ENGENHARIA DE REQUISITOS

AULA 4

Profª Rosemari Pavan Rattmann

CONVERSA INICIAL

Anteriormente, descrevemos o que são requisitos, seus tipos e como identificá-los. Falamos sobre os passos para elicitar os requisitos, quais os envolvidos e como lhes dar a atenção necessária, devido a sua importância para o entendimento da solução a ser projetada e construída.

Nesta etapa, abordaremos como escrever todo o levantamento de informações, utilizando as técnicas de prototipação, comuns e úteis no processo de análise de requisitos, criando versões completas e rápidas de um *software* e refinando requisitos.

Estudaremos como, ao identificar os requisitos, devemos escrever e explicitar os fluxos de informações de forma clara a todos os *stakeholders*, garantindo um projeto completo com o funcionamento adequado as necessidades.

Após todo o processo de elicitação e análise de requisitos, é necessário fazer uma avaliação detalhada e os devidos refinamentos, como quais requisitos serão elencados para a solução, e quais serão descartados nas versões do *software* em construção.

Falaremos ainda sobre a importância de verificar e validar os requisitos para garantir o entendimento de todos os envolvidos – tanto os usuários, quanto os desenvolvedores –, para que o *software* a ser entregue seja o que foi esperado e definido.

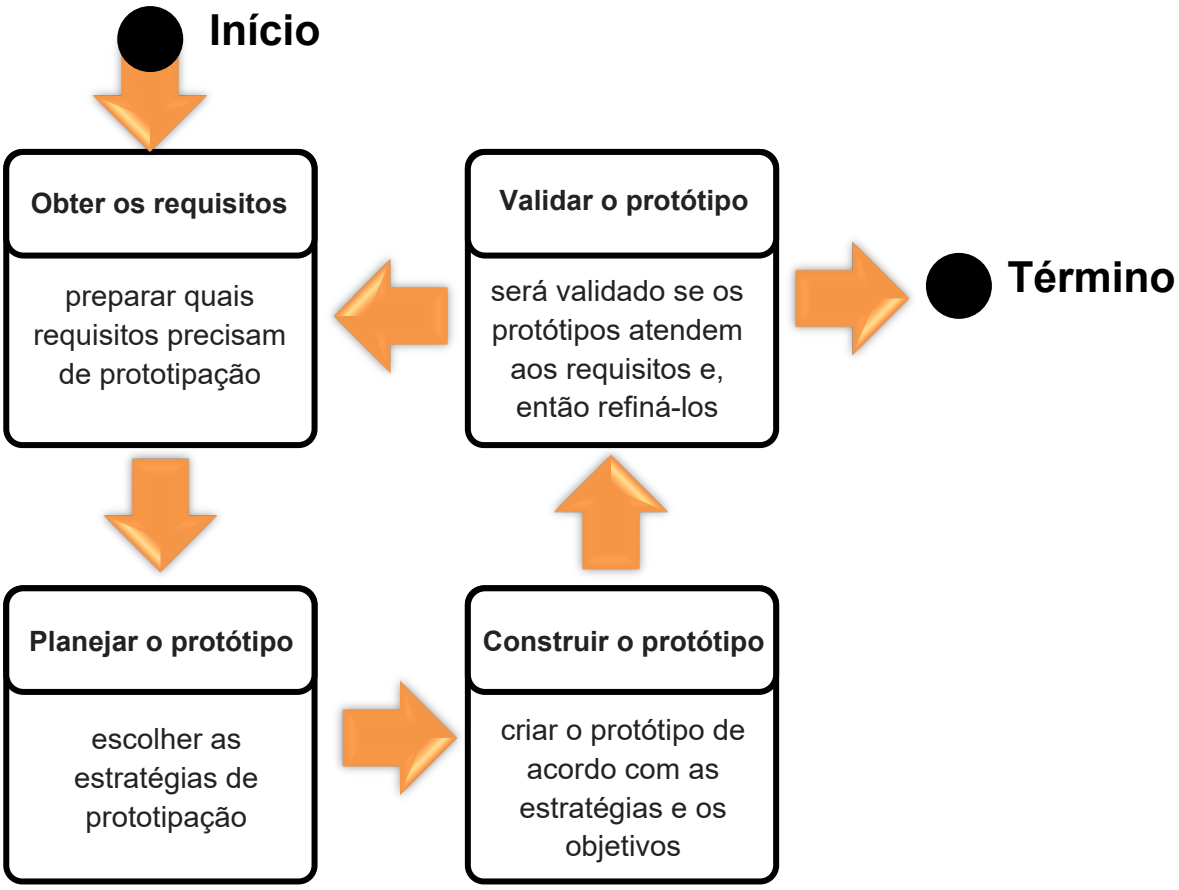
TEMA 1 – PROTOTIPAÇÃO DE *SOFTWARE*

A prototipação está associada a telas e desenhos, e é utilizada para demonstrar modelos de produtos e serviços. No desenvolvimento de um *software*, a preocupação com a interface gráfica, integração com o usuário e outros elementos de *hardware* e serviços, é responsabilidade da disciplina de Análise e Projeto (arquitetura) e, atualmente, o termo UX (*user experience*, ou, em português, “experiência do usuário”) tem sido muito utilizado. Ele se refere ao estudo da experiência do usuário de um produto ou serviço, com o objetivo de estimular melhorias. Porém, nada disso é responsabilidade da área de engenharia de requisitos. O enfoque da prototipação é para a validação e descoberta de requisitos.

Na fase de requisitos, a prototipação visa representar visualmente como serão as interfaces do *software*, simulando as telas e a interação com o usuário. A técnica de prototipação simula para o usuário (e todos os *stakeholders*) o funcionamento dos seus requisitos, antes de o *software* ser construído. Protótipos são também utilizados para validar os requisitos de forma que o próprio usuário possa verificar se suas necessidades serão atendidas, e o analista de requisitos pode perceber eventuais pontos de falha.

O ciclo de uma prototipação começa na obtenção dos requisitos, seguido do planejamento e construção do protótipo. Por fim, o usuário e o analista de requisitos farão a validação dos requisitos atendidos e seu refinamento, como mostrado na Figura 1. Este processo de refinamento pode se repetir várias vezes, até que os requisitos estejam bem definidos para o desenvolvimento.

Figura 1 – Ciclo de uma prototipação



Crédito: Pavan Rattmann/Arte UT.

Existem várias técnicas de prototipação, e o analista de requisitos deverá avaliar qual utilizar, de acordo com o objetivo a ser atingido. Como já dissemos,

nessa fase de análise de requisitos, a criação de protótipos visa refinar os requisitos funcionais, incluindo elementos lógicos necessários, e não interfaces gráficas completas. Segundo Vasquez e Simões (2016), o produto final de maior valor da prototipação é o *feedback* gerado pelo cliente.

1.1 Prototipação de baixa fidelidade *versus* alta fidelidade

A classificação da prototipação como sendo de baixa ou alta fidelidade depende do nível de aparência do protótipo em comparação ao produto final.

O protótipo de baixa fidelidade é mais simples, requer menos esforço, tempo e custo para sua elaboração e, geralmente, é feito no início do projeto. O foco é avaliar as funcionalidades entendidas pelos requisitos apresentados, apenas como um esboço que pode ser facilmente alterado.

O protótipo de alta fidelidade apresenta um alto nível de detalhe e aparência gráfica, com riqueza de interfaces, recursos gráficos e navegabilidade. Normalmente, esse tipo de protótipo é construído utilizando a mesma ferramenta de desenvolvimento do sistema, e é ótima para descobrir novos requisitos de usabilidade e requisitos não funcionais. No entanto, há o risco de o usuário criar expectativas e entender que o sistema está “quase” pronto. Outra desvantagem é que é necessário um tempo maior para a construção desse tipo de protótipo.

1.2 Prototipação horizontal *versus* vertical

Elaborar protótipos completos para um sistema grande pode necessitar de muito tempo, o que diminui os benefícios da prototipação. Então, para evitar esse risco, muitas vezes constroem-se protótipos parciais, com uma visão mais ampla e menos profunda, ou uma visão específica de uma parte do produto final. A escolha entre as duas abordagens deve ser baseada no propósito da prototipação: se é validar o escopo, ou detalhar os elementos já conhecidos.

A prototipação horizontal busca uma avaliação mais ampla de várias funcionalidades, sem aprofundamento sobre seu funcionamento; é vantajosa para apresentar uma visão geral do sistema entendido até esse ponto.

A utilização da prototipação vertical visa detalhar algumas funcionalidades e requisitos, geralmente mais complexos, ou com menor conhecimento do analista de requisitos, que requeiram mais esclarecimento. A vantagem dessa

abordagem é avaliar requisitos específicos e aprofundar o entendimento dos detalhes.

1.3 Prototipação descartável *versus* evolutiva

O objetivo da prototipação descartável é permitir uma avaliação prática do sistema para ajudar a encontrar problemas com os requisitos, para depois ser descartado. O sistema será desenvolvido com outro processo. O objetivo é validar e refinar os requisitos que ainda não estão bem compreendidos.

Na prototipação evolutiva, o protótipo inicial será produzido e refinado ao longo de várias versões, até atingir o sistema final. O objetivo aqui é entregar aos usuários um sistema funcionando.

Após uma primeira análise da prototipação, os requisitos devem ser melhorados, e pode ser necessário novo processo de prototipação para que novos problemas ou falhas nos requisitos sejam identificados, e novas soluções sejam estabelecidas. Cada versão dos protótipos aumenta a precisão dos requisitos para propiciar o projeto do produto final.

