Unidade 1

Compreendendo Requisitos e Processo

UniFil

Abertura

Apresentação

Olá, queridos alunos. Vamos iniciar a disciplina de Gerência de Requisitos. Essa disciplina é muito importante para vocês que estão iniciando a área de tecnologia da informação. Afinal, é a partir do conhecimento desta disciplina que vocês vão compreender algumas formas de conversar com os seus clientes e/ou pessoas que estão solicitando o desenvolvimento do projeto de software.

Saber identificar o que o seu *stakeholders* necessita, compreender a forma de documentá-la são alguns conteúdos que vamos abordar. Vamos também abordar sobre a relação do Processo de Desenvolvimento de Software com a Gerência de Requisitos.

E quando falamos em Processo, não podemos deixar de falar sobre o *Rational Unified Process* (RUP) e seus artefatos. Os artefatos do RUP serão a base para o desenvolvimento da nossa documentação.

Para finalizar, vou mostrar para vocês a ferramenta Astah UML para desenvolvermos o Diagrama de Caso de Uso da UML. O levantamento de requisitos é tão importante para o desenvolvimento, assim como a definição e documentação da construção de uma casa é importante para a Engenharia Civil.

Imagine uma pessoa construindo a sua casa sem conversar com você? E se o engenheiro conversar com você e não documentar, de que vale essa conversa? Acho que consegui demonstrar um pouco sobre a importância da disciplina.

Bom estudo para vocês.

Apresentação do Professor

Meu nome é Simone Sawasaki Tanaka. Fiz o meu mestrado em Ciência da Computação na Universidade Estadual de Londrina, conclui a Especialização em Engenharia de Software na Unopar e uma Especialização Educação a Distância no Senac.

Sou graduada em Tecnologia em Processamento de Dados no Cesulon, O Cesulon é a antiga UniFil. Sou docente na UniFil desde 2003, atualmente ministro aulas nos cursos de Ciência da Computação, Engenharia de Software, Análise e Desenvolvimento de Sistemas. Estou como coordenadora de Estágio em todos os cursos citados e coordeno o Projeto Aluno Tutor Google.

Objetivos

- Compreender os conceitos de Requisitos e sua importância;
- Formas de levantar os Requisitos;
- Documentar os Requisitos.

Quer assistir às videoaulas em seu celular? Basta apontar a câmera para os **QR Codes** distribuídos neste conteúdo.

Caso necessário, instale um aplicativo de leitura QR Code no celular e efetue o login na sua conta Gmail.

Apresentação da Disciplina

Utilize o QR Code para assistir!

Assista!

Minicurrículo

Utilize o QR Code para assistir!

Assista!

Introdução da Unidade

Olá, amigo(a) discente! Seja bem-vindo(a)!

Quando falamos em desenvolvimento de software, muitas pessoas acham que é simplesmente sentar em frente de um computador e programar. Basta dominar uma linguagem de programação e pronto!

O desenvolvimento de software não acontece dessa maneira. Assim como na Engenharia Civil, o engenheiro vai até o cliente, levanta a necessidade, o desejo do mesmo, documenta, mostra uma planta e etc., a área do desenvolvimento de software acontece da mesma forma.

No caso do desenvolvimento de software o analista deve procurar o seu cliente e efetuar o levantamento de requisitos. Esse é o momento de entender o que o seu cliente necessita, documentar e validar com o cliente. Agora é possível compreender que para desenvolver um sistema não basta somente dominar uma linguagem de programação.

Na aula 1 será abordado sobre os conceitos de requisitos e qual a sua relação com as metodologias de desenvolvimento de software.

Na aula 2 será abordado sobre a Gerência de Requisitos no Processo Unificado e o Processo de Engenharia de Requisitos.

Objetivos

- Compreender os conceitos de Requisitos;
- Compreender a relação dos requisitos com os modelos de processo;
- Demonstrar onde a Gerência de Requisito se enquadra no RUP;
- Explanar sobre o Processo de Engenharia de Requisitos.

Conteúdo Programático

Aula 01 – Compreendendo Requisitos

Aula 02 – Relação do Processo com os Requisitos

Compreendendo Requisitos

Produzir um sistema de software está relacionado com o objetivo do software e/ou a necessidade do usuário ou cliente. A satisfação dos clientes é de extrema importância para o sucesso do seu software. Um software mal especificado, ou com os requisitos mal levantados pode gerar um software de má qualidade ou que não se adéqua à necessidade do seu cliente.

Antes de prosseguir é necessário compreender o que é um requisito, para isso foi efetuado o levantamento de alguns autores, que definem requisitos como:

- Requisitos é um passo fundamental para o desenvolvimento de um bom produto (PAULA FILHO, 2000).
- Requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais. Os requisitos demonstram as necessidades de um cliente de um sistema que ajuda a resolver um determinado problema, como por exemplo, controlar um dispositivo, enviar um pedido ou encontrar informações. Sommerville (2007).
- Requisito é como uma condição ou capacidade que um software deve realizar (LOPES *apud* KRUCHTEN, 2000).
- Uma condição ou capacidade necessitada por um usuário para resolver um problema ou alcançar um objetivo (IEEE, 1990).
- Requisito pode ser definido como “condição necessária para a obtenção de certo objetivo, ou para o preenchimento de certo fim”. (Michaelis, 2021).

Com o levantamento das definições dos requisitos de diversos autores fica ainda mais explícito compreender a importância dos requisitos no processo de desenvolvimento de software. Quando se fala em requisitos, muitos imaginam somente as documentações de especificação de requisitos, porém ele é muito mais que isso, os requisitos são partes do processo de desenvolvimento de software.

Requisitos e o Processo de Desenvolvimento de Software

Um processo de desenvolvimento pode ser definido como um conjunto de atividades ou práticas, que devem ser seguidas para o desenvolvimento de um determinado sistema de software. Existem muitos processos de software diferentes, mas todos devem incluir quatro atividades fundamentais para a engenharia de software: (Sommerville, 2011)

1. Especificação de software: A funcionalidade do software e as restrições a seu funcionamento devem ser definidas.

2. Projeto e implementação de software: O software deve ser produzido para atender às especificações.
3. Validação de software: O software deve ser validado para garantir que atenda às demandas do cliente.
4. Evolução de software: O software deve evoluir para atender às necessidades de mudança dos clientes.

Das atividades mencionadas, a que vamos focar é a Especificação de software. O modelo de ciclo de vida é a primeira escolha a ser feita no processo de software. A partir desta escolha definir-se-á desde a maneira mais adequada de obter as necessidades do cliente, até quando e como o cliente receberá sua primeira versão operacional do sistema (Devmedia, 2011).

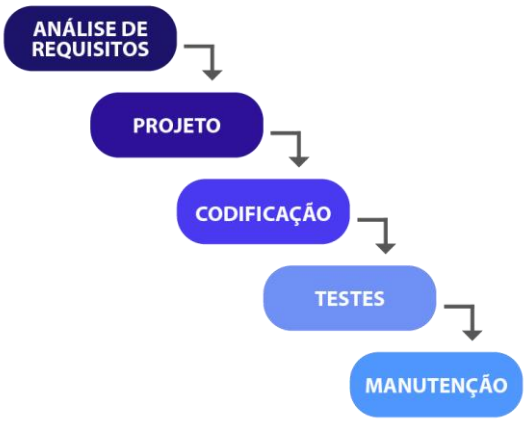
Neste contexto, vamos apresentar alguns ciclos de vida como o Modelo em Cascata, Espiral, Prototipação, Incremental, Iterativo e RUP.

Modelo em Cascata

Divide todo o processo de desenvolvimento de software em fases separadas e sequenciais onde o desenvolvimento movimenta-se somente num sentido, de modo que as etapas não podem ser repetidas.

Somente quando uma fase estiver concluída a outra fase poderá iniciar. Neste modelo nenhum componente do sistema será entregue até a proximidade final do projeto. Observe a Figura 1 onde uma etapa deve acontecer após a outra.

Figura 1: Modelo Cascata



Fonte: Marcoratti, 2021.

Videoaula 1

Utilize o QR Code para assistir!

Agora vamos assistir uma videoaula para compreender um pouco mais sobre os requisitos e processos e processo em cascata.

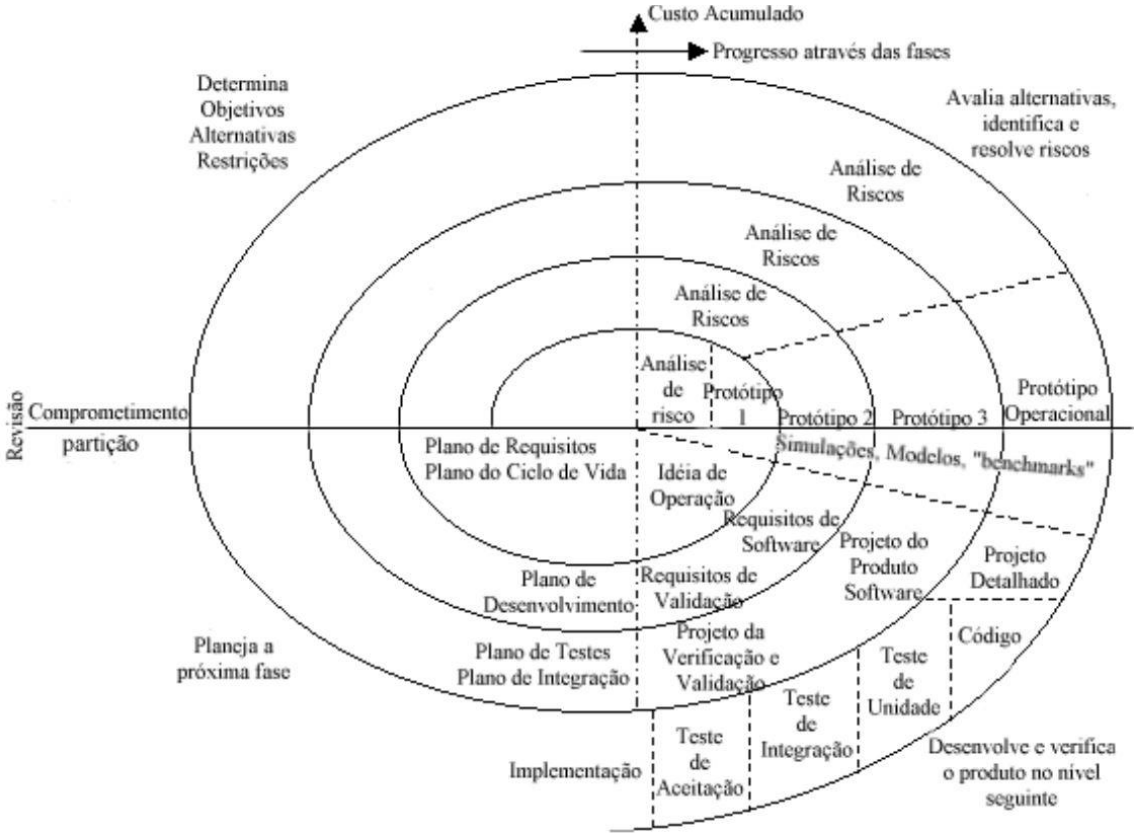
Modelo Espiral (Marcoratti, 2021)

É uma combinação dos modelos interativo e sequencial com ênfase na análise de riscos e no planejamento, de acordo com a Figura 2. O sistema de software é entregue em versões, onde cada versão passa por cada etapa do ciclo de desenvolvimento. Dessa forma o produto final é entregue rapidamente.

Determina um ciclo de atividades dividido em 4 estágios:

1. Determina objetivos, soluções alternativas e restrições.
2. Analisa os riscos do estágio anterior
3. Atividades da fase de desenvolvimento: design, especificação, codificação e verificação
4. Revisão das etapas anteriores e o planejamento da próxima fase

Figura 2 - Modelo Espiral



Fonte: Marcoratti, 2021.

Prototipação (Macoratti, 2021)

Constrói um protótipo do produto de software para mostrar suas funcionalidades de forma limitada. Essa construção é muito importante para a validação junto ao cliente. Essa validação é de extrema importância para receber um feedback do cliente e dar continuidade nas próximas etapas do projeto. Observe a Figura 3, como a Prototipação acontece.

É usado para permitir que os usuários avaliem propostas de desenvolvedores testando-as antes da implementação final do produto. Ajuda a compreender as exigências específicas do usuário que não podem ter sido consideradas pelos desenvolvedores durante a fase de concepção do produto.

Figura 3 - Prototipação

Fonte: Marcoratti, 2021.

Desenvolvimento Incremental

Barry Boehm sugeriu, tendo em vista as limitações da abordagem tradicional, que o desenvolvimento de sistemas de informação poderia ser administrado numa série de incrementos. Assim, poderia haver uma série de ciclos de vida tradicionais para cada incremento.

O Modelo Incremental foi desenvolvido através da combinação entre os modelos linear e prototipação. O desenvolvimento é dividido em etapas, denominadas “incrementos”, que produzirão incrementalmente o sistema, até a sua versão final.

Em cada incremento é realizado todo o ciclo do desenvolvimento de software, do planejamento aos testes do sistema já em funcionamento. Cada etapa produz um sistema totalmente funcional, apesar de ainda não cobrir todos os requisitos.

O Modelo Incremental apresenta diversas vantagens para o desenvolvimento de um software, especialmente se os requisitos não estão claros inicialmente. Por exemplo: quando o Modelo Incremental é utilizado, o primeiro incremento é normalmente constituído do núcleo do sistema. Isto é, os requisitos básicos são implementados, e os detalhes suprimidos.

Esse produto será entregue para uma avaliação, que poderá detectar, inicialmente, problemas que poderiam ser de dimensões muito maiores se detectados somente na entrega do produto final.

Outra vantagem para o desenvolvedor é que, em contato com o sistema, o cliente esclarece seus requisitos e suas prioridades para os próximos incrementos, além de contar com os serviços da versão já produzida.

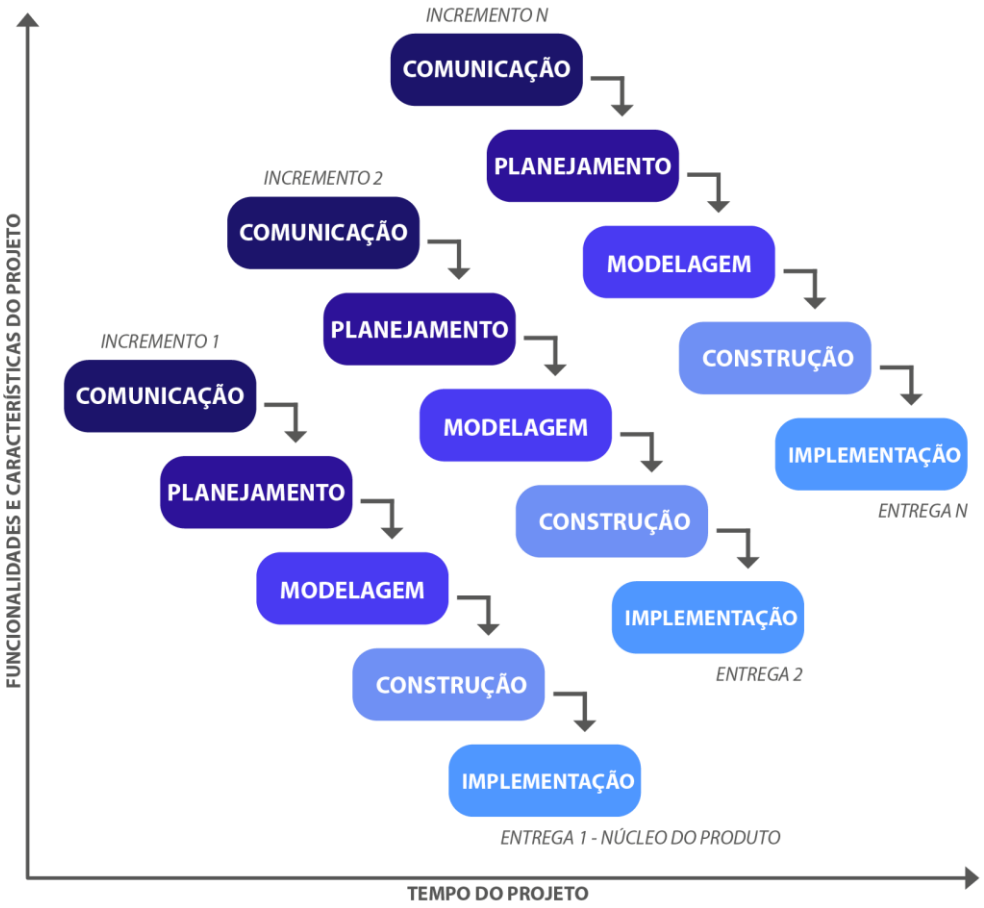
Outras vantagens são:

● A construção de um sistema menor é sempre menos arriscada que a construção de um grande;

● Se um grande erro é cometido, apenas o último incremento é descartado;

● Reduzindo o tempo de desenvolvimento de um sistema, as chances de mudanças nos requisitos do usuário durante o desenvolvimento são menores. Observe a Figura 4 os pequenos blocos que fazem parte de cada entrega.

Figura 4 - Processo Incremental



Fonte: (BATISTA; SANTOS, 2021)

Videoaula 2

Utilize o QR Code para assistir!

Vamos assistir uma videoaula para compreender sobre o modelo Espiral, Prototipação e Incremental.

Processo iterativo

Ao contrário ao ciclo de vida em cascata, onde as atividades são feitas sequencialmente e sem validações ou confirmações, o processo iterativo "divide" o projeto em pequenas e rápidas etapas (as iterações), onde é possível avaliar os requisitos de forma mais clara, estipulando ao final de cada iteração o nível de qualidade e confiabilidade dos requisitos em relação ao projeto, com isso obtém-se grande produtividade e menores níveis de falha.

O processo unificado (*Unified Process*) surgiu com o intuito de auxiliar no desenvolvimento de sistemas orientados a objeto, por meio do conceito de processo iterativo. O Processo Unificado da Rational foi uma "customização" do processo unificado.

Segundo Larman (2008), o processo unificado combina as melhores práticas comumente aceitas, como ciclo de vida iterativo e desenvolvimento guiado por riscos, em uma descrição de processo coesa e bem documentada.

Além disso, esse processo é flexível e pode ser aplicado em uma abordagem leve e ágil que inclui práticas de outros métodos ágeis (tais como XP e Scrum).

Processo Unificado da Rational (RUP)

Segundo Sommerville (2007)

O Rational Unified Process (RUP), é um exemplo de modelo de processo moderno, que foi derivado do trabalho sobre a UML e do Processo Unificado de Desenvolvimento de Software associado. Ele traz elementos de todos os modelos genéricos de processos, apoia a iteração e ilustra boas práticas de especificação e projeto.

Sommerville (2007)

É possível perceber que, devido à maturidade do RUP, e da união de várias metodologias que o compõem, o processo é realizado de uma forma completa, ou seja, por meio dele podemos perceber que o processo:

- É dinâmico, pois, podemos acompanhar as fases do modelo ao longo do tempo.
- É estático, pois existem atividades definidas realizadas no processo.
- É prático, pois sugere boas práticas que podem ser usadas no processo.

Do ponto de vista do gerenciamento, o ciclo de vida de software do Rational Unified Process (RUP) é dividido em quatro fases sequenciais, cada uma concluída por um marco principal, ou seja, cada fase é basicamente um intervalo de tempo entre dois marcos principais. A cada final de fase uma avaliação é executada para determinar se os objetivos da fase foram alcançados. Uma avaliação satisfatória permite que o projeto passe para a próxima fase (IBM, 2007).

O RUP organiza o desenvolvimento de software em quatro fases, onde são tratadas questões sobre planejamento, levantamento de requisitos, análise, implementação, teste e implantação do software. Cada fase tem um papel fundamental para que o objetivo seja cumprido, distribuídos entre vários profissionais como o Analista de sistema, Projetista, Projetista de testes, entre outros. As fases do RUP são:

Iniciação

Elaboração

Transição

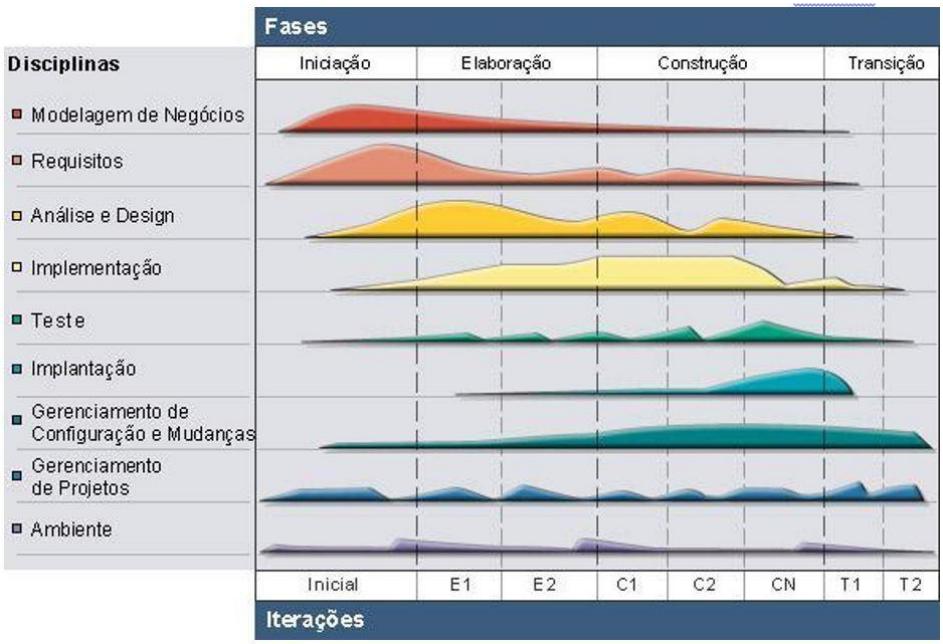
Construção

O RUP foca as atividades que acontecem no processo de desenvolvimento. Elas são denominadas disciplinas. O RUP oferece nove disciplinas, sendo elas:

- Modelagem de Negócios;
- Requisitos;
- Análise e Projeto ("Design");
- Implementação;
- Teste;
- Implantação;
- Ambiente;
- Configuração e Gerência de Mudança;
- Gerência de Projeto.

A disciplina que vamos focar é a de **Requisitos**. Observe a Figura 5, as fases e disciplinas do RUP.

Figura 5 – Disciplinas do RUP



Fonte: IBM,2007.

Videoaula 3

Utilize o QR Code para assistir!

Agora vamos assistir uma videoaula sobre o RUP e toda a importância dele no entendimento das nossas aulas.

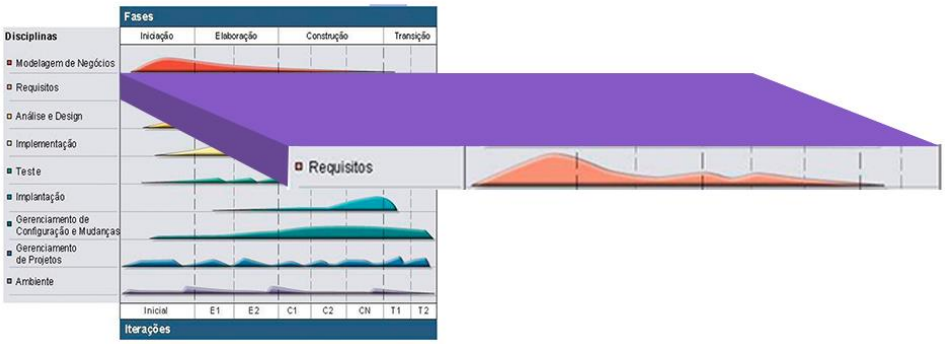
Indicação de Leitura

Leia mais sobre o **RUP**.
Disponível em: https://bit.ly/3wGAoeQ. Acesso em: 15 jul. 2021.

Relação do Processo com os Requisitos

Como foi possível observar na aula 1, o requisito se faz presente em todos os ciclos de vida citados devido a sua importância. O nosso foco é explorar a disciplina de Requisito no RUP, Figura 6.

Figura 6 - Disciplina de Requisitos

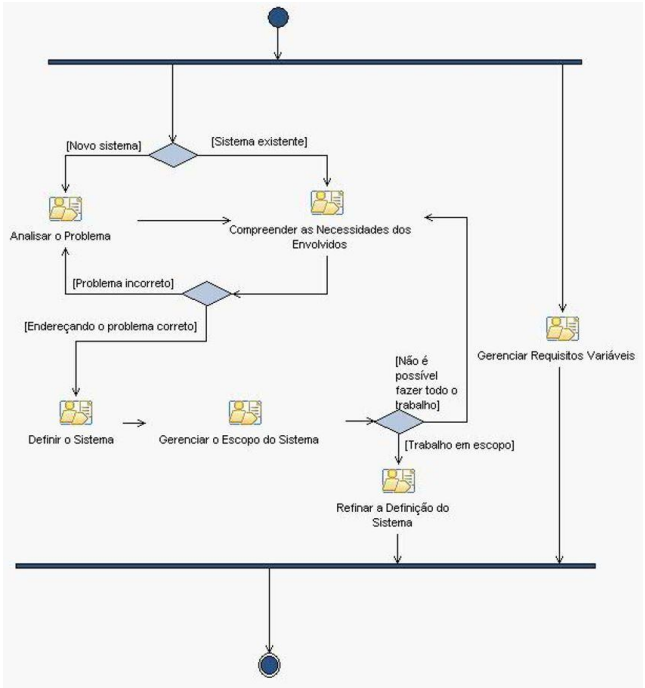


Fonte: IBM,2007.

A disciplina de Requisitos explica como eliciar os requisitos dos investidores e transformá-los em um conjunto de requisitos de Produtos de Trabalho, no escopo do sistema a ser construído e fornece requisitos detalhados sobre o que faz o sistema.

A disciplina de requisitos do RUP traz um *workflow* com o detalhamento das atividades, papéis e responsabilidades e Produtos de trabalhos gerados, conforme detalhado na Figura 7.

Figura 7 - Workflow da Disciplina de Requisitos



Fonte: IBM,2007.

A finalidade da disciplina Requisitos é:

- Estabelecer e manter concordância com os clientes e outros investidores sobre o que o sistema deve fazer;
- Oferecer aos desenvolvedores do sistema uma compreensão melhor dos requisitos do sistema;
- Definir os limites do sistema (ou delimitar o sistema);

- Fornecer uma base para planejar o conteúdo técnico das iterações;
- Fornecer uma base para estimar o custo e o tempo de desenvolvimento do sistema;
- Definir uma interface de usuário para o sistema, focando nas necessidades e metas dos usuários.

Para atingir essas metas é importante, antes de tudo, entender a definição e o escopo do problema que estamos tentando resolver com este sistema. Investidores são identificados e os Requisitos dos investidores são elucidados, reunidos e analisados.

A disciplina de Requisitos captura os Produtos de Trabalho utilizados para definir as capacidades necessárias ao sistema. Os produtos de trabalho são:

- Atributos de Requisitos;
- Esboço Sequencial;
- Especificação de Requisitos de Software;
- Especificações Suplementares;
- Glossário;
- Modelo de Casos de Uso;
- Pedidos dos Envolvidos;
- Plano de Gerenciamento de Requisitos;
- Requisito de Software;
- Visão.

Videoaula 1

Utilize o QR Code para assistir!

Agora vamos assistir uma videoaula sobre o RUP e toda a importância dele no entendimento das nossas aulas.

Indicação de Leitura

Leia mais sobre os **Produtos de Trabalho do RUP**.
Disponível em: https://bit.ly/3hG97oP. Acesso em: 15 jul. 2021.

Saiba mais sobre o *workflow* da disciplina de Requisitos.
Disponível em: https://bit.ly/3wJ53Im. Acesso em: 15 jul. 2021.

Além dos produtos de trabalhos citados acima, o processo possui os papéis, ou seja, responsáveis pelos produtos de trabalho. No caso da disciplina de Requisitos, o papel de maior

destaque é o de Analista. Uma pessoa que desempenha essa função precisa ser, acima de tudo, um especialista na identificação e compreensão de problemas e oportunidades. Isso inclui a capacidade de articular as necessidades que são associadas ao problema-chave a ser resolvido ou a oportunidade a ser realizada.

Além disso, uma pessoa nessa função precisa ser um bom facilitador e deve ter habilidades de comunicação. Ter conhecimento dos domínios dos negócios e da tecnologia são habilidades adicionais convenientes para todos que agem nessa função.

No entanto, essas habilidades podem ter menos importância se a pessoa tiver a capacidade de absorver e de entender as novas informações rapidamente. Como núcleo na equipe do projeto, uma pessoa que desempenha essa função deve ser capaz de colaborar efetivamente com outros membros da equipe.

Indicação de Leitura

Leia mais sobre o analista.
Disponível em: https://bit.ly/3B3O8Ul. Acesso em: 15 jul. 2021.

Aspectos do Gerenciamento de Requisitos

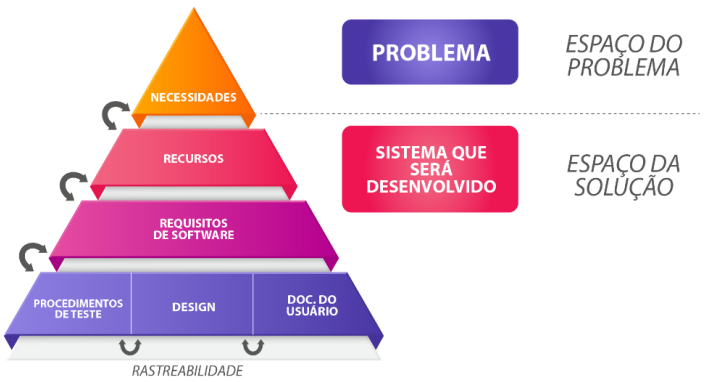
Em relação ao conceito de Gerenciamento de Requisitos trata-se de uma abordagem sistemática para localizar, elicitar, organizar, documentar e gerenciar os requisitos estabelecendo e mantendo um acordo entre cliente/usuário e a equipe do projeto nas mudanças dos requisitos.

A Figura 8 mostra uma visão geral do gerenciamento de requisitos com casos de uso, nesta figura você pode observar que existem diversos níveis de requisitos, cada qual com um direcionamento para o interessado no projeto de software. No mais alto nível de gerenciamento de requisitos pode-se observar que se trata de requisito de alto nível que é a **necessidade** do usuário em relação ao projeto de software.

No segundo nível de requisitos, temos uma visão do que o analista de sistemas e/ou arquiteto de software está vislumbrando em termos de **recursos** para o software que será construído.

O terceiro nível é uma visão do desenvolvedor onde podemos **identificar os casos de uso e os requisitos não funcionais** do projeto em questão.

Figura 8 - Visão Geral do Gerenciamento de Requisitos



Existem vários tipos de requisitos, porém vamos considerar 3, sendo Requisitos do Negócio (Necessidade), Requisitos do Usuário (recursos) e Requisitos de Software.

Requisitos do Negócio

Descrevem as necessidades do negócio que o software precisa atender, como por exemplo, prazo, custo, regras, alinhamento com os objetivos estratégicos, etc. (Ponto mais alto da pirâmide - Necessidades).

Requisitos do Usuário

Descrevem as necessidades do usuário do ponto de vista das tarefas a serem realizadas no software, definindo os objetivos geral e específicos, bem como as suas funcionalidades. (Ponto intermediário da pirâmide - Recursos).

Requisitos de Software

São as ações que o software deve executar, possuindo características e condições próprias, de forma a automatizar uma tarefa de um processo de negócio. Aqui definimos os requisitos funcionais e não funcionais. (Base da pirâmide - Requisitos Funcionais e Não Funcionais).

Analise a tirinha abaixo (Figura 9) e veja como é importante o Requisito de Negócio.

Figura 9 - Tirinha Dilbert

Fonte: ADAMS, 2006

Videoaula 2

Utilize o QR Code para assistir!

Agora vamos assistir um vídeo para entender a Visão Geral da Engenharia de Requisitos e falar da importância dos mesmos.

No decorrer da disciplina será abordado sobre os Requisitos Funcionais e Não Funcionais.

Requisitos Funcionais

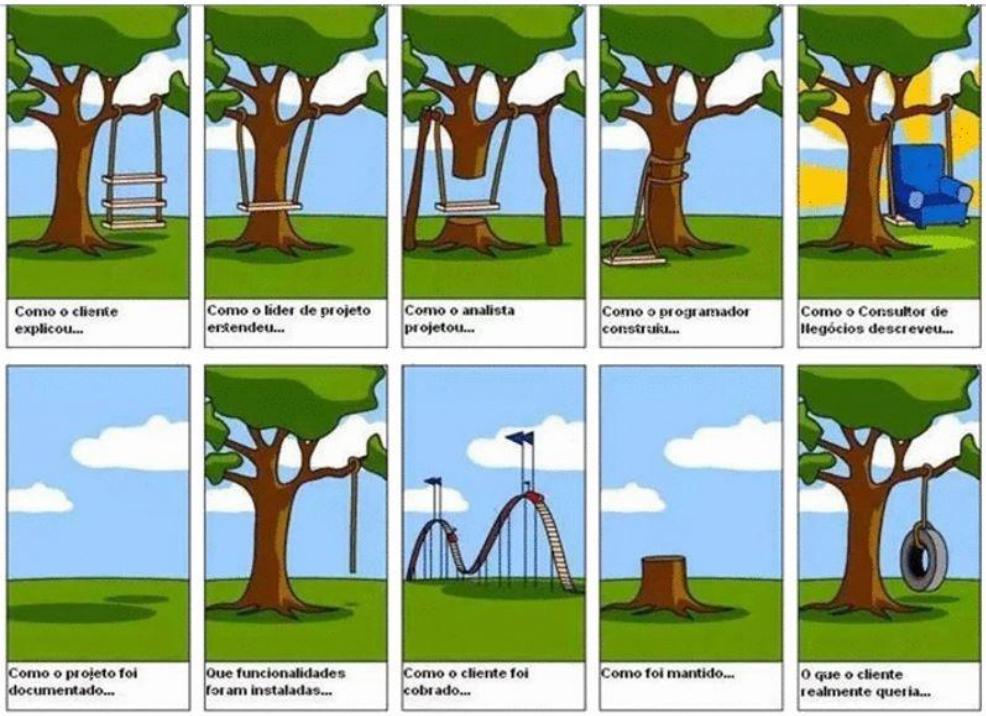
São declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer (Sommerville, 2011).

Requisitos Não Funcionais

São restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo (Sommerville, 2011).

A Figura 10 descreve claramente um dos problemas existentes no Gerenciamento de Requisitos. A falta de comunicação, documentação, gerenciamento entre outras ações.

Figura 10 - Falha na Especificação de Requisitos



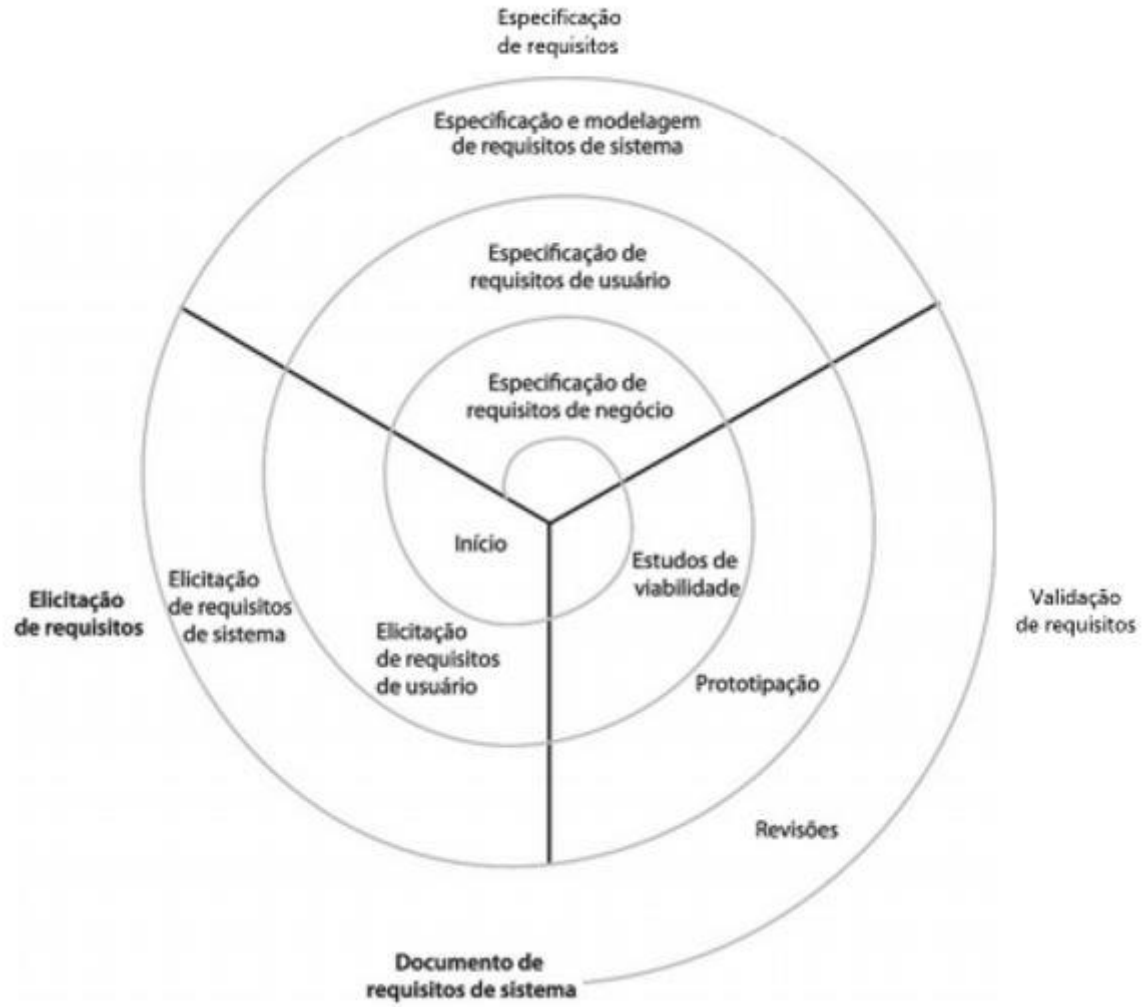
Processo de Engenharia de Requisitos

De acordo com Sommerville, 2011, os processos de engenharia de requisitos podem incluir quatro atividades de alto nível. Elas visam avaliar se o sistema é útil para a empresa (estudo de viabilidade), descobrindo requisitos (elicitação e análise), convertendo-os em alguma forma-padrão (especificação) e verificar se os requisitos realmente definem o sistema que o cliente quer (validação) e documentação, conforme mostrada na Figura 11.

Figura 11 - Processo de Engenharia de Requisitos

No entanto, na prática, a engenharia de requisitos é um processo iterativo em que as atividades são intercaladas, conforme exibido na Figura 12.

Figura 12 - Processo Espiral de Engenharia de Requisitos



Fonte: Sommerville, 2011.

Esse modelo espiral acomoda abordagens em que os requisitos são desenvolvidos em diferentes níveis de detalhamento. O número de iterações em torno da espiral pode variar, assim, a espiral pode acabar depois da definição de alguns ou de todos os requisitos de usuário. No lugar de prototipação, o desenvolvimento ágil pode ser usado para que os requisitos e a implementação do sistema sejam desenvolvidos em conjunto (Sommerville, 2011).

Indicação de Leitura

Leia mais sobre o “Processo de Engenharia de Requisitos” nos livros:

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Education, 2011.

SIMÕES, G. S.; VAZQUEZ, C. E. **Engenharia de Requisitos:** software orientado ao negócio. Rio de Janeiro: Brasport, 2016.
Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/160193/epub/0. Acesso em: 15 jul. 2021.

Videoaula 3

Utilize o QR Code para assistir!

Agora assista a videoaula sobre o processo de gerenciamento de requisitos.

Encerramento

Chegamos ao final da nossa primeira unidade. O objetivo dessa unidade foi conceituar os requisitos e demonstrar que os mesmos existem e fazem parte de vários modelos de ciclos de vida.

Foi explanado brevemente os tipos de requisitos existentes e a importância dos mesmos. Nas próximas aulas, vamos aprofundar um pouco mais nos requisitos e nas documentações.

Como falei na introdução da unidade, se o engenheiro civil conversa com o seu cliente analisando a sua necessidade, o analista também precisa conversar com o seu cliente, não tem como construir o software sem conhecer os requisitos necessário.

Referências

AMBLER, S. **Modelagem Agil práticas eficazes para programação eXtrema**. [S.l.]: Bookman, 2004.

BATISTA, D. N.; SANTOS, M. **Modelos Incremental, Espiral e de Prototipação**: modelo incremental. Modelo Incremental. Disponível em: http://engenhariadesoftwareuesb.blogspot.com/2012/12/blog-post.html. Acesso em: 22 maio 2021.

BOOCH, G.; RUMBAUGH, J; JACOBSON, I. **UML - Guia do Usuário**. 1.ed. Rio de Janeiro: Campus, 2006.

COCKBURN, A. **Escrevendo Casos de Uso Eficazes.** 1.ed. [S.l.]: Bookman, 2005.

DEVMEDIA. **Ciclos de Vida do Software**. Disponível em: https://www.devmedia.com.br/ciclos-de-vida-do-software/21099. Acesso em: 22 maio 2021.

GUEDES, G.T. A. **UML 2 – Uma Abordagem Prática**. 3. ed. São Paulo: Novatec, 2018.

IBM. **RATIONAL SOFTWARE CORPORATION**. IBM Rational Unified Process v2007. IBM, 2007.

KRUCHTEN, P. **Introdução ao RUP.** 2. ed. [S.l.]: Ciência Moderna, 2003.

LARMAN, C. **Utilizando UML e Padrões.** 3. ed. Porto Alegre: Bookman, 2007.

MACORATTI, J. C. **O ciclo de vida do desenvolvimento de Software**. Disponível em: http://www.macoratti.net/17/09/net_slcd1.htm. Acesso em: 22 maio 2021

MICHAELIS. **Dicionário.** Disponível em: https://michaelis.uol.com.br/busca?r=0&f=0&t=0&palavra=requisito. Acesso em: 22 maio 2021.

PAULA FILHO, W. **Engenharia de Software:** fundamentos, métodos e padrões. 3. ed.

PRESSMAN, R. S. **Engenharia de Software.** 6. ed. McGraw Hill, 2006.

SIMÕES, G. S.; VAZQUEZ, C. E. **Engenharia de Requisitos**: software orientado ao negócio. 1. ed. Rio de Janeiro: Brasport, 2016.

SOMMERVILLE, I. **Engenharia de software.** 9. ed. São Paulo: Pearson Education, 2011.

Esperamos que este guia o tenha ajudado compreender a organização e o funcionamento de seu curso. Outras questões importantes relacionadas ao curso serão disponibilizadas pela coordenação.

Grande abraço e sucesso!

UniFil