisitos funcionais, considerando o Briefing realizado no tópico 1.2, o Quadro 1.1 mostra alguns requisitos funcionais criados a partir do mesmo.

Os requisitos funcionais iniciam direto com um verbo ou com “O sistema” em menção ao software que será desenvolvido. No exemplo, são utilizadas ambas as abordagens. No entanto, em um cenário real, deve-se utilizar apenas uma destas.

Código	Requisito
#RF 01	O sistema deve permitir o cadastro de aluno.
#RF 02	O sistema deve permitir realizar relatório dos alunos cadastrados.
#RF 03	O sistema deve permitir que o professor salve a frequência dos alunos.
#RF 04	O sistema deve listar as disciplinas e os alunos matriculados.
#RF 05	O sistema deve mostrar relatório de aulas ministradas pelo professor.
#RF 06	Cadastrar usuário professor.
#RF 07	Listar usuário professor.
#RF 08	Login do usuário professor.
#RF 09	Editar dados do usuário professor.
#RF 10	Cadastrar usuário professor como coordenador.

Quadro 1.1 - Requisitos Funcionais

Fonte: Elaborado pelo autor.

Para compreender melhor a diferença entre os requisitos funcionais e não funcionais, será utilizado o Briefing realizado no tópico 1.2 para criar os exemplos de requisitos não funcionais que são mostrados pelo Quadro 1.2.

Código	Requisito
#RNF 01	O sistema deve ser executado em celulares Andoid e IoS.
#RNF 02	O sistema deve ser responsivo.
#RNF 03	O sistema deve estar disponível on-line.
#RNF 04	O sistema deve ser acessível a deficientes visuais.
#RNF 05	O sistema deve garantir a segurança dos dados armazenados.
#RNF 06	O sistema deve permitir que usuários façam login com redes sociais.

Quadro 1.2 - Requisitos Não Funcionais

Fonte: Elaborado pelo autor.

Como os requisitos não funcionais envolvem diversos fatores externos ao sistema em si, uma dica para compreender quais elementos podem compor os requisitos não funcionais é utilizar o fluxograma proposto por Sommerville (2011) com os tipos de requisitos não funcionais, conforme a Figura 1.3.

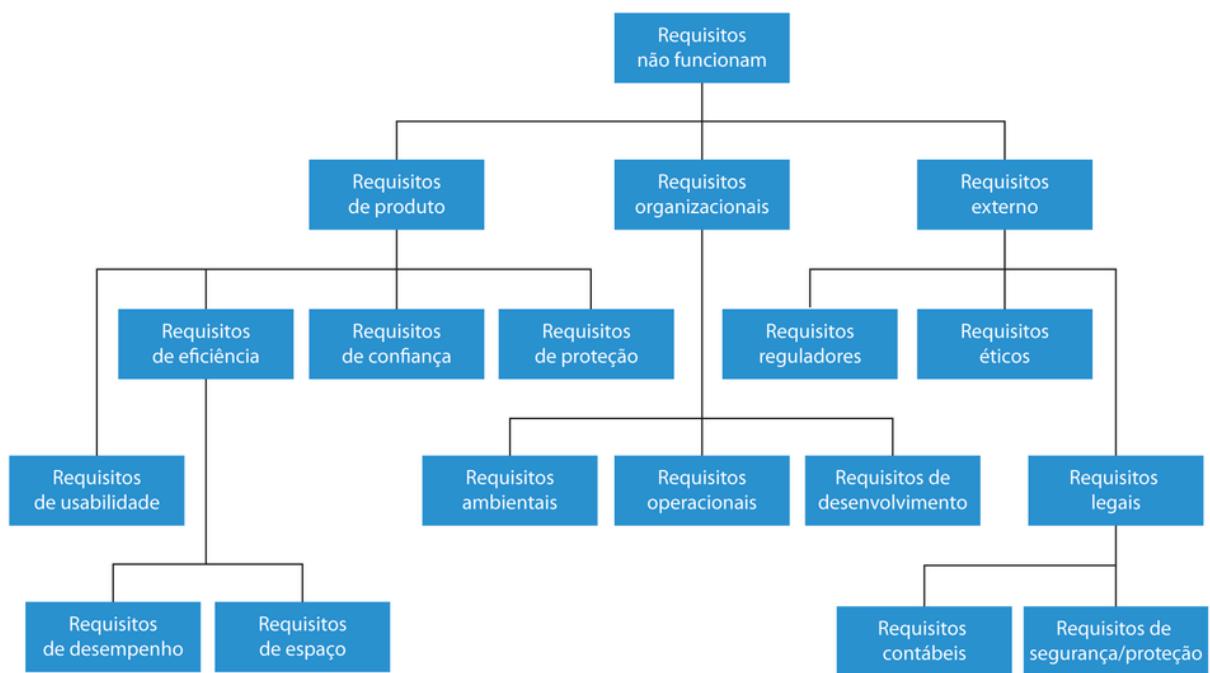


Figura 1.3 - Tipos de Requisitos não funcionais

Fonte: Sommerville (2011, p. 75).

#PraCegoVer: Imagem reflete um organograma classificador de requisitos não funcionais. Este organograma foi idealizado pelo autor Sommerville e é dividido em três grandes grupos de requisitos não funcionais, os requisitos de produto, os requisitos organizacionais e os requisitos externos. Os requisitos de produto possuem quatro grandes áreas, os requisitos de usabilidade, os requisitos de confiança, os requisitos de proteção e os requisitos de eficiência que é detalhado em requisitos de desempenho e requisitos de espaço. Os requisitos organizacionais são detalhados em requisitos ambientais, requisitos operacionais e requisitos de desenvolvimento. Por fim, os requisitos externos são divididos e, requisitos reguladores, requisitos éticos e requisitos legais que é detalhado em requisitos contábeis e requisitos de segurança/proteção.

Segundo Mello (2010, p. 2),

[...] o levantamento de requisitos é uma das partes mais importantes do processo que resultará no desenvolvimento de um sistema. Entender aquilo que o cliente deseja ou o que o cliente acredita que precisa e as regras do negócio ou processos do

negócio.

Ou seja, os requisitos são a primeira etapa da documentação de um software. Ainda que exista o Briefing anterior aos requisitos, os requisitos serão utilizados para evoluir com a documentação do sistema, bem como apoio à distribuição de tarefas na gestão de projetos.

praticar

Vamos Praticar

Tendo como objetivo complementar o Briefing solicitado anteriormente simule uma entrevista com o proprietário da faculdade. Essa entrevista deve ser documentada com:

- a. Briefing;
- b. Requisitos funcionais;
- c. Requisitos não-funcionais.



Casos de Uso - Modelagem de negócios



Dando continuidade ao processo de desenvolvimento de um projeto de software você irá evoluir seus conhecimentos através do conceito de modelagem conceitual de software.

A modelagem conceitual de software é uma atividade não trivial que se relaciona com vários desafios, como: dificuldades na elicitação de requisitos, problemas de comunicação entre o analista e o usuário, uso de documentos de requisitos que nem sempre expressam claramente as funcionalidades do sistema (OLIVEIRA, 2017, p. 11).

Apesar de que em sua definição todos os elementos de representação de um software fazem parte da Modelagem Conceitual de Software, o termo tem maior popularidade pelos diagramas que dão suporte ao desenvolvimento de software, principalmente os diagramas UML.

A UML (*Unified Modeling Language*), linguagem de modelagem unificada, foi desenvolvida nos meados dos anos de 1990 por Grady Booch, James

Rumbaugh e Ivar Jacobson, ambos autores de importantes métodos orientados a objetos. A sua obra “UML: Guia do Usuário” tem sido utilizada como referência na área de modelagem de software desde então.

De modo geral a UML fornece um conjunto de diagramas que permite realizar a modelagem de um sistema de informação, desde a perspectiva do usuário, passando por detalhes da implementação, bem como diagramas que refletem as classes que serão implementadas.

A modelagem permite a compreensão de um sistema. Nenhum modelo é inteiramente suficiente. Sempre serão necessários vários modelos, conectados entre si, para tornar possível entender qualquer aspecto, ainda que seja o sistema mais trivial. No caso de sistemas que fazem uso intenso de software, torna-se necessária uma linguagem capaz de abranger as diferentes visões relacionadas à arquitetura do sistema, como essa arquitetura evolui ao longo do ciclo de vida de desenvolvimento do software (BOOCH; RUMBAUGH; JACOBSON, 2012, p. 48).

Durante a utilização de modelagem utilizando UML integrado ao ciclo de vida de um software, uma vez desenvolvidos os requisitos, os casos dão continuidade ao ciclo de vida de um projeto de software. Os casos de uso fazem parte da modelagem de negócios, uma etapa do ciclo de vida do software para compreender melhor as regras e como funciona o negócio.

Os casos de uso são documentados por um diagrama de casos de uso de alto nível. O conjunto de casos de uso representa todas as possíveis interações que serão descritas nos requisitos de sistema. Atores, que podem ser pessoas ou outros sistemas, são representados como figuras ‘palito’. Cada classe de interação é representada por uma elipse. Linhas fazem a ligação entre os atores e a interação. Opcionalmente, pontas de flechas podem ser adicionadas às linhas para mostrar como a interação se inicia. Essa situação é ilustrada na Figura 4.6, que mostra alguns dos casos de uso para o sistema de informações de pacientes (SOMMERVILLE, 2011, p. 75).

Os casos de uso fazem parte da modelagem de negócios, uma etapa do ciclo de vida do software que serve para compreender melhor o funcionamento e as regras de negócio. De modo geral, quando se fala de caso de uso, existe o diagrama de caso de uso e a descrição de caso de uso. Nas próximas seções, abordaremos cada um deles com mais detalhes.

Diagrama de Casos de Uso

Os diagramas de caso de uso nada mais são que as representações gráficas de uma visão geral de como o sistema irá funcionar de acordo com a visão de um stakeholder.

Saiba mais
Saiba mais

Durante esta seção estamos trabalhando o conceito de diagrama de casos de uso. Existem diversas ferramentas para elaborar esse tipo de diagrama. Como sugestão fica neste livro a opção do uso do StarUML, que é uma ferramenta completa muito útil no desenvolvimento de diagramas de caso de uso, bem como outros diagramas.

Fonte: Elaborado pelo autor.

[ACESSAR](#)

Segundo Malucelli *et al.* (2010), diagramas de caso de uso são considerados fundamentais na modelagem de sistemas, pois neles são identificadas as interações necessárias, descritas no levantamento de requisitos. Durante o desenvolvimento dos casos de uso, os usuários, equipamentos, banco de dados, ou mesmo outros sistemas que possam interagir com o sistema a ser desenvolvido, são representados por atores (stickman - bonequinho de homem); os casos de uso são representados por elipses e as interações são

representadas por uma linha contínua.

Os casos de usos podem ser aplicados para captar o comportamento pretendido do sistema que está sendo desenvolvido, sem ser necessário especificar como esse comportamento é implementado. Os casos de uso fornecem uma maneira para os desenvolvedores chegarem a uma compreensão comum com os usuários finais do sistema e com os especialistas do domínio. Além disso, os casos de uso servem para ajudar a validar a arquitetura e para verificar o sistema à medida que ele evolui durante seu desenvolvimento. À proporção que você implementa o seu sistema, esses casos de uso são realizados por colaborações cujos elementos trabalham em conjunto para a execução de cada caso de uso (BOOCH; RUMBAUGH; JACOBSON, 2012, p. 345).

Os elementos principais dentro de um diagrama de caso de uso são os atores e os casos de uso. Os atores são todos os elementos do sistema, podem ser usuários, impressora, banco de dados e demais componentes. Os casos de uso são as ações executadas pelos atores.

Os autores são desenhados na forma de bonecos. No caso de uma impressora ou banco de dados ainda podem ser utilizados ícones para serem representados. O caso de uso deve ser representado por uma elipse que indica sua ação. A Figura 1.4 mostra o exemplo de como são desenhados esses objetos.

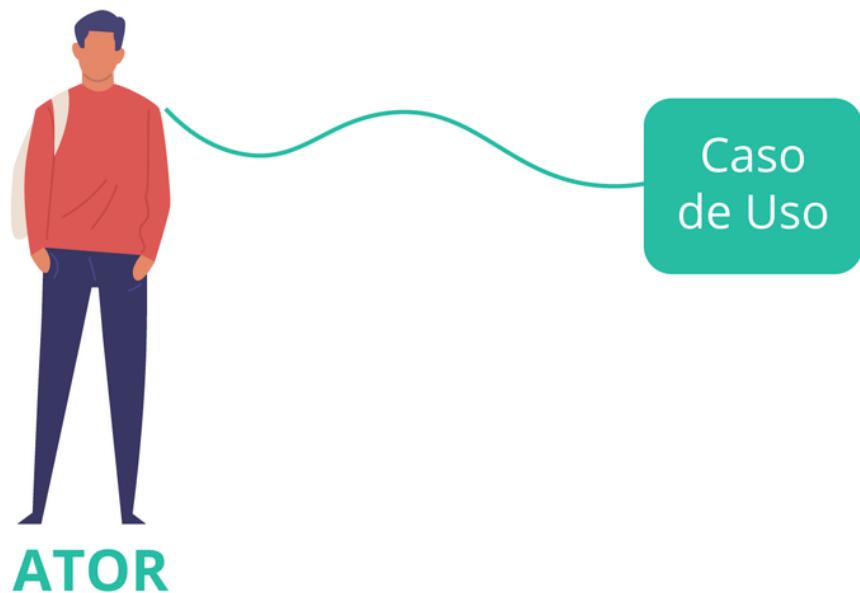


Figura 1.4 - Ator e Caso de Uso

Fonte: Elaborada pelo autor.

#PraCegoVer: Imagem reflete o relacionamento entre um ator e um caso de uso, o ator reflete um papel que interage com o sistema enquanto o caso de uso reflete a descrição de um cenário.

Além de conectores entre atores e casos de uso, também é possível conectar os casos de uso entre si. Tais conexões indicam que um caso de uso pode acontecer na sequência de outro, podendo ser obrigatório ou não. A relação de obrigatoriedade acontece a partir do emprego do <include> e <extend>, compreenda melhor como cada um se comporta.

O <Include> acontece quando o caso de uso A “incluir” o caso de uso B, significa que sempre que o caso de uso A for executado o caso de uso B também será executado. A direção do relacionamento é do caso de uso que está incluindo para o caso de uso incluído.

O <Extend> quando o caso de uso B estende o caso de uso A, significa que quando o caso de uso A for executado o caso de uso B poderá (poderá – talvez não seja) ser executado também. A direção do relacionamento é do caso de uso extensor (aqui o caso de uso B) para o caso de uso estendido (aqui o caso de uso A). (VENTURA, 2014, p. 3).

O diagrama passa a existir quando há a combinação entre os atores e os casos de uso, fazendo, assim, a modelagem do funcionamento de todo o sistema. A Figura 1.5 mostra um exemplo completo de um diagrama de caso de uso.

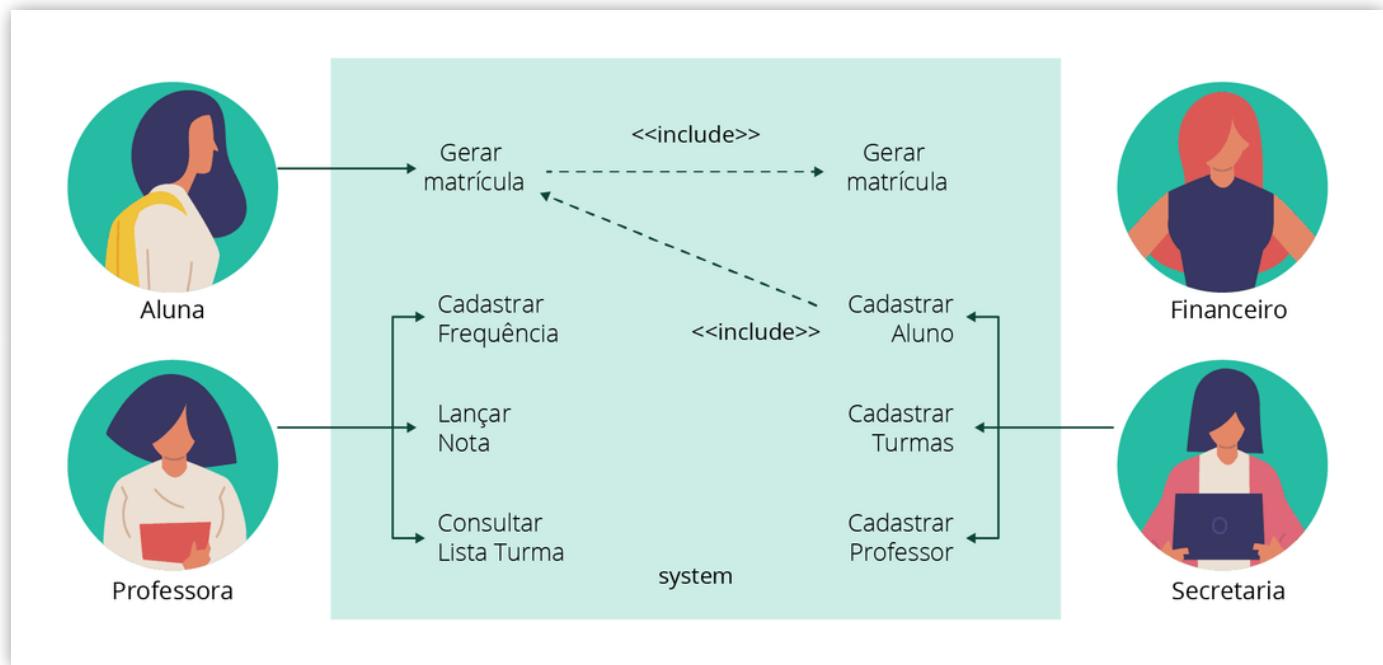


Figura 1.5 - Exemplo de Caso de Uso Completo

Fonte: Santos (s.d., p. 9).

#PraCegoVer: Imagem reflete o relacionamento entre vários atores de uma instituição de ensino e um caso de uso completo, os atores são representações dos papéis de aluna, professora, secretaria e financeiro que interagem com as funcionalidades gerar matrícula, cadastrar frequência, lançar nota, consultar lista turma, cadastrar aluno, cadastrar turmas e cadastrar professor.

Com o desenvolvimento dos casos de uso se encerra um ciclo, de certo modo ele dará vida aos requisitos levantados. Com o diagrama de casos de uso já será possível ter uma visão geral do funcionamento do sistema.

Vamos Praticar

Considerando que o objetivo do projeto que você está desenvolvendo teve início com a entrevista (realizando o Briefing) e que, posteriormente, foi feito o levantamento de requisitos, crie um diagrama de casos de uso.

Descrição do Casos de Uso

A descrição dos casos de uso, também conhecida como caso de uso descritivo, é um documento que descreve detalhadamente cada caso de uso contido no diagrama.

São dois os modos de serem feitas as descrições do caso de uso:

- O modo resumido: descreve em um parágrafo corrido as ações de um caso de uso.
- O modo completo: descreve de maneira mais completa, identificando pré-condições, ações posteriores, deixando o projeto documentado em um maior nível de detalhamento.

Para compreender melhor a diferença entre os tipos de caso de uso, considere o seguinte diagrama de caso de uso:

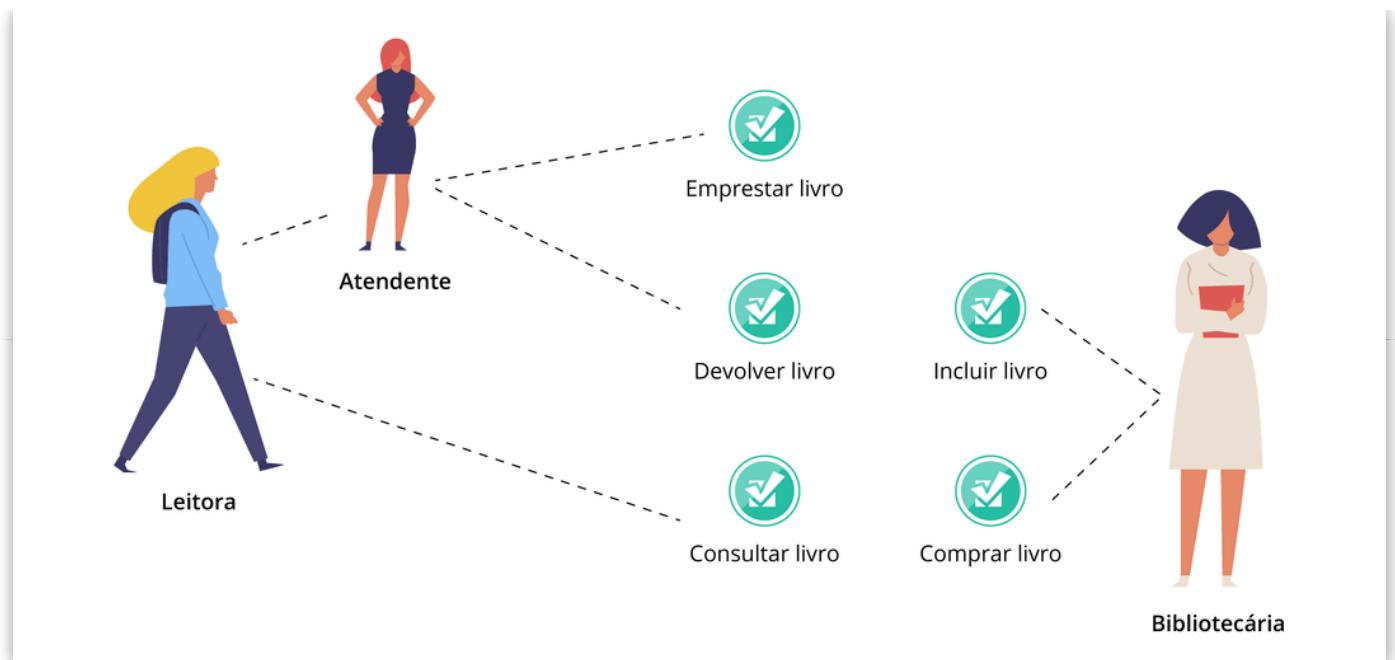


Figura 1.6 - Exemplo de Caso de Uso Completo

Fonte: Nacagawa (2019, p. 6).

#PraCegoVer: Imagem reflete o relacionamento entre vários atores de uma biblioteca e um caso de uso completo, os atores são representações dos papéis de leitora, atendente e bibliotecária que interagem com as funcionalidades emprestar livro, devolver livro, consultar livro, incluir livro e comprar livro.

A partir do diagrama de caso de uso apresentado pela Figura 1.6, é possível desenvolver o seguinte caso de uso do modo descritivo resumido, que é mostrado pelo Quadro 1.3.

Caso de Uso	Alugar livro
Descrição	A Atendente da biblioteca realiza o empréstimo de um ou mais livros a um leitor apto a emprestar livros. O empréstimo é válido por um determinado período de tempo, de acordo com o tipo de leitor. Os livros são levados pelo leitor, depois de devidamente desmagnetizados e marcados como emprestados.

Quadro 1.3 - Caso de Uso Resumido

Fonte: Nacagawa (2019, p. 7).

O caso de uso completo contém mais elementos e deve explicar passo a passo como o sistema funcionará, mesmo sem existir telas e nem ao mesmo linguagem de programação definida.

O desenvolvimento de um bom caso de uso descritivo implica não somente na evolução do ciclo de vida do software, mas também na qualidade do que está sendo desenvolvido. Esse documento servirá como base para o desenvolvimento e teste da aplicação.

Um caso de uso completo pode variar de acordo com o autor (pela perspectiva acadêmica) e, no mercado de trabalho, de acordo com as práticas da empresa. No entanto, tradicionalmente é um documento composto por:

01

Atores Envolvidos

02

Précondições

03

Póscondições

04

Cenário Principal

05

ALTERNATIVO

o usuário pode
cadastrar
um novo
usuário diferente.

Fluxo Alternativo

O Quadro 1.4 mostra um exemplo de um caso de uso completo com o mesmo exemplo visto anteriormente.

Caso de Uso	Alugar livro
Autor Principal	Atendente
Interessados e Interesses	<ul style="list-style-type: none">• Atendente: deseja registrar que um ou mais livros estão em posse de um leitor, para controlar se a devolução será feita no tempo determinado.• Leitor: deseja emprestar um ou mais livros, de forma rápida e segura.• Bibliotecário: deseja controlar o uso dos livros, para que não se percam e para que sempre se saiba com que leitor estão no momento.
Pré-condições	O Atendente está identificado e autenticado.
Pós-condições	Os dados do novo empréstimo são armazenados no sistema. Os livros emprestados possuem status "emprestado".
Cenário Principal	<ol style="list-style-type: none">1. O Leitor chega ao balcão de atendimento da biblioteca e diz ao Atendente que deseja emprestar um ou mais livros da biblioteca.2. O Atendente seleciona a opção para realizar um novo empréstimo.3. O Atendente solicita ao leitor sua carteira de identificação, seja de estudante ou professor.4. O Leitor fornece sua carteira de identificação.5. O Atendente informa ao sistema a identificação do leitor.6. O Sistema exibe o nome do leitor e sua situação.7. O Atendente solicita os livros a serem emprestados.8. O Leitor entrega os livros para a Atendente.9. A Atendente informa ao sistema o código de

identificação dos livros.

10. O Sistema informa a data de devolução de cada livro.
11. O Atendente desbloqueia os livros para que possam sair da biblioteca.
12. O Leitor sai com os livros.

(2-11). A qualquer momento, o Leitor informa ao Atendente que desistiu do empréstimo. Cancelar toda a operação.

Fluxo Alternativo	6. O Sistema informa que o Leitor está impedido de fazer empréstimo, por ter uma situação irregular. Cancelar toda a operação.
--------------------------	---

Quadro 1.4 - Caso de Uso Completo

Fonte: Nacagawa (2019, p. 9)

Durante o estudo dos casos de uso você aprendeu que o diagrama de caso de uso oferece uma visão geral sobre o funcionamento do sistema com seus atores. Posteriormente aprendeu a criar o caso de uso descritivo. Esses dois documentos são de extrema importância para continuidade de um projeto de software, uma vez que servirão de base para o desenvolvimento do sistema e do banco de dados, bem como serão fundamentais para que a equipe de qualidade de software realize os testes necessários no sistema desenvolvido.

praticar

Vamos Praticar

No decorrer desta unidade você realizou: briefing, levantamento de requisitos e fez

o design de um diagrama de casos de uso. Para finalizar o processo de desenvolvimento até aqui, crie ao menos 4 casos de uso completos, para o diagrama criado na atividade anterior.

Utilizando Scrum com Engenharia de Software

Durante a Unidade você aprendeu diversos conceitos relacionados a duas áreas: Engenharia de Software (aplicando UML) e Metodologias Ágeis (utilizando Scrum). A partir de agora verá como interligar tais conceitos, bem como podem ser trabalhados de maneira individual.

Primeiramente é necessário saber que Scrum é uma metodologia ágil para a gestão de projetos, tal metodologia não se aplica exclusivamente a equipes de tecnologia e desenvolvimento de sistemas. Ainda que tenha sido desenvolvida com ênfase em extreme programming, pode ser aplicada em diversas áreas de projeto como marketing, design, arquitetura, entre outras.

O levantamento de requisitos, bem como o desenvolvimento de casos de uso fazem parte da Engenharia de Software, uma área do desenvolvimento de Software. Essa área utiliza-se da UML para documentar todo processo de desenvolvimento de Software.

A partir de agora vamos conhecer um pouco mais do SCRUM para compreender de que maneira poderá contribuir para o desenvolvimento ágil de software integrando com as ferramentas da Engenharia de Software.

Scrum



Figura 1.7 - Rugby / Scrum

Fonte: Ostill / 123RF.

#PraCegoVer: Imagem reflete a relação direta da metodologia scrum e um time, na imagem temos um time do esporte rugby que representa bem a união e o trabalho orquestrado entre seus integrantes.

O Scrum é uma metodologia ágil desenvolvida nos anos 2000 com ênfase em equipes de Software, sua ênfase é a gestão de projetos com entrega rápida e foco em trabalho de equipe. Sempre que se fala do SCRUM se tem uma forte referência ao Rugby e o trabalho em equipe desta modalidade esportiva.

Os All Blacks, lendário time de rúgbi da Nova Zelândia, são uma equipe transcendente. Antes de cada jogo, eles realizam uma cerimônia dos guerreiros Maori de haka, uma dança de guerra que energiza pessoas prestes a entrar no campo de batalha. Ao assistir àquilo, você quase consegue ver a energia emanando de cada jogador e misturando até formar uma unidade. Com pisadas sincronizadas, palmas e cantos — movimentos ritualizados que simulam a ação de cortar a garganta dos inimigos — você assiste a

homens comuns se transformarem em algo maior, algo grandioso. O time invoca o espírito guerreiro que não aceita a derrota nem o desânimo (SUTHERLAND, 2016, p. 45).

O criador do SCRUM, Jeff Sutherland (SUTHERLAND, 2016), apresentou o vídeo dos All Blacks para o seu time e utilizou como fator de inspiração para desenvolvimento da metodologia. Através do vídeo e da relação com o Rugby, juntamente com seu time, foi possível extrair os atores da metodologia (*Scrum Master e Scrum Team*), bem como analisar situações cotidianas que levaram ao ciclo de um projeto utilizando SCRUM.

Saiba mais

Se você ficou curioso sobre a dança haka e sobre como o time All Blacks influenciou a criação da metodologia SCRUM, esta dança mostra o espírito de equipe, bem como o trabalho em grupo.

Tendo como objetivo sair um pouco do mundo do software e entender melhor como que esse ritual funciona e o lado motivacional, assista o vídeo.

Fonte: Elaborado pelo autor.

ASSISTIR

Com o passar dos anos surgiram diversas adaptações do SCRUM, o que você aprenderá será a estrutura original de um projeto SCRUM.

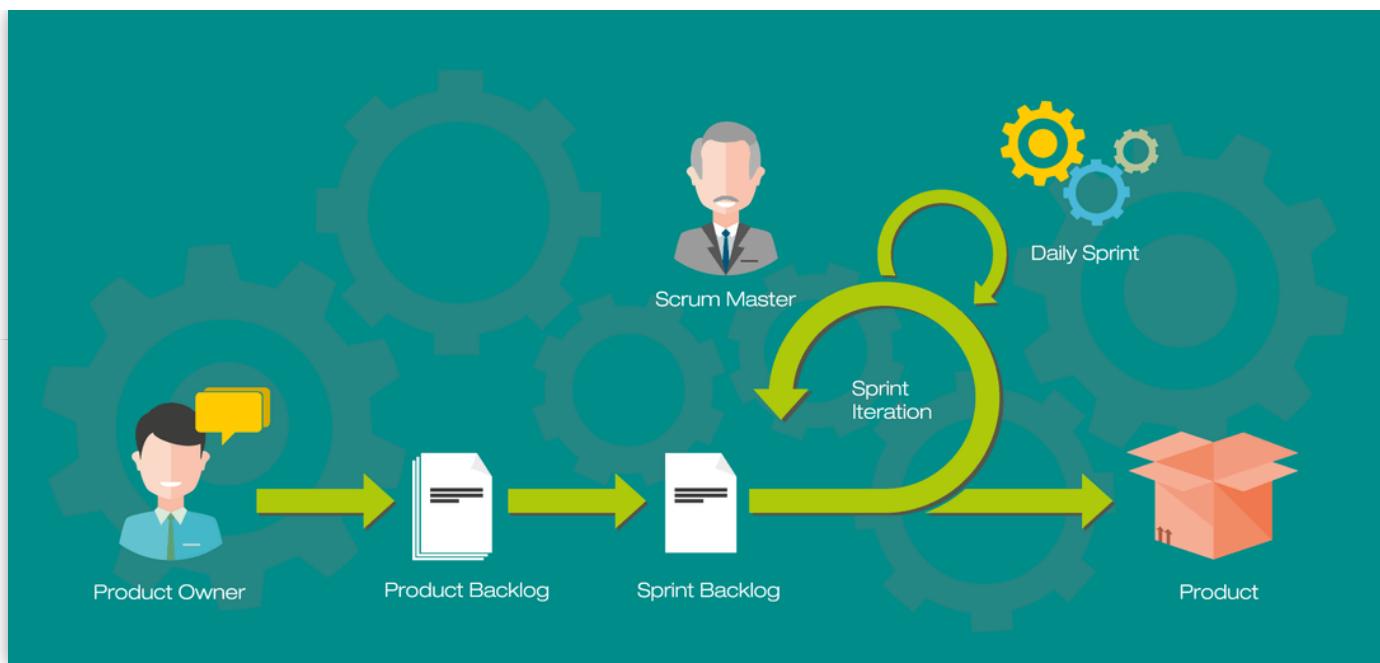


Figura 1.8 - Ciclo Scrum

Fonte: Achmad Fahmi Rosyad / 123RF.

#PraCegoVer: Imagem reflete o ciclo SCRUM, começando o fluxo com o product owner, responsável por definir as funcionalidades do produto e priorizar os itens do product backlog. O fluxo segue para o próximo passo, definição do product backlog, representa uma lista de funcionalidades desejadas para o produto. O fluxo vai para o próximo passo que é definido como sprint backlog, representada por uma lista de atividades que precisam ser realizadas durante a sprint. Na sequência encontramos o sprint iteration que é composto por reuniões diárias e rápidas conhecidas como daily sprint comandadas pelo scrum master. O fluxo entra em um ciclo repetitivo de sprint e a cada ciclo vai recebendo incrementos geridos pelo scrum master até que as entregas aconteçam através da saída dos produtos.

O ciclo Scrum foi mostrado pela Figura 1.8 no qual cada item significa:

- Product Owner: da tradução, o dono do produto, é a pessoa na equipe responsável por compreender toda a necessidade do cliente.
- Product Backlog: é uma lista ordenada de tudo o que é necessário para chegar ao produto final. Em outras palavras, são as “coisas” que devem ser desenvolvidas para chegar àquilo que foi acordado entre todos os envolvidos no projeto (DUARTE, 2019).

- Sprint Backlog: É um conjunto de itens do Product Backlog reunidos que serão inseridos em um Sprint. Pode-se compreender que será uma ou várias funcionalidades que se tornarão tarefas de um membro da equipe.
- Scrum Master: É o líder da equipe, deve ter conhecimentos técnicos e de gestão de projeto que permitam auxiliar os demais membros, bem como garantir a finalização do projeto.
- Scrum Team: São os membros de uma equipe scrum.
- Daily Meeting: é a reunião diária realizada por um time scrum, a reunião é conduzida pelo Scrum Master que sempre faz as seguintes perguntas sobre os sprints de cada membro:

- O O que você fez ontem?*
- O O que está fazendo hoje?*
- O O que fará amanhã?*
- O Está tendo algum problema?*

- Product: O produto é aquilo que se espera quando o projeto for concluído. Em termos de software, pode-se compreender que o produto é o sistema que atenda todas as funcionalidades acordadas com o cliente.

O SCRUM é a metodologia de software mais utilizada por times de software, o seu principal benefício, conforme o ciclo visto acima, são as entregas parciais. Isto significa que ao utilizar o SCRUM um projeto vai sendo entregue em partes. Por exemplo, um sistema acadêmico pode ter o módulo de gerenciamento de alunos entregue, enquanto as demais funcionalidades são desenvolvidas.

Realizando a Integração

No decorrer desta Unidade você aprendeu a realizar a entrevista com o cliente, documentar a entrevista através do Briefing, formalizar o Briefing em requisitos dos sistemas e ter uma visão geral do funcionamento do sistema

através dos casos de uso. Posteriormente, compreendeu os elementos da metodologia SCRUM, uma metodologia ágil que permite a entrega parcial de componentes de um projeto de software.

A partir de agora você verá como documentar um sistema utilizando UML de maneira integrada com metodologia SCRUM. A documentação correta do sistema deve garantir que o que está sendo desenvolvido reflete as necessidades do cliente e o SCRUM deve garantir que seja feito no tempo certo.

O primeiro documento da Engenharia de Software que você aprendeu a desenvolver foi o documento de requisitos. Quando um Product Owner realiza o levantamento de requisitos, todo esse conjunto forma o Product Backlog.

O Backlog do Produto é uma lista ordenada de tudo que deve ser necessário no produto, e é uma origem única dos requisitos para qualquer mudança a ser feita no produto. O Product Owner é responsável pelo Backlog do Produto, incluindo seu conteúdo, disponibilidade e ordenação. (SCHWABER; SUTHERLAND, 2011, p. 9).

Todos os requisitos compõem o Product Backlog, ou seja, todas as funcionalidades que um sistema deve realizar. Um desafio para as equipes de SCRUM acontece durante a Sprint Planning Meeting, ou seja, na reunião de planejamento dos sprints.



Figura 1.9 - Sprint Planning Meeting

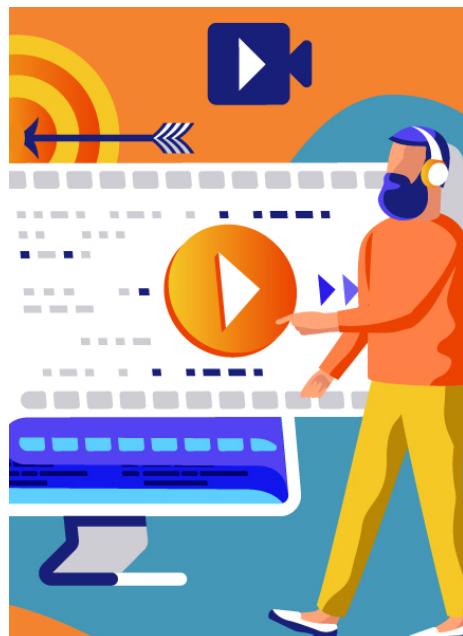
Fonte: Aleksandra Sabelskaia / 123RF.

#PraCegoVer: Imagem reflete planejamento de sprint do ciclo SCRUM, geralmente montado em metodologia kanban com opções de ações em situação de a fazer “to do”, em execução “in progress”, em teste “testing” e pronto “done”.

A etapa de quebrar o Product Backlog em sprint Backlogs é um tanto quanto complexa, existem diversas estratégias, mas não há uma regra geral. No entanto, quando falamos da integração com UML temos um componente que descreve cada funcionalidade do sistema com alto nível de detalhamento: os casos de uso. Desse modo, uma abordagem inicial é transformar cada caso de uso em uma tarefa a ser atribuída a um membro da equipe.

indicações

Material Complementar



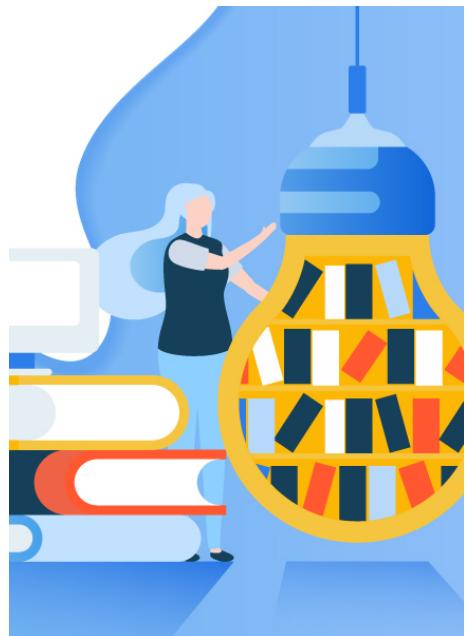
FILME

Golpe de Gênio

Ano: 2009

Comentário: Durante esta unidade você aprendeu muito sobre documentação e gestão de projetos. Nossa indicação de filme não é da área de tecnologia, mas trata do desenvolvimento de produtos e traz uma reflexão sobre fatores que levam ao sucesso ou ao fracasso de uma ideia.

TRAILER



LIVRO

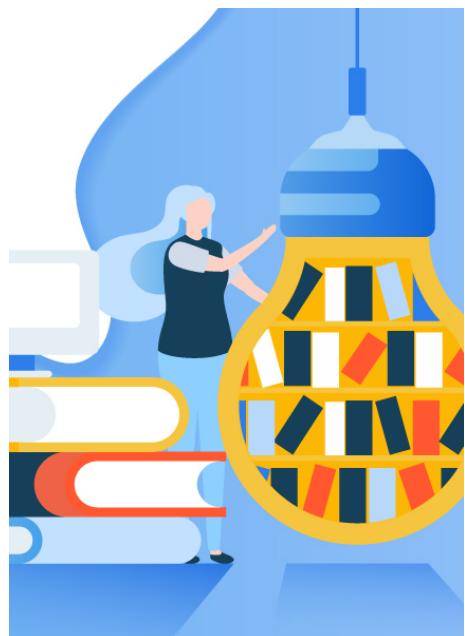
SCRUM: A arte de fazer o dobro de trabalho na metade do tempo

Jeff Sutherland

Editora: leya brasil

ISBN: 8544100880

Comentário: A Unidade teve como objetivo trazer aspectos gerais do desenvolvimento de sistemas utilizando metodologias ágeis integradas com Engenharia de Software. O foco da Unidade foi na integração e desenvolvimento de itens específicos. Para se aprofundar na metodologia SCRUM deixamos a leitura do livro do criador da tal metodologia.

**LIVRO****Engenharia de Software**

Ian Sommerville

Editora: PEARSON BRASIL

ISBN: 8579361087

Comentário: A Unidade 1 teve como objetivo trazer aspectos gerais do desenvolvimento de sistemas utilizando metodologias ágeis integradas com Engenharia de Software. O foco da Unidade foi na integração e desenvolvimento de itens específicos. Para aprofundar seus conhecimentos sobre Engenharia de Software e Engenharia de Requisitos, fica a leitura recomendada do livro mais citado sobre o tema.

conclusão

Conclusão

Prezado aluno, durante essa unidade você percorreu pelo ciclo inicial da vida de um projeto de software, primeiramente com a entrevista ao cliente, aprendendo que tal momento deve ser documentado pelo briefing. Com o Briefing pronto foi possível realizar o processo de levantamento de requisitos funcionais e os não funcionais.

Utilizando UML para mapear todas as ações e atores do sistema criou o diagrama de casos de uso, tal diagrama pode ser complementado com a descrição completa de cada caso de uso. No que se refere à gestão ágil de projetos, aprendeu mais sobre a metodologia Scrum, conhecendo seus principais componentes.

Por fim, realizou a integração de todos os conceitos da Unidade no ciclo de vida de Software, sabendo que a utilização da engenharia de software e uml são essenciais para a documentação do projeto, bem como o emprego de uma metodologia como Scrum permite que o projeto seja desenvolvido dentro do prazo.

referências

Referências Bibliográficas

ALVES FILHO, E. M.; MARTINEZ, A. L. Requisitos funcionais de um sistema de informações para gestão de custos no setor público. In: CONGRESSO BRASILEIRO DE CUSTOS, XXVI, 11 a 13 nov. 2019, Curitiba. **Anais [...]**. Curitiba: [s.n.], 11 a 13 nov. 2019. Disponível em: <https://anaiscbc.emnuvens.com.br/anais/article/view/1764>. Acesso em: jan. 2020.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML** : guia do usuário. Rio de Janeiro: Elsevier Brasil, 2012.

DUARTE, J. Backlog do Produto - Passo a passo como construir e priorizar. **GP4US - Project Management Digital Magazine**. [2019]. Disponível em: https://www_gp4us_com_br/backlog-do-produto/. Acesso em: 16 dez. 2019.

MALUCELLI, A. *et al* . Sistema de informação para apoio à Sistematização da Assistência de Enfermagem. **Revista Brasileira de Enfermagem** , v. 63, n. 4, p. 629-636, 2010.

MELLO, L. **Levantamento de Requisitos** . [18 nov. 2010]. Disponível em: [www.ice.edu.br/TNX/encontrocomputacao/artigos-internos/aluno_leandro_cicero_levantamento_de_requisitos.pdf](http://www_ice_edu_br/TNX/encontrocomputacao/artigos-internos/aluno_leandro_cicero_levantamento_de_requisitos.pdf) . Acesso em: 1 dez. 2019.

NACAGAWA, E. Y. **Descrição de Casos de Uso (Casos de Uso Textuais)** . [2019].

Disponível em: https://edisciplinas.usp.br/pluginfile.php/1906994/mod_resource/content/1/Aula03_DescricaoCasosDeUso.pdf . Acesso em: 4 dez. 2019.

OLIVEIRA, F. H. de. **Caracterização de Modelos Conceituais Utilizando Ontologias de Domínio** : aplicação da Ontologia IMS LD na Construção de Modelos Conceituais Para E-learning. Dissertação (Mestrado) - Universidade Federal da Bahia, Salvador, 2017.

SANTOS, J. C. F. dos. **Criando Diagramas UML com o StarUML** . [s.d.]. Disponível em: <https://cnx.org/contents/b0a7a15b-f4e5-4cdc-bc21-bd43bdc5436f/Criando-Diagramas-UML-com-o-StarUML> . Acesso em: 4 dez.

2019.

SCHWABER, K.; SUTHERLAND, J. **Guia do Scrum** . out. 2011. Disponível em: <https://bit.ly/2O0xnT8> . Acesso em: 21 jan. 2020.

SOMMERVILLE, I. **Engenharia de Software** . São Paulo: Addison-Wesley/Pearson, 2011.

STARUML 3: A sophisticated software modeler for agile and concise modeling. [2020]. Disponível em: <http://staruml.io/> . Acesso em: jan. 2020.

SUTHERLAND, J. **Scrum** : a arte de fazer o dobro do trabalho na metade do tempo. São Paulo: Leya, 2016.

VENTURA, P. **Caso de Uso** - Include, Extend e Generalização. 28 dez. 2014. Disponível em: <https://www.ateomomento.com.br/caso-de-uso-include-extend-e-generalizacao/> . Acesso em: 15 dez. 2019.

WORLD RUGBY. **Latest All Blacks Haka intimidates the French** . [s.d.]. Disponível em: <https://www.youtube.com/watch?v=PptTeyYShdw> . Acesso em: jan. 2020.