Antes de realizar a prototipação, o analista de requisitos precisa esclarecer com o cliente o objetivo do protótipo a ser criado, e explicar sobre o tempo reduzido nessa construção, o qual não reflete o tempo real para a construção do sistema final. A explicação de que o protótipo é uma ferramenta de simulação sobre como o sistema se comportará quando ficar pronto, não possuindo processamento real, é bastante importante, para que o usuário não compare a velocidade no desempenho de um protótipo com a do produto final (Vasquez; Simões, 2016).

TEMA 2 – ANÁLISE DE REQUISITOS

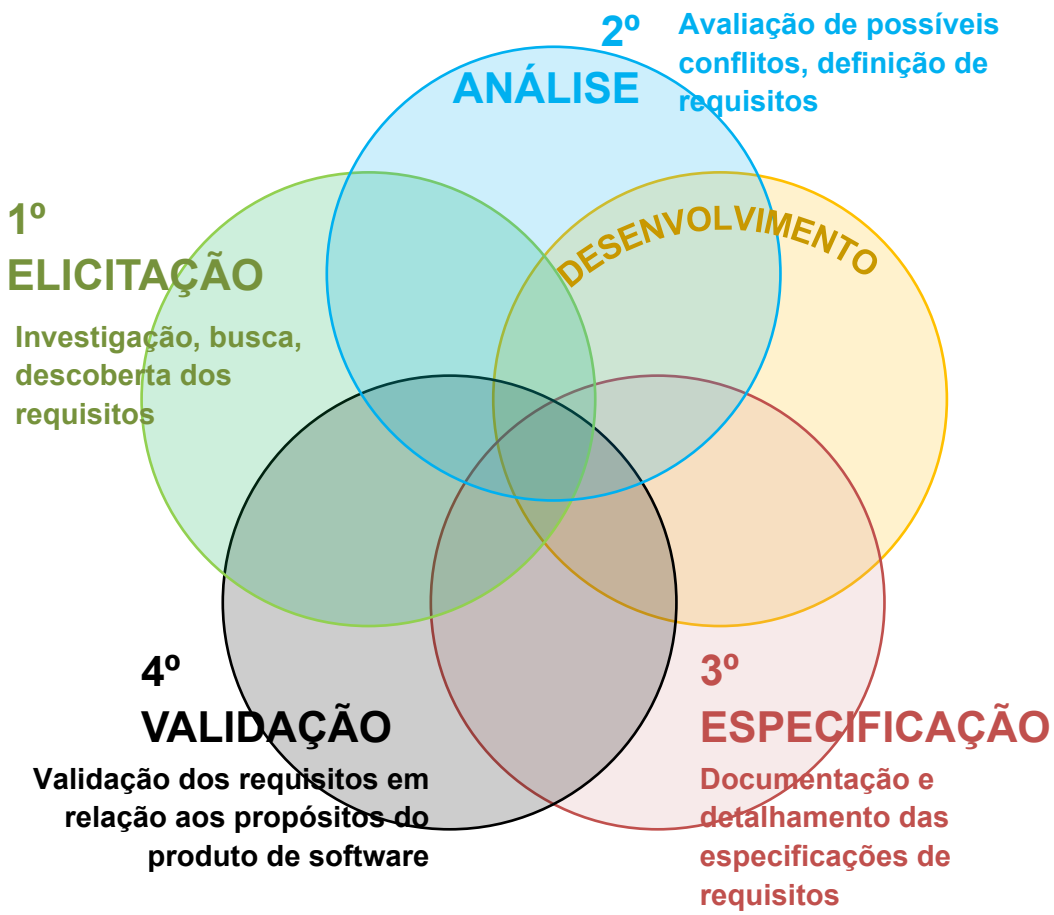
Segundo Vasquez e Simões (2016), a elicitação de requisitos identifica as peças do quebra-cabeças, e a análise de requisitos procura montá-lo. As informações descobertas na elicitação das necessidades de negócio e dos requisitos são as entradas para a análise de requisitos, melhorar o entendimento de todas as informações e completá-las, identificando o que faltou para minimizar problemas futuros.

Essa fase de análise de requisitos é necessária porque podem existir vários *stakeholders*, cada um com suas vontades e necessidades, e não é bom

levar todos os requisitos deles para a construção de uma solução única devido aos conflitos e ambiguidades existentes. Ao final, além de mais tempo e custo para a conclusão, ainda podem ter sido elencadas informações equivocadas ou com erros no funcionamento do *software* construído, o que levará a retrabalho, gasto de mais tempo e recursos, além da insatisfação do cliente. Ao ampliar o entendimento das informações levantadas, buscamos identificar ambiguidades, conflitos e erros nos requisitos. Ainda nessa fase, formulamos questionamentos a serem usados como importantes insumos para novas atividades de elicitação que sejam necessárias.

Para realizar a análise de requisitos, várias tarefas precisam ser planejadas, no sentido de avaliar cada pedaço de informação revelado na elicitação, e como eles se relacionam com outras partes, além de possibilitar um detalhamento de cada requisito e compará-lo ao escopo do projeto, visando atender à necessidade primária do cliente. Relembrando das etapas anteriores, a Figura 2 mostra as fases do processo de elicitação de requisitos.

Figura 2 – Fases do processo de elicitação de requisitos

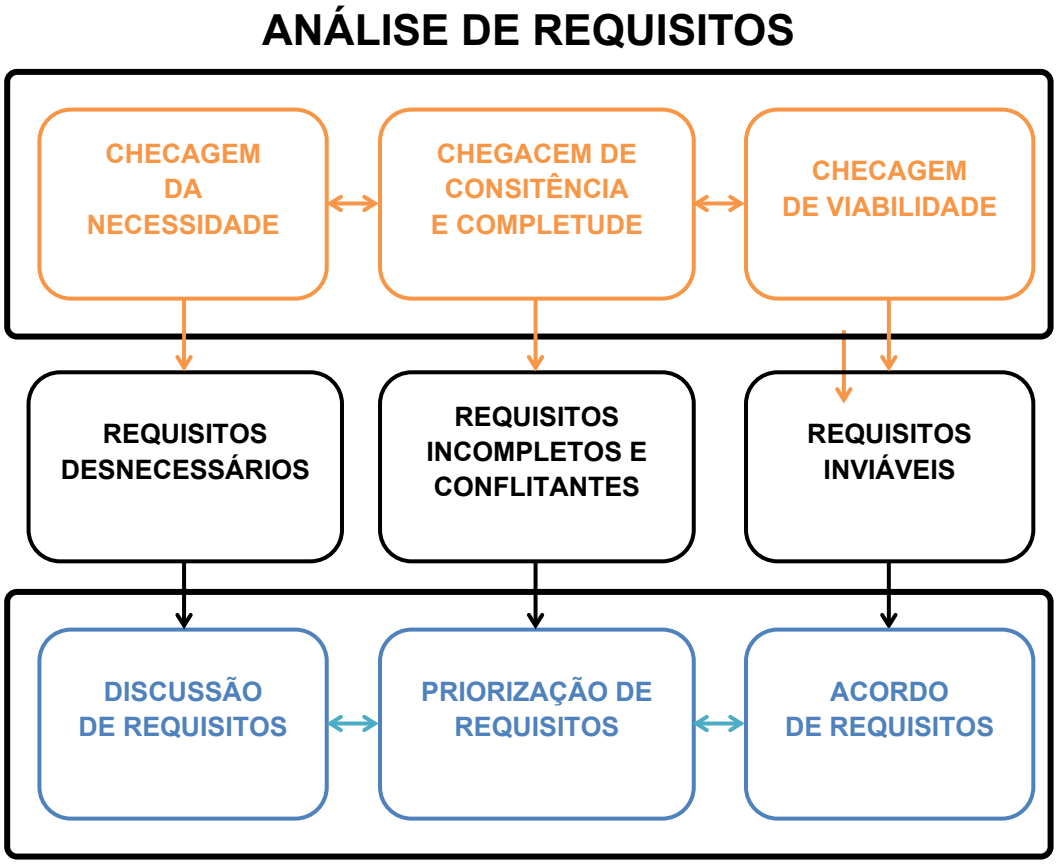


Crédito: Pavan Rattmann/Arte UT.

Outro dos objetivos da análise de requisitos é descobrir problemas, incompletudes e inconsistências nos requisitos elicitados, além de possibilitar o rastreamento entre os requisitos para verificar conflitos e requisitos sobrepostos.

Sempre que o analista de requisitos perceber problemas ou falhas, retornará aos *stakeholders* para resolvê-los e negociar as divergências e prioridades. Nessa fase, geralmente, utiliza-se um *checklist*, ou seja, uma lista de verificação de problemas, para ajudar na análise, conferindo cada requisito. A Figura 3 apresenta as várias tarefas desse processo.

Figura 3 – Tarefas no processo de análise de requisitos



Crédito: Pavan Rattmann/Arte UT.

- **Checagem da necessidade**: a lista de requisitos identificados pode não atender ao escopo do projeto, por não contribuir com os objetivos de negócio da organização, ou para o problema específico tratado pelo sistema. Cada requisito deve ser analisado e retornado aos *stakeholders* para garantir sua importância e necessidade para a solução a ser desenvolvida.

- **Checagem da consistência e completude**: os requisitos são verificados entre si para determinar consistência e completude. Consistência significa que nenhum requisito deve ser contraditório; completude refere-se a que nenhum serviço (ou limitação) necessário foi esquecido.
- **Checagem da viabilidade**: os requisitos são verificados para garantir que são viáveis dentro do orçamento e tempo disponíveis para o desenvolvimento do sistema. O estudo de viabilidade indica se o esforço, custo e tempo para o desenvolvimento valem a pena, visando à tomada de decisão por parte do cliente e, também, possíveis alternativas de solução técnica.

Tendo os requisitos identificados e o modelo de análise criado, os *stakeholders* e desenvolvedores negociam as prioridades de cada requisito para desenvolver um plano de projeto real. As necessidades do cliente e o escopo contratado precisam ser avaliados em relação às prioridades, para garantir que o sistema seja construído de forma correta e que atenda às expectativas.

Como já estudamos anteriormente, elicitar requisitos significa identificar e descrever os requisitos de um *software* a ser construído. Os processos de elicitação de requisitos, análise e negociação são interativos e intercalados, e devem ser repetidos várias vezes. A negociação de requisitos é sempre necessária para resolver conflitos e remover a sobreposição de requisitos. Por fim, o documento de especificação oficial descreverá os requisitos de um *software* a ser construído, e representará um acordo contratual entre cliente e os fornecedores do *software*.

TEMA 3 – FLUXOS OPERACIONAIS

Na fase de elicitação de requisitos, o analista de requisitos reúne várias documentações, resumos de reuniões, questionários, entrevistas e anotações sobre as conversas com todos os *stakeholders*. Todo esse material deve ser avaliado na análise de requisitos para considerar como o *software* afetará o funcionamento da empresa ou do departamento, e que mudanças nos fluxos operacionais irá produzir. Todas as tarefas que são realizadas sem o *software* são procedimentos formais e práticas estabelecidas na organização e, portanto, são consideradas em relação aos requisitos levantados.

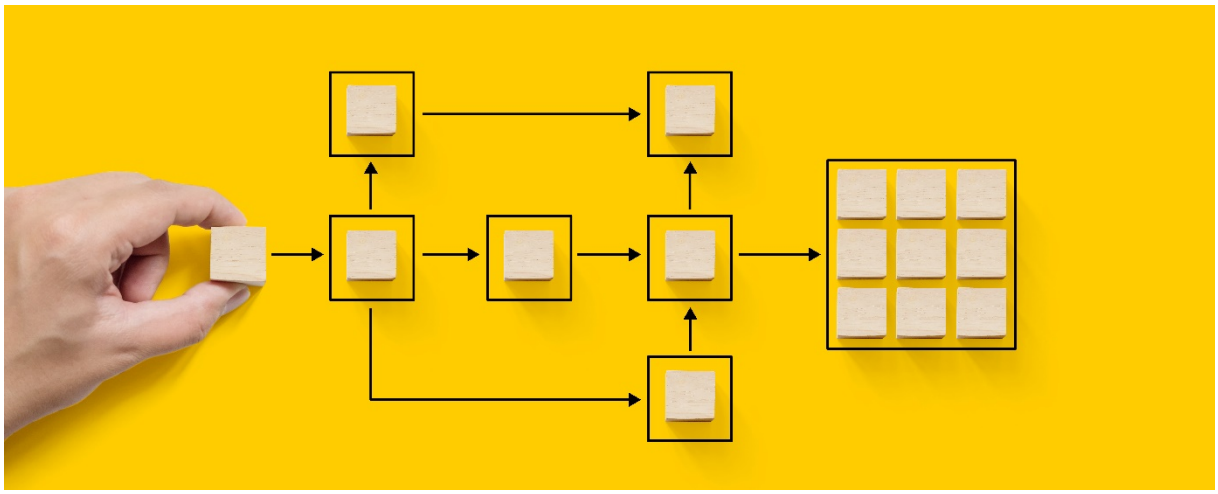
O principal objetivo da análise de requisitos é organizá-los para que possam ser tratados de maneira mais concreta e menos abstrata, de acordo com as necessidades dos usuários.

Vejamos, a seguir, um exemplo de requisitos abstratos que precisam ser descritos como tarefas.

Considere estes dois requisitos levantados junto às partes interessadas para “controlar as entradas e saídas de pessoas de uma empresa”, e “validar o CPF do visitante”. O primeiro não explica como o processo de controle de pessoas será efetuado, que pessoas serão autorizadas, quais as restrições de acesso, idade, horários etc. O segundo sugere uma validação técnica sobre o CPF de cada visitante, provavelmente, comparado a um CPF válido; porém, é necessário verificar se a respectiva pessoa é autorizada, ou se precisa validar mais informações? Tampouco está totalmente descrito e completo: não define as tarefas para essa validação.

A importância dessa análise e decomposição – ou síntese de informações de forma a descrever tarefas – se deve ao desenvolvimento de *software*, que precisa converter os comportamentos em ações de processamento, interagindo com os usuários. A Figura 4 apresenta fluxos de decomposição e síntese.

Figura 4 – Fluxos de decomposição e síntese



Crédito: Monster Ztudio/Shutterstock.

O escopo do projeto que foi definido pelas partes interessadas deve ser constantemente avaliado, e decisões devem ser tomadas sobre quais tarefas devem ser desenvolvidas pelo *software* para a melhor adequação às necessidades de negócio do cliente. Vale lembrar que as tarefas identificadas

são existentes e de conhecimento dos usuários e partes interessadas, e elas deverão ser incorporadas, total ou parcialmente, pelo *software* a ser produzido. Em muitos casos, o *software* acabará substituindo pessoas devido a sua capacidade de execução das tarefas e rapidez. No entanto, nem todas as tarefas podem ser transferidas para o meio digital.

Os fluxos operacionais podem – e são –, na maioria dos casos, alterados pelo *software*, causando inovações, melhorias, agilidade maior, entre outros benefícios da informatização de processos operacionais.

TEMA 4 – MODELOS PARA REFINAMENTO

Modelagem é o processo de criação de modelos que são esboços de uma solução – no caso, um *software*. Cria-se esse esboço para termos ideia do todo e do encaixe de suas partes. Se for necessário compreender melhor o esboço, efetua-se o refinamento com mais detalhes.

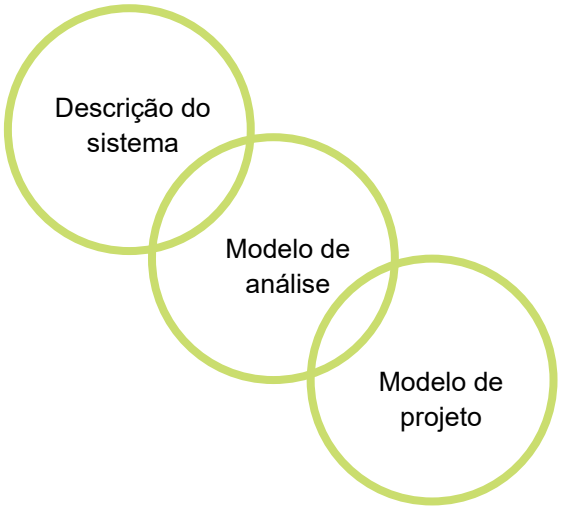
Segundo Pressman e Maxim (2018), nas últimas décadas foram identificados problemas de análise de requisitos e suas causas, o que possibilitou o desenvolvimento de vários métodos de modelagem de requisitos para resolver essas questões.

O objetivo da modelagem de requisitos é representar o que o cliente solicitou de várias maneiras, para que seja projetada uma solução coerente e que possa ter validade após a construção.

Na modelagem de requisitos, o foco principal está em “o quê”, e não em “como”. Ou seja, descrever os tipos de interação do usuário, quais objetos o sistema deverá manipular, que funções executará, quais interfaces serão definidas, e quais as restrições que se aplicam, de modo a atender ao que o cliente precisa e solicitou.

Resumindo, o modelo de requisitos deve alcançar três objetivos principais: descrever o que o cliente solicita, estabelecer informações bem definidas para o projeto do *software* e definir os requisitos que serão validados assim que este for construído. O modelo de análise mostra o sistema como um todo: o negócio a ser atendido pelo *software*, e o projeto do *software* descreverá a arquitetura, interfaces do usuário e todos os componentes do *software*, como mostrado na Figura 5.

Figura 5 – Modelagem de requisitos



Crédito: Pavan Rattmann/Arte UT.

O modelo de análise deve ser uma ponte entre a descrição do sistema e o projeto, e deve ser claro o suficiente para todos os envolvidos na solução, agregando valor para todos. Isso significa que o modelo deve exibir todas as necessidades tanto dos usuários, quanto dos desenvolvedores, porém, de forma simples e com clareza.

Pressman e Maxim (2021) sugerem alguns princípios para se ter em mente ao modelar e refinar requisitos.

- O domínio de informações de um problema deve ser representado e compreendido, englobando os dados (informações) do sistema, como eles fluem para fora e como são armazenados e mantidos pelo sistema.
- As funções executadas pelo *software* devem ser definidas, pois são as partes que oferecem aos usuários os serviços de que eles precisam, pois são visíveis e produzem resultados. Além disso, as funções podem transformar dados e controlar elementos do sistema.
- O comportamento do *software* e seus eventos externos devem ser representados. O *software* interage com o ambiente externo ao sistema, recebe dados pelos usuários ou por outros sistemas, os quais fazem o *software* se comportar de uma certa maneira (de acordo com as necessidades do cliente).
- Os modelos que representam informação, função e comportamento devem ser divididos de modo a revelar detalhes em camadas. A modelagem de requisitos deve possibilitar descrever os problemas

complexos de maneira mais simples. Assim, um problema grande e complexo deve ser dividido em subproblemas, até que cada parte seja claramente compreendida. Essa é uma estratégia muito importante na modelagem de requisitos, fornecendo bases para o projeto da solução.

- A análise deve partir da informação essencial para os detalhes da implementação, assim o problema deve ser descrito sem levar em consideração como será implementada a solução.

Ao aplicar esses princípios, o analista de requisitos terá sistematizado o problema que precisa resolver. E como efetuar uma modelagem de requisitos na prática? Há uma variedade grande de modelos e escolhas que o analista de requisitos deve avaliar e considerar, como que modelos aplicar para que os requisitos sejam bem descritos e estejam completos, claros e consistentes. É provável que nem todos os modelos sejam utilizados em um mesmo projeto. A escolha poderá combinar mais de um modelo, com visões e perspectivas diferentes mas que se completam, permitindo o entendimento por parte de todos os envolvidos.

Alguns usuários podem ter mais ou menos facilidade para entender diagramas, por exemplo; outros vão preferir avaliar um protótipo navegável; outros, se sentirão mais confortáveis lendo as descrições em formato de texto. A Figura 6 exibe os tipos cognitivos de usuários para que o analista de requisitos avalie o melhor tipo de modelagem.

Figura 6 – Tipos cognitivos dos usuários

Fonte Vasquez; Simões, 2016.

O modelo tem o papel de apresentar as informações em diferentes perspectivas, diminuindo a complexidade e auxiliando o analista de requisitos a perceber com mais facilidade pontos falhos em sua especificação. Cabe ao analista de requisitos entender o cliente e a melhor forma de comunicação.

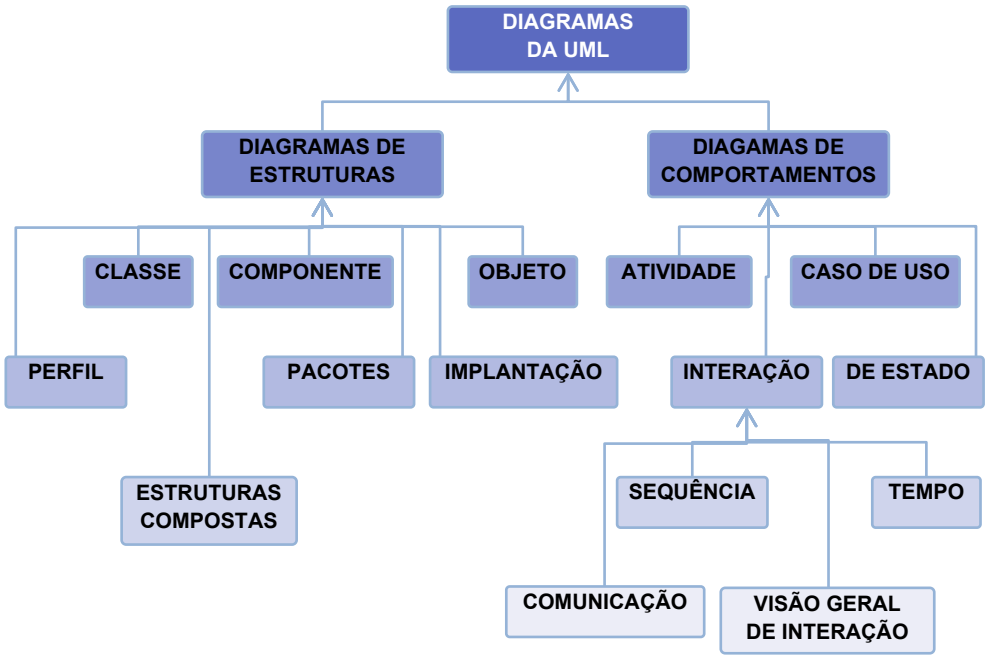
De acordo com Pressman e Maxim (2021), podemos citar alguns tipos de modelagem de requisitos mais utilizados que ajudarão a perceber os requisitos e suas prioridades:

- modelagem baseada em cenários;
- modelagem baseada em classes;
- modelagem funcional; e
- modelagem comportamental.

Na modelagem baseada em **cenários**, estes são criados com o objetivo de entender como os usuários interagem com o sistema e outros elementos fora da fronteira da solução proposta. Abordaremos a UML (Booch; Rumbaugh; Jacobson, 2012) para modelagem de requisitos e criação de cenários na forma de diagramas de casos de uso, de atividades e de sequência.

A UML (do inglês *Unified Modeling Language*, ou "Linguagem de Modelagem Unificada", em português) é uma notação gráfica para modelagem de *software*. A linguagem define um conjunto de diagramas para documentar e ajudar no design de sistemas de *software*, como podemos observar na Figura 7.

Figura 7 – Diagramas da Linguagem de Modelagem Unificada (UML)



Crédito: Pavan Rattmann/Arte UT.

Segundo Pressman e Maxim (2021), a UML é o resultado de um esforço para unificar as notações gráficas que surgiram no final da década de 1980 e início da de 1990, e passou a ser um padrão gerenciado pela *Object Management Group* (OMG, ou “Grupo de Gerenciamento de Objetos”), que é uma organização de padronização financiada por indústrias de *software*.

A utilização da UML para modelagem de requisitos tem início com a criação de cenários sob a forma de diagramas de casos de uso, diagramas de atividades e diagramas de sequência.

Os casos de uso são usados para representar as etapas lógicas que precisam ser consideradas durante o ciclo de desenvolvimento do *software*, de modo que se entenda a interação do usuário com os objetos do sistema. Um ator representa os papéis que os usuários (ou outros sistemas) do caso de uso desempenham dentro do cenário do usuário. Eles capturam o objetivo de uma ação – o evento acionador que inicia um processo – e, em seguida, descrevem cada etapa desse processo, incluindo entradas, saídas, erros e exceções. Os casos de uso geralmente são escritos na forma de um ator ou usuário executando uma ação, seguida pela resposta esperada do sistema e resultados alternativos. O conjunto de casos de uso representa todas as interações possíveis que serão descritas nos requisitos do sistema (Sommerville, 2018), e são descritos gráfica ou textualmente.

O levantamento dos requisitos fornece as informações necessárias para iniciar a descrição dos casos de uso, e as reuniões com os *stakeholders* identificam os envolvidos (atores), definem o escopo, prioridades, requisitos funcionais e todos os objetos que deverão ser manipulados pelo sistema. Com essas informações, pode ser criado um caso de uso preliminar, como na Figura 8.

Figura 8 – Exemplo de sequência ordenada de ações para um caso de uso

1. O proprietário faz login no *site Sistema Casa* com seu ID;
2. O proprietário insere a senha com 8 caracteres;
3. O sistema mostra os botões de todas as

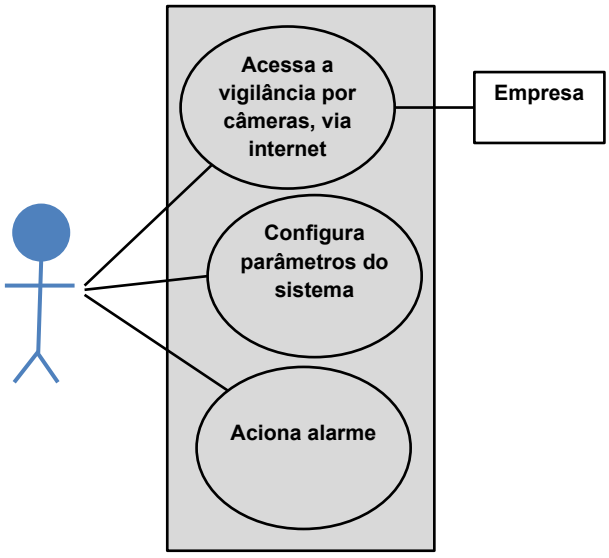
funções;

4. O proprietário seleciona a função "vigilância" clicando no botão;
5. O proprietário seleciona "escolher câmera";
6. O sistema mostra a planta da casa;
7. O proprietário seleciona o ícone da câmera da planta da casa;
8. O proprietário seleciona "visualização";
9. O sistema mostra uma janela de visualização identificada pelo ID da câmera

Crédito: Pavan Rattmann/Arte UT.

Assim que o analista de requisitos tiver mais detalhes, poderá refinar o caso de uso e enumerar as funções ou atividades realizadas por um ator específico, e listar a sequência de ações do usuário e as exceções. A Figura 9 mostra um exemplo de uma sequência ordenada de ações de usuário para uma determinada função do sistema com o ator “proprietário”.

Figura 9 – Exemplo de diagrama de caso de uso preliminar (alto nível)



Crédito: Pavan Rattmann/Arte UT.

Sommerville (2018) alerta para o fato de que nem sempre um caso de uso é de fato interessante para essa fase de análise de requisitos, pois são muito refinados e os *stakeholders* podem não compreender o termo “caso de uso”, não

acharem útil um modelo gráfico, ou não estarem muito interessados em uma descrição detalhada de cada interação do sistema. Segundo esse autor, os casos de uso seriam mais úteis no projeto de sistemas, e não na engenharia de requisitos. Mais uma vez, cabe ao analista de requisitos conhecer seu cliente e suas principais necessidades, e escolher uma abordagem que seja útil e eficiente nesta fase.

Na modelagem baseada em **classes**, o primeiro passo é identificar as classes de análise. Em UML, uma classe representa um objeto ou um conjunto de objetos que compartilham uma estrutura e um comportamento comuns e, no mundo do usuário, as classes podem ser identificadas observando os cenários criados para o sistema a ser construído, avaliando sintaticamente os casos de uso desenvolvidos. Geralmente, a palavra (um substantivo) dos requisitos listados e ações enumeradas, representa as classes do sistema. Nesse ponto, é importante lembrarmos de que estamos trabalhando com requisitos, e algumas classes podem ser importantes para a solução (domínio da solução), e não para o problema (domínio do problemas). Este último é o que descreve o interesse nessa fase de requisitos. Observe, a seguir, as dicas para identificar uma classe de análise.

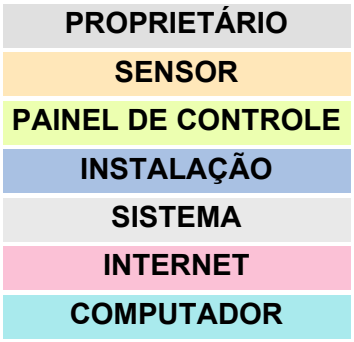
- Entidades externas que produzem ou consomem informações a serem usadas por um sistema (ex.: outros sistemas, dispositivos, pessoas).
- Coisas que fazem parte do domínio de informações para o problema (ex.: relatórios, letras, sinais).
- Ocorrências ou eventos que ocorrem no contexto da operação do sistema (ex.: transferência de conta, conclusão de um cálculo tributário).
- Papéis desempenhados por pessoas que interagem com o sistema (ex.: gerente, engenheiro, vendedor).
- Unidades organizacionais relevantes para o sistema (ex.: equipe, setor de almoxarifado, divisão da empresa).
- Locais do contexto do problema e função geral do sistema (ex.: pátio de carga, fábrica).
- Estruturas que definem uma classe (ex.: sensores, computadores, veículos).

Sommerville (2018) demonstra, por meio da Figura 10, como identificar classes considerando uma análise sintática (os substantivos sublinhados) para uma descrição do sistema de vigilância citado acima:

> A função de segurança domiciliar do sistema permite que o proprietário do imóvel configure o sistema de vigilância quando ele é instalado, monitore os sensores conectados ao sistema e interaja com o proprietário por meio da internet, computador ou painel de controle. (Sommerville, 2018, grifos do autor)

No texto citado, as palavras sublinhadas representam os substantivos e, com base nessa análise, podemos propor as seguintes classes:

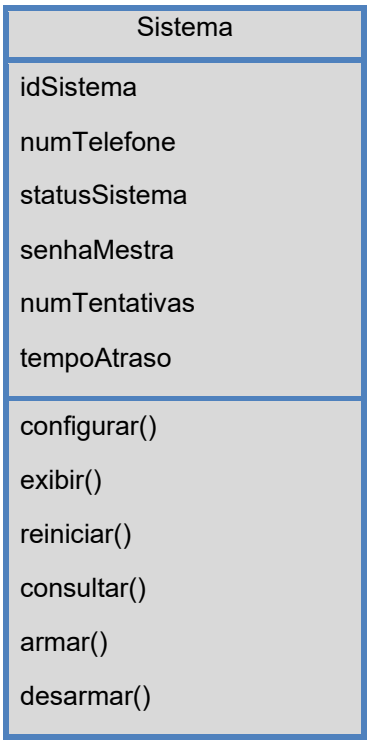
Figura 10 – Exemplo de classes identificadas



Crédito: Pavan Rattmann/Arte UT.

Atributos descrevem a classe, esclarecem o que a classe representa no domínio do problema, e as operações definem o comportamento de um objeto referenciado pela classe. Ao criar um conjunto de atributos que fazem sentido para uma classe de análise, obtemos também as operações que serão executadas pelos objetos referenciados pela classe. A Figura 11 exibe um exemplo de definição de classe.

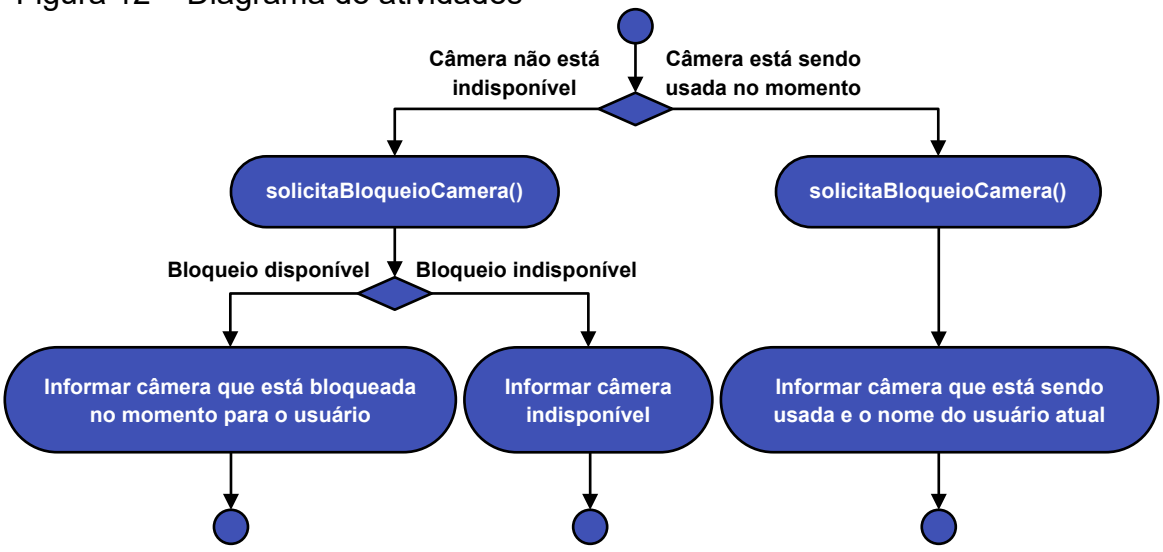
Figura 11 – Diagrama de classe para o sistema exemplo



Crédito: Pavan Rattmann/Arte UT.

A modelagem **funcional** avalia as funcionalidades percebidas pelo usuário que o sistema fornecerá, e as operações contidas nas classes de análise, ou seja, o processamento a ser realizado por elas. Nesse nível de análise dos requisitos, os diagramas de atividades devem ser usados apenas quando a funcionalidade é complexa. A Figura 12 apresenta um exemplo de diagrama de atividades.

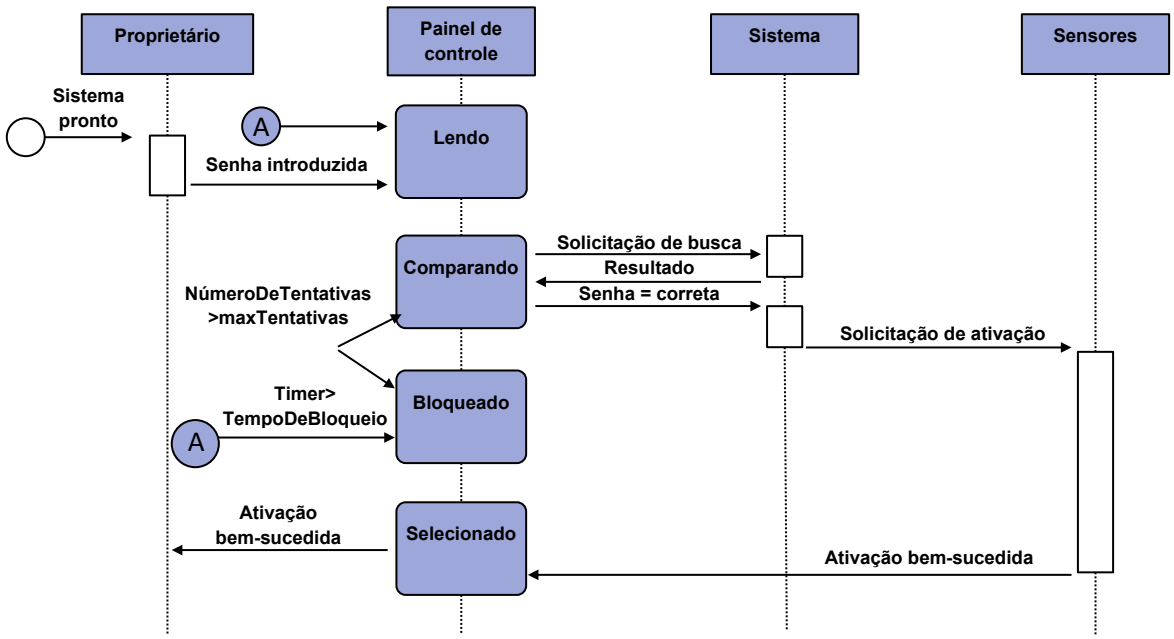
Figura 12 – Diagrama de atividades



Crédito: Pavan Rattmann/Arte UT.

O diagrama de sequência pode ser utilizado na modelagem comportamental, de forma a mostrar os eventos que provocam transições de objeto para objeto. Após identificar os eventos pelo caso de uso, faz-se a representação de como os eventos provocam o fluxo de um objeto para outro em função do tempo. A Figura 13 ilustra um diagrama de sequência. Essas informações são úteis na criação de um projeto eficaz para o sistema a ser construído.

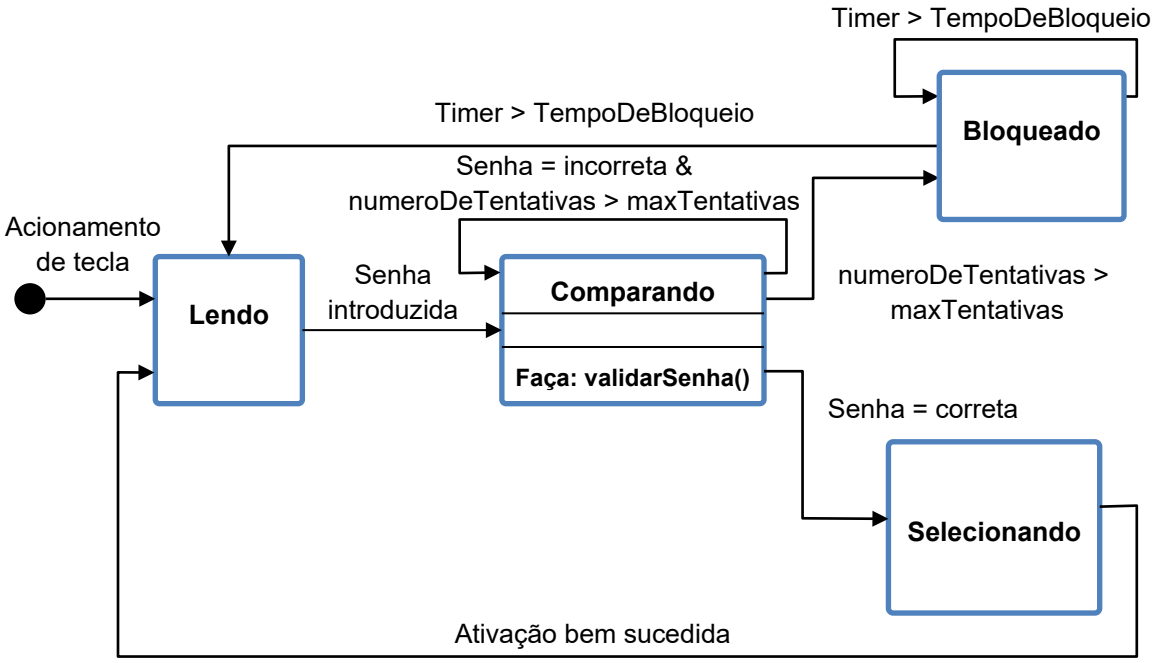
Figura 113 – Exemplo de diagrama de sequência



Crédito: Pavan Rattmann/Arte UT.

Um modelo **comportamental** indica como o *software* responde a estímulos ou eventos internos ou externos, os quais podem ser modelados pelo diagrama de estados da UML. O caso de uso descreve uma sequência de atividades entre os atores e o sistema e, normalmente, um evento ocorre toda vez que o ator e o sistema trocam informações. Uma ação ocorre concomitantemente com uma transição de estado e como consequência dele, envolvendo pelo menos uma operação. Cada troca de evento é um estado, e é representado como mostrado na Figura 14.

Figura 14 – Exemplo de diagrama de estados



Crédito: Pavan Rattmann/Arte UT.

Para desenhar os diagramas propostos pela UML, existem ferramentas chamadas de ferramentas CASE (do inglês *Computer-Aided Software Engineering*, ou, em português, "engenharia de *software* auxiliada por computador"), pagas ou de livre uso. Com uma rápida pesquisa na internet, podemos identificar algumas para nossos estudos de modelagem de requisitos com diagramas UML.

A cada modelo produzido, o analista de requisitos percebe melhor os requisitos a serem contemplados ou não nas primeiras versões do sistema, ou que são prioritários, ou não. O analista de requisitos utilizará os modelos que melhor representarem o contexto de todos os requisitos elicitados.

TEMA 5 – VERIFICAÇÃO E VALIDAÇÃO DE REQUISITOS

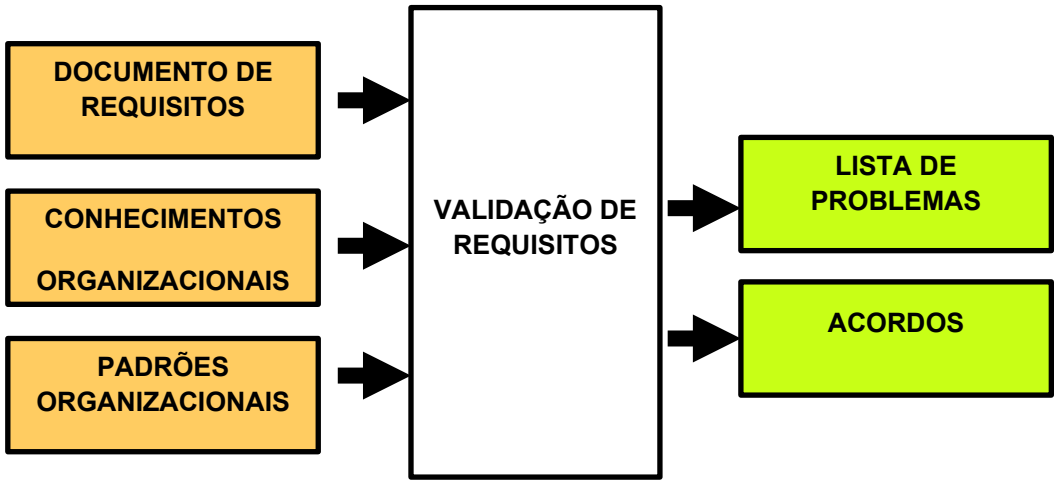
A validação de requisitos deve confirmar que o documento de requisitos está completo e correto para o sistema ser desenvolvido. A verificação dos requisitos é uma etapa inicial na qual se analisa a completude e a consistência, se os padrões da organização são atendidos, se há requisitos conflitantes ou ambíguos, ou algum tipo de erro técnico. Assim, pensamos em duas partes: a verificação e a validação. Verificar requisitos é avaliar se existem falhas na especificação, comparando os produtos da modelagem e da especificação,

considerando o escopo, e apontar as não conformidades que devam ser corrigidas ou justificadas. Geralmente, essa verificação é realizada por outra pessoa da equipe, e não o próprio analista de requisitos, autor da especificação. Isso ajuda na qualidade da verificação. A validação de requisitos procura saber se a especificação atende ao cliente e exige a participação dele no processo.

Para entendermos a diferença entre análise e validação, podemos pensar em análise como a etapa que trabalha com as informações que foram elicitadas dos *stakeholders*, ou seja, os dados "crus". A principal pergunta que se faz nesta fase é: "Temos os requisitos certos?". Na etapa de validação, é utilizado o documento final de requisitos, após estes serem analisados, priorizados e negociados. A pergunta nessa fase é: "Temos certos os requisitos?".

Para iniciar a etapa da validação (como mostrado na Figura 15), são necessários: o documento de requisitos, concluído e formatado com os padrões da organização; o conhecimento dos *stakeholders* da organização, contemplando informações implícitas; e, os padrões da organização (cada empresa tem suas características e padrões a serem seguidos, e o novo *software* deve incorporar esses padrões). Nessa etapa de validação, são encontrados problemas, erros e dúvidas que devem ser listados para novas conversas com os *stakeholders*; esse é o processo para validar, de fato, os requisitos elicitados, e os acordos são as ações que foram combinadas e acordadas em reposta aos problemas encontrados sobre os requisitos.

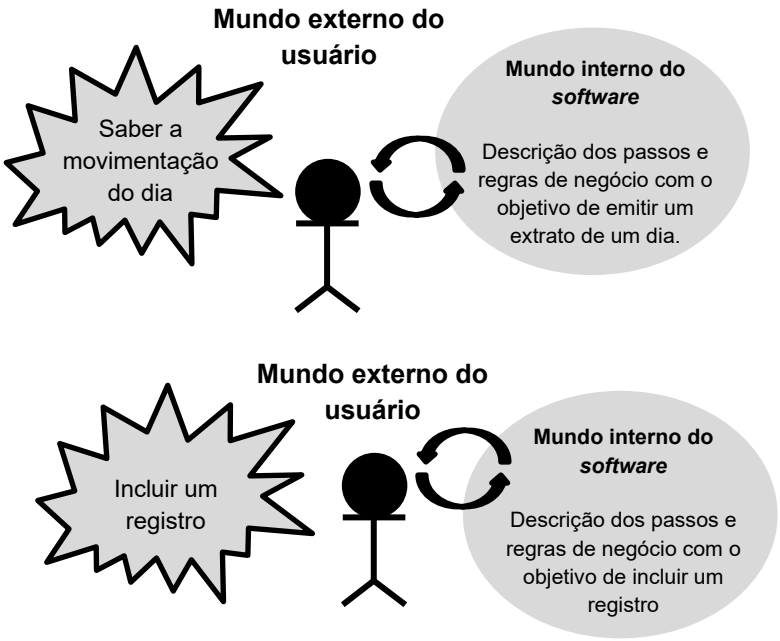
Figura 15 – Entradas e saídas da validação de requisitos



Crédito: Pavan Rattmann/Arte UT.

A verificação de requisitos reúne todos os produtos do trabalho realizado (as memórias do levantamento), os eventos externos que interagem com o *software*, que resultados o *software* deve produzir, como é a interação do usuário com o *software*, e todas as ações esperadas. Além disso, a verificação permite avaliar se o escopo está sendo atendido pela descrição dos requisitos, se todos os requisitos funcionais estão descritos com todas as possibilidades e quais ações a serem tomadas e, em caso de falhas, quais os fluxos alternativos. Essas descrições serão primordiais para o desenvolvedor entender apropriadamente o comportamento exigido para o *software*. A Figura 16 ilustra eventos para os quais o *software* deve prover respostas.

Figura 16 – Exemplo de eventos que o *software* deve atender



Crédito: Pavan Rattmann/Arte UT.

Para entender melhor, avalie o exemplo a seguir.

Requisito descrito: “O gerenciador de tarefas deve fornecer mensagens de *status* a intervalos regulares e não menos que a cada 60 segundos”.

Problemas encontrados na validação deste requisito:

- Quais são as mensagens de *status*?
- Quais as condições para fornecer a mensagem?
- Como as mensagens são fornecidas?
- Por quanto tempo as mensagens são exibidas?

Avaliando o exemplo, o analista de requisitos deve providenciar respostas para completar a descrição do requisito.

5.1 Verificação de requisitos

Uma técnica bastante comum aplicada na verificação de requisitos é o uso de um *checklist*, ou seja, uma lista de perguntas a serem avaliadas e respondida com o objetivo de verificar se os requisitos estão sendo atendidos de forma adequada.

Vasquez e Simões (2016) indicam um exemplo de lista de verificação sobre um diagrama de contexto:

1. Verificar se a identificação do conceito de negócio está correta;
2. Verificar se o conceito de negócio é externo à solução proposta;
3. Verificar se o conceito de negócio é necessário;
4. Verificar se o quadro de requisitos de armazenamento está completo.

Ainda segundo Vasquez e Simões (2016), o motivo pra se usar listas de verificação é aumentar a qualidade do trabalho entregue e identificar de forma objetiva o que foi elicitado.

Outra técnica é a Medição de Pontos de Função, que avalia as funcionalidades, esforço e custo de um projeto, e beneficia diretamente o planejamento e o acompanhamento do projeto. Para a verificação de requisitos, a utilidade da medição é o efeito colateral, ou seja, a identificação de diversas falhas na especificação de requisitos, como consistência, clareza e completude.

5.2 Validação de requisitos

A validação de requisitos é uma garantia de qualidade que busca assegurar que todos os requisitos especificados estejam alinhados com os requisitos de negócio, e o escopo delimitado para o projeto. A finalidade dessa etapa da engenharia de requisitos é garantir que a definição do produto a ser desenvolvido está correta e atende às necessidades dos *stakeholders*.

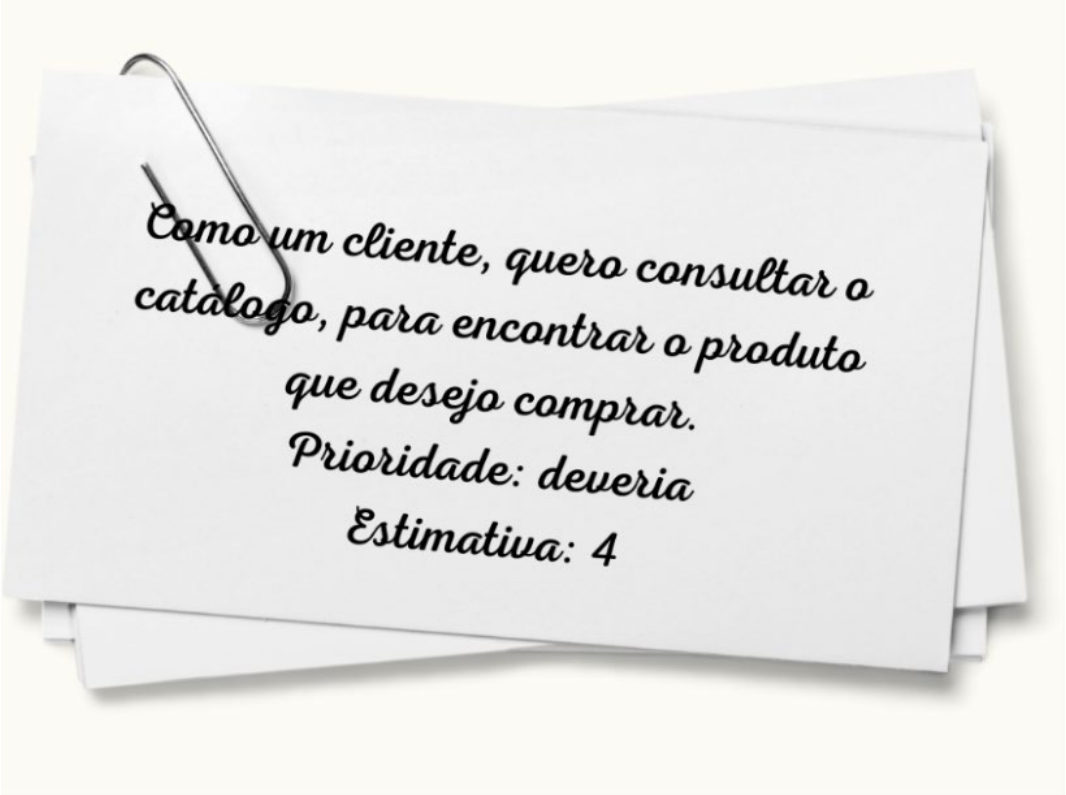
Uma preocupação básica na etapa de validação é a consistência, isto é, se todos os requisitos foram declarados de forma não ambígua, que as inconsistências, omissões e os erros tenham sido detectados e corrigidos, e que os artefatos produzidos na especificação de requisitos estejam de acordo com os critérios e padrões da organização estabelecidos para o processo.

A literatura elenca várias técnicas para validação de requisitos. A seguir, abordaremos algumas das mais utilizadas.

5.2.1 História do usuário

Uma técnica utilizada é a *história do usuário*, uma descrição do que o sistema deve fazer para o usuário. Essa técnica é muito adotada para projetos que utilizam a metodologia ágil que estudaremos mais adiante. A Figura 17 exemplifica um exemplo de história do usuário.

Figura 17 – História do usuário



Fonte: Vasquez; Simões, 2016.

5.2.2 Modelagem de processo

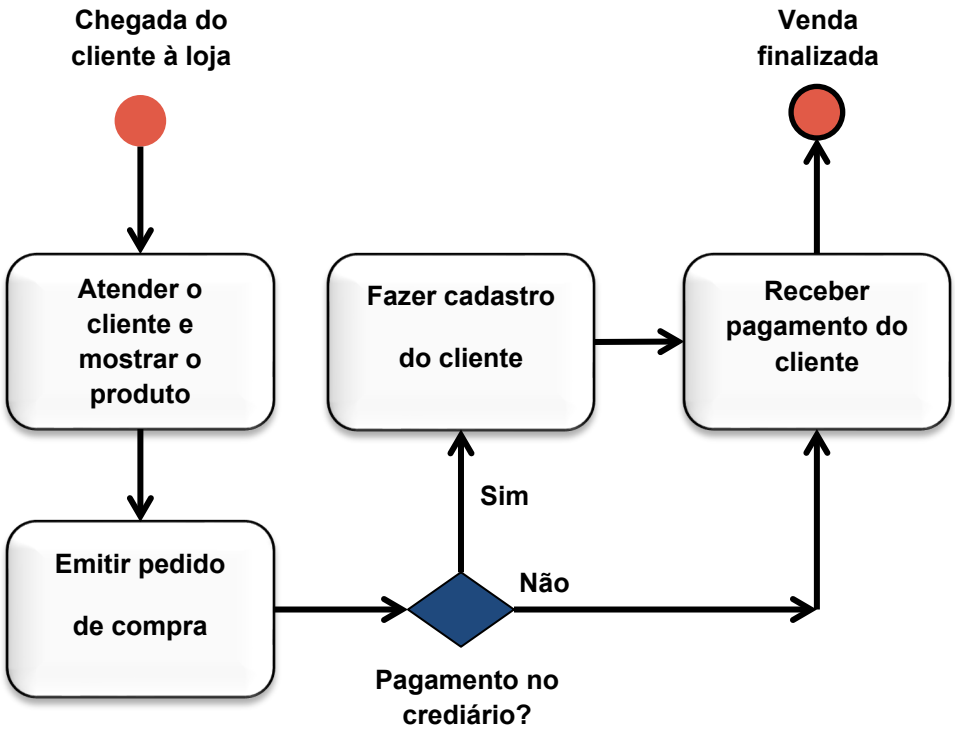
Outra técnica muito útil para a fase de validação de requisitos é a modelagem de processos. Por meio de uma representação de processos, é possível organizar os requisitos em tipos de negócios e em diferentes níveis, facilitando sua interpretação e leitura.

Devemos entender que processo é um conjunto de atividades ordenadas e encadeadas que devem ser executadas para atender a um determinado propósito, com objetivo, início, fim, e entradas e saídas bem definidas. A

modelagem de processos visa criar uma representação do funcionamento do processo, com todo o detalhamento necessário. Essa modelagem fornece ao analista de requisitos um entendimento sobre como as atividades são realizadas e, com base nesse conhecimento, consegue determinar os requisitos técnicos necessários para que uma solução de *software* atenda de forma completa às necessidades do usuário.

Um modelo implica na representação de um determinado estado do negócio (Vasquez; Simões, 2016) e dos recursos envolvidos, representando o funcionamento do que está sendo modelado, considerando o seu comportamento. A UML estabelecida pelo OMG criou padrões de modelos que podem ser utilizados na modelagem de processos de negócio. Essa documentação permitirá analisar se o processo está sendo atendido pelos requisitos propostos por meio da simulação do funcionamento de um protótipo, por exemplo, ou da análise das atividades descritas. A Figura 18 apresenta um exemplo de modelo de processo de venda.

Figura 18 – Modelo de processo de venda



Crédito: Pavan Rattmann/Arte UT.

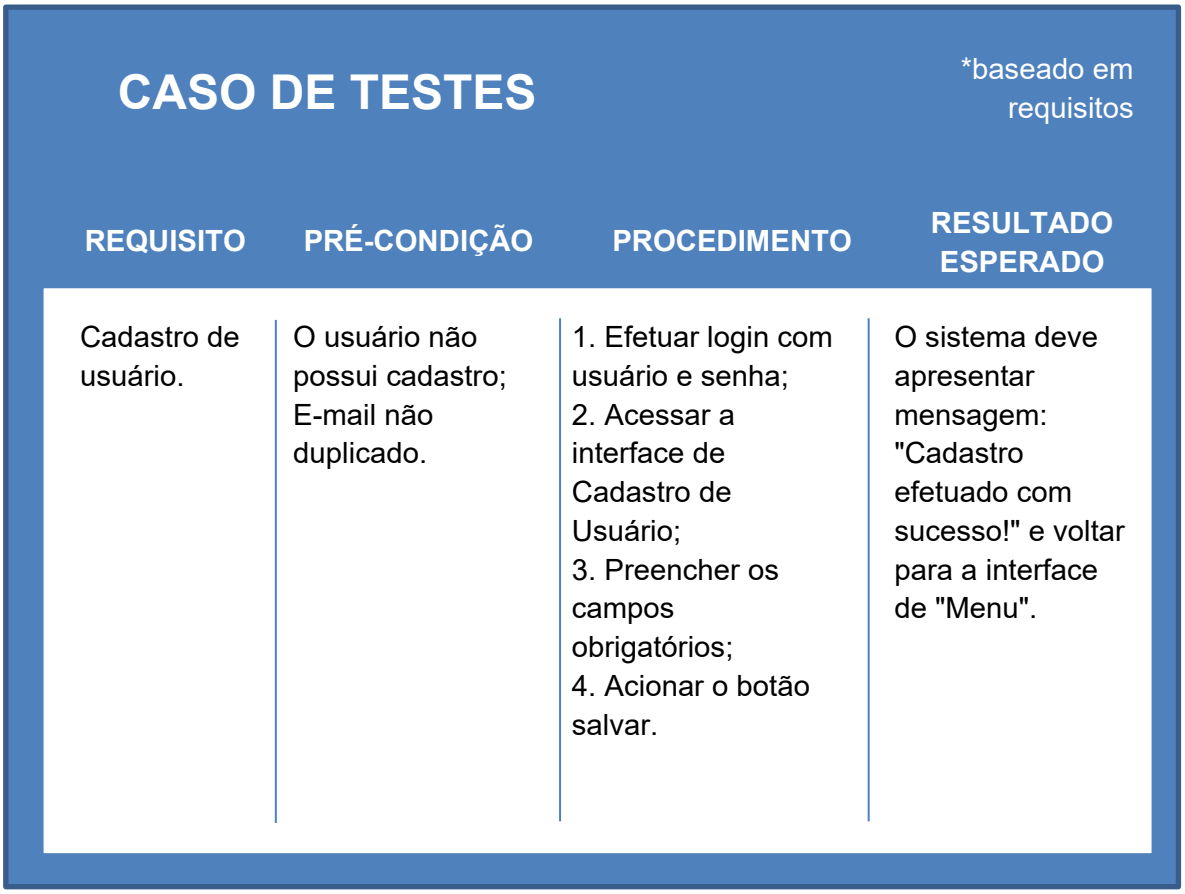
5.2.3 Geração de casos de testes

Os requisitos devem ser testáveis, para permitirem identificar problemas não resolvidos e requisitos de difícil implementação. Além disso, é possível desenvolver testes com base nos requisitos de usuário antes de qualquer código ser escrito.

Muitas vezes, a equipe de testes participa da especificação de requisitos para elaborar os casos de testes. Cada requisito, ou conjunto deles, atende a um determinado processo, e os casos de testes comprovam sua eficiência, demonstrando se o *software* realmente atenderá às necessidades do cliente.

Essa é uma ferramenta poderosa de verificação de requisitos, que ajuda a melhorar sua qualidade (Vasquez; Simões, 2016). Todo requisito funcional deve ser associado a um caso de teste. A Figura 19 exibe um exemplo de caso de testes elaborado com base em uma especificação de requisitos.

Figura 19 – Exemplo de caso de teste de requisito



Crédito: Pavan Rattmann/Arte UT.

5.2.4 Listas de verificação (*checklist*)

As listas de verificação, também chamadas de *checklist*, são muito usadas em todas as áreas, como no caso de cirurgias, quando há uma série de verificações, de modo a garantir segurança ao paciente.

No contexto de engenharia de requisitos, o uso de listas de verificação e validação são muito úteis para garantir que os requisitos, além de atender às necessidades e estarem completos, possam ser implementados nos processos do cliente. Uma lista de verificação organiza o processo de análise daquele contexto, gerando perguntas encadeadas que, ao serem respondidas, podem indicar falhas no entendimento ou na descrição dos requisitos, ou ainda, requisitos faltantes. Cada *checklist* deve ser criado considerando o contexto de sua aplicação. É possível ainda haver vários deles para ajudar na validação de todos os requisitos definidos. A Figura 20 apresenta um exemplo de *checklist*.

Figura 220 – Exemplo de lista de verificação (*checklist*)

Crédito: Crédito: Pavan Rattmann/Arte UT; Yalcin Sonat/Shutterstock.

Essas técnicas podem ser utilizadas em conjunto, conforme o tamanho do projeto, e ajudam a aumentar a qualidade do produto a ser desenvolvido. Elas indicam elementos críticos, e também que condições devem ser avaliadas com maior cautela e atenção. Os problemas encontrados na validação de requisitos devem ser considerados e analisados, devendo ter proposta de solução, e nunca devem ser subestimados. No entanto, é raro encontrar todos os problemas de requisitos durante o processo de validação (Pressman; Maxim, 2021). Possivelmente, serão necessárias outras alterações. Assim, um bom processo de validação de requisitos minimiza erros futuros e aumenta as chances de um produto de qualidade.

FINALIZANDO

Nesta etapa, apresentamos os tipos de prototipação, validação e análise de requisitos, com o objetivo de facilitar a descrição de todos os requisitos. Vimos como criar protótipos que simulam o funcionamento do *software* para apresentar aos *stakeholders* pode ser útil e, assim, refinar os requisitos elencados. Uma vez elicitados os requisitos, conversamos sobre o processo de analisá-los como se estivéssemos montando um quebra-cabeças, no qual cada peça seria um requisito, e a conclusão seria todas as necessidades atendidas.

Percebemos que o processo da análise de requisitos tem por objetivo descobrir problemas, incompletude e inconsistências nos requisitos elicitados.

Com todas as informações, documentos e análises, conversamos sobre como os fluxos operacionais da organização podem ser alterados pelo *software* em desenvolvimento, e como preparar os usuários para esses novos processos. Estudamos modelos para refinamento dos requisitos, ou seja, como representar “o que” o cliente solicitou, para projetar uma solução correta após a construção do sistema. Estudamos que, na modelagem de requisitos, há várias formas de representação, com padrões internacionais, como a UML e seus diagramas, baseadas em cenários (casos de uso), entre outros. A cada modelo produzido e analisado, o analista de requisitos percebe e entende melhor os requisitos e consegue validá-los com o cliente.

Na última parte desta etapa, estudamos sobre a importância de revisar e confirmar todos os requisitos: se estão completos, corretos e atendem às necessidades do escopo contratado. Vimos como é importante utilizar técnicas diversas, como *checklist*, protótipos, história de usuário, modelos de processo e

casos de testes para verificar e validar requisitos, antes mesmo de iniciar o desenvolvimento da solução.

Por fim, avaliamos que, embora haja grande esforço para validações de requisitos, é comum surgirem mudanças no futuro. Por falar em mudanças, gerenciamento dos requisitos será o assunto de outra ocasião. Até lá!

REFERÊNCIAS

BOOCH, G; RUMBAUGH, J.; JACOBSON, I. **UML**: guia do usuário. Tradução: Fábio Freitas da Silva; Rio de Janeiro: Campus, 2012.

CASTRO, M. C. V. de. et al. Principais causas de fracasso em projetos. **PMKB**, 29 dez. 2015. Disponível em: <https://pmkb.com.br/artigos/principais-causas-de-fracasso-em-projetos/>. Acesso em: 30 mar. 2022.

IIBA – International Institute of Business Analysis. **Guia para o corpo de conhecimento de análise de negócios**: (Guia BABOK(r)) – Versão 2.0. Toronto, Canadá: IIBA, 2011

PRESSMAN, R.; MAXIM, B. R. **Engenharia de software**: uma abordagem profissional. 9. ed. Porto Alegre: Mc Graw Hill; Bookman, 2021.

SOMMERVILLE, I. **Engenharia de software**. 6. ed. São Paulo: Pearson, 2018.

VASQUEZ, C.; SIMÕES, G. **Engenharia de requisitos**: software orientado ao negócio. 1. ed. Rio de Janeiro: Brasport, 2016.