ESCOLA TÉCNICA ESTADUAL DE SÃO PAULO
ANÁLISE E PROJETO DE SISTEMAS

FELIPE FERREIRA DE SOUZA
FERNANDA YUMI ASSONUMA
GABRIEL LUÍS DA SILVA
GABRIEL YUMETO NAKAYA HASSEGAWA
HENRIQUE SANTANA PINHEIRO

ENGENHARIA DE REQUISITOS

SÃO PAULO
2021

FELIPE FERREIRA DE SOUZA
FERNANDA YUMI ASSONUMA
GABRIEL LUÍS DA SILVA
GABRIEL YUMETO NAKAYA HASSEGAWA
HENRIQUE SANTANA PINHEIRO

ENGENHARIA DE REQUISITOS

Trabalho Acadêmico apresentado para obtenção de menção bimestral, Análise e Projeto de Sistemas, Escola Técnica Estadual de São Paulo.

Orientador: Prof. Luís Ricardo

SÃO PAULO
2021

SUMÁRIO

1 INTRODUÇÃO

Com o avanço das tecnologias atuais e o aumento da presença delas nos mais diversos espaços, como em empresas, escolas, hospitais, entre outros, tem-se, consequentemente, um aumento da demanda e da necessidade dos meios digitais. Nesse âmbito, torna-se primordial o desenvolvimento de programas que conciliem a comunicação de aparatos eletrônicos com as funcionalidades e utilidades atreladas às pessoas e ao contexto em que se fazem presentes. Estes programas, os softwares, possuem intrínseca importância no mercado. À medida que o mundo digital se expande, sua presença nas empresas vem sendo cada vez mais intensificada.

A ampla possibilidade de funcionalidades presentes abre diversas intercepções, restritas através da análise das necessidades do contexto que cada software tem como base para ser elaborado. Deste modo, é fundamental a consciência das imposições de um cliente para o desenvolvimento de um programa.

O relatório CHAOS Report (THE STANDISH GROUP, 2015), analisou cerca de cinquenta mil projetos de software entre os anos de 2011 e 2015. Verificou-se que a porcentagem dos concluídos com sucesso, isto é, dentro do prazo e custo previstos, além de apresentar resultados satisfatórios, foi de apenas 29% no ano final do levantamento.

É necessário ter conhecimento das características do sistema a ser produzido desde o início de seu desenvolvimento, para obter-se, assim, um produto que tenha resultados satisfatórios para os consumidores e partes envolvidas (CHUNG; NIXON; et al., 2000 apud TAVEIRO, 2016). Frequentemente, a causa de resultados não conformes está relacionada à documentação equívoca ou a um não entendimento das funcionalidades necessárias para o software. Nesse contexto, surge a importância da Engenharia de Requisitos, responsável pela análise, documentação, levantamento de requisitos, além de outras funções, de clientes para o desenvolvimento de softwares.

Portanto, questiona-se: haja vista os diversos modos em que pode ser fragmentada, como decorre a Engenharia de Requisitos e por que corresponde à uma fração tão relevante da produção de um sistema e, consequentemente, da Engenharia de Software?

Este trabalho tem como objetivo geral apresentar os conceitos da Engenharia de Requisitos, interligando as propostas de conceitos já apresentadas. Com isso em mente, tem-se como objetivos específicos identificar seu papel na elaboração e produção de um software, além de conceituar e exemplificar as etapas a partir de modelos fictícios e a demonstração de ferramentas e métodos, para assim obter-se uma compreensão completa do tema.

Foi desenvolvida uma pesquisa descritiva com abordagem qualitativa, a fim de definir os termos e etapas da Engenharia de Requisitos e sua importância no desenvolvimento de softwares. O método utilizado corresponde ao fenomenológico, cujos procedimentos aplicados foram o bibliográfico e documental, através de pesquisas em livros, teses, monografias e sites online, atreladas a análise de dados e números estatísticos.

A partir desse momento, deduz-se que a Engenharia de Requisitos deve ser um processo restringido em etapas que, se seguidas, obtém como resultado o desenvolvimento de um sistema satisfatório para aqueles que o contrataram. Se há uma padronização, um melhor entendimento e organização dos requisitos de um projeto, então, é possível concluir que há um declínio de eventuais motivos para crises de software.

O texto está organizado em quatro capítulos. O segundo define a Engenharia de Requisitos, a trajetória e os aspectos históricos de seu desenvolvimento e sua ligação com a Engenharia de Software. O terceiro capítulo descreve o que são os requisitos, além de definir o que são requisitos funcionais e não funcionais, representando suas diferenças e subtipos e dar importância que possuem em um projeto. Já o quarto detalha e interliga as etapas do processo da Engenharia de Requisitos por meio de conceitos e definições. São definidos nessa parcela desta pesquisa as etapas e os subtemas da Elicitação de Requisitos; Análise de Requisitos; Documentação de Requisitos; Validação; e da Gerência de Requisitos. E, por fim, conclui-se com o quinto capítulo, que apresenta as considerações finais.

2 ENGENHARIA DE REQUISITOS

O processo de produção de um software conta com muitas fases em suas diversas atividades. Através da disciplina chamada Engenharia de Softwares busca-se otimizar e aperfeiçoar este processo, promovendo, através de fundamentos científicos e técnicas sistemáticas, um melhor custo-benefício e qualidade quanto ao resultado final do projeto. O sucesso de determinado sistema de software está diretamente ligado ao nível de satisfação do cliente ao final, tendo estes seus requisitos e necessidades para o software alcançadas (DIDIER, 2004, p. v).

Segundo Sommervile (2011, p. 57) Engenharia de Requisitos é a fase do processo na qual se pretende coletar, documentar e manter estes requisitos de forma sistemática. É o momento em que se deve buscar uma clara noção de o que é o problema, quais são os objetivos e necessidades do cliente em relação ao sistema, além das regras de negócio. Consiste em um conjunto de atividades que incluem a elicitação desses requisitos, a validação, documentação, gerenciamento, dentre outras.

Seu principal objetivo é permitir tanto a desenvolvedores quanto a clientes terem a mesma visão do projeto, entendendo as necessidades específicas, estabelecendo um consenso entre os interessados, documentando e gerenciando esses requisitos. Costuma-se usar o termo *stakeholder* para definir uma pessoa, organização ou qualquer outra entidade interessada no negócio e que detém o conhecimento necessário para a coleta desses requisitos. Esta será a fonte direta dos requisitos, e, portanto, deve ter domínio dos negócios e processos internos da organização. “Um *stakeholder* é uma pessoa ou papel que, de alguma maneira, é afetado pelo sistema.” (SOMMERVILE, 2011, p. 60)

As atividades realizadas na E.R. se caracterizam por uma concepção para além do universo do software, uma vez que ele está inserido em um contexto e possui uma função em seu meio. Este meio no qual está inserido é chamado de domínio de negócio e compreende à área de conhecimento e parte do mundo real na qual o problema se encontra – uma empresa, por exemplo. É imprescindível que, para uma melhor obtenção desses requisitos, haja um certo envolvimento com o domínio do cliente e uma boa comunicação com este, com um melhor conhecimento da organização e de seus processos internos, já que estes podem sofrer futuras mudanças, afetando o projeto. A sistematização desse conhecimento é denominada Modelagem de Organização, e serve de base para o levantamento de requisitos (PÁDUA apud PINOTTI, 2004, p. 2).

Além de tudo, essa disciplina deve ser acompanhada por uma análise da viabilidade de implementação do projeto, levando em conta aspectos como lucro, gastos, oportunidades e riscos, benefícios, orçamentos, dentre outros. Esta atividade é chamada de Estudo De Viabilidade.

Ao fim desse processo, quando feito de maneira correta, os desenvolvedores possuirão a informação necessária para a implementação do projeto, com todas as regras e restrições. A validação deste se dá quando todos os requisitos analisados são satisfeitos pelo projeto; neste caso, o sistema enfim é uma solução válida para o problema.

3 REQUISITOS

Um requisito representa qualquer condição ou capacidade que um sistema deve alcançar necessária para resolver um problema, alcançar um objetivo, satisfazer um contrato, padrão, especificação ou outro documento formalmente imposto (IEEE, 1990). Dentro do contexto da Engenharia de Software, representa funções e serviços que um software deve apresentar, aliados às restrições nas quais está inserido (SOMMERVILE, 2011).

É imprescindível que, antes de uma abordagem sobre os processos que envolvem a Engenharia de Requisitos, compreenda-se como estes requisitos são representados, classificados e organizados durante o processo. Diversos autores se utilizam de variadas abordagens em relação ao assunto; neste trabalho, foi optado por usar as ideias de Sommervile (2011) sobre o tema.

Os requisitos podem apresentar diferentes níveis de especificação e detalhamento. Pode-se usar o termo Requisitos de Usuário para designar os requisitos descritos de uma forma mais geral e menos detalhada, e Requisitos de Sistema, para aqueles descritos de forma mais específica. Geralmente, os requisitos de usuário aparecem em uma linguagem abstrata de alto nível, sendo possível o uso de diagramas, e deles podem se ramificar diversos outros requisitos de sistema mais específicos.

Os requisitos de sistema descrevem exatamente o que o sistema deve operar e quais são suas funções, informando tudo o que deve ser implementado. Essa é uma dicotomia proposta por Sommervile (2011), e é uma ideia muito importante de ser considerada, pois diferentes tipos de leitores dos requisitos requerem diferentes níveis de detalhamento destes. O que deve ser considerado é como estes requisitos serão utilizados por aquele leitor específico. Um gerente cliente não tem a necessidade de conhecer os mínimos detalhes de um programa da mesma forma que um usuário final que será diretamente afetado pelas funções de um sistema, ou um dos desenvolvedores. Observe nos exemplos a seguir alguns requisitos de um sistema acadêmico hipotético e perceba os diferentes níveis de detalhamento entre eles:

• O sistema deve gerar um relatório contendo as menções e faltas de todos os alunos ao fim do bimestre

• O sistema deve agrupar os alunos por área, curso, disciplina, período e turma

• O sistema só poderá ser operado por usuários autorizados

• O login de usuário deverá ser efetuado através do código de identificação e de senha

• O sistema deve operar durante os dias úteis da semana

• O sistema deve estar disponível entre as 07h e ás 18h

3.1 Tipos de requisitos

De forma geral, os requisitos são separados em dois tipos: os Requisitos Funcionais, que estão diretamente ligados às funcionalidades do sistema, e os Requisitos Não-Funcionais**,** que expressam restrições a essas funcionalidades**.** Em resumo, os RF definem o que o sistema fará, e os RNF, como ele fará.

Ambos são importantes para a especificação de um software e, na maior parte do tempo, são dependentes entre si, podendo um RNF gerar uma série de outros RF necessários para poder sem implementado. Além disso, a situação oposta também pode ocorrer, quando um requisito restringe um outro requisito.

3.2 Requisitos funcionais

Requisitos Funcionais traduzem demandas que estão diretamente ligadas às funcionalidades que o sistema deve apresentar. Especificam o que o sistema deve fazer e o que não devem fazer, por meio de suas tarefas e funções. São geralmente a maior parte dos requisitos e os mais fáceis de se descobrir. Também são chamados de requisitos comportamentais pois especificam as entradas do sistema, as saídas, e as relações de comportamento entre elas (TAVEIRO. 2016). Geralmente estão ligadas a um banco de dados.

Segundo Sommervile (2011), é importante que a especificação dos RF seja completa e consistente, isto significa que todos os requisitos especificados pelo cliente devem ser definidos, e devem ser especificados ao ponto de não apresentarem ambiguidades. Definições contraditórias sobre os requisitos podem gerar problemas quanto a implementação do projeto, podendo abrir espaço para uma interpretação pessoal do desenvolvedor que conflita com a real necessidade do cliente. De tal forma, é evidente a necessidade de uma ótima especificação dos requisitos do sistema.

Veja alguns exemplos de requisitos funcionais do mesmo sistema acadêmico tratado anteriormente:

• O sistema deve calcular a frequência de cada aluno por bimestre.

• O sistema deve permitir o cadastro de alunos

• O sistema deve permitir o cadastro de professores

• O sistema deve permitir a efetuação da rematrícula
• O sistema deve possuir um calendário escolar, com as datas do ano letivo vigente

Perceba que todos os requisitos estão diretamente ligados com alguma ação ou alguma funcionalidade que o sistema deve apresentar.

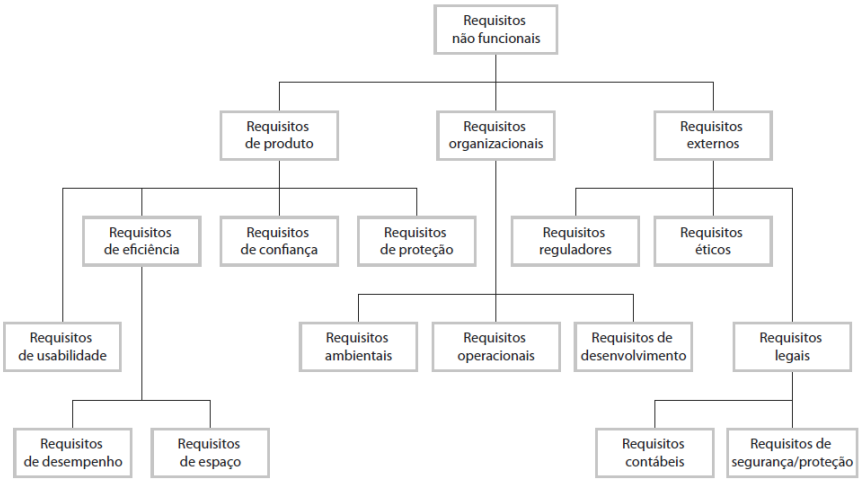
3.3 Requisitos não-funcionais

Os requisitos não-funcionais descrevem demandas que não estão relacionadas a uma funcionalidade específica do sistema, mas a características emergentes do sistema. Também chamados de atributos de qualidade, geralmente apontam as restrições sob as quais o sistema deve operar e as propriedades que ele deve apresentar. Alguns exemplos de RNF são aqueles que restringem a performance que o sistema deve possuir, a segurança, o tempo de resposta, a manutenibilidade, dentre outros.

De forma geral, os RNF podem ser mais difíceis de serem definidos e de serem relacionados a uma implementação do projeto, já que estão presentes no sistema como um todo e que a consolidação de algum requisito não-funcional possa exigir a implementação de uma nova funcionalidade. Além disso, são mais críticos, podendo inutilizar todo um sistema. Um requisito de segurança não atendido para um sistema bancário pode inviabilizar qualquer tentativa de uso por exemplo (SOMMERVILE, 2011).

Os RNF são classificados nas seguintes categorias: requisitos de produto, requisitos organizacionais e requisitos externos, além de suas respectivas subcategorias, como está descrito no seguinte diagrama na Figura 1.

Figura 1 - Requisitos Não-Funcionais



Fonte: (SOMERVILE, 2011, p. 61)

Os requisitos de produto surgem a partir de demandas que estão diretamente ligadas ao funcionamento do software. Podem representar restrições quanto a eficiência do sistema, como o tempo de resposta, ou a memória requerida; restrições ligadas à usabilidade, como o tempo de treinamento necessário para operá-lo com o mínimo de falhas possíveis, ou seja, a facilidade com que se pode usar o software; ou até mesmo a proteção que ele oferece.

Os requisitos organizacionais dizem respeito aos procedimentos presentes tanto na organização do cliente quanto na organização do desenvolvedor. Podem restringir, por exemplo, os processos de desenvolvimento daquele software, como a linguagem de programação a ser utilizada, os paradigmas, ambientes de desenvolvimento, dentre outros fatores. Podem definir também os processos que definem como o sistema será usado pela organização do contratante, ou o ambiente em que será utilizado, como o sistema operacional.

Já os requisitos externos são influenciados por fatores que não estão diretamente ligados ao sistema e seu ambiente em si. Alguns desses fatores podem ser requisitos éticos, requisitos reguladores, ou requisitos legais, que permitem que o software possa operar dentro da lei ou sob a aprovação de um órgão regulador, como o MEC por exemplo;

Para o mesmo sistema acadêmico, observe alguns exemplos de requisitos não-funcionais:

- O sistema deve possuir um tempo de resposta máximo de 10 segundos.
- O sistema deve seguir as diretrizes da LGPD.
- O sistema deverá operar em múltiplos sistemas operacionais.

• O sistema deve ser desenvolvido com a linguagem Java, utilizando-se de programação orientada a objetos.

• O sistema deve apresentar um número máximo de falhas de duas por hora de funcionamento.

• O tempo máximo de treinamento dos usuários deve ser de 3 horas.

• O design da interface visual do sistema deve seguir as diretrizes do manual de identidade visual da instituição de ensino.

• O sistema deverá apresentar performance aceitável em computadores de baixo rendimento.

• O tempo máximo de desenvolvimento de software deve ser de 7 meses.

Pode-se observar, nos exemplos acima, que os requisitos não-funcionais devem ser especificados de forma quantitativa sempre que possível. Esta necessidade acontece pelo fato de que muitas vezes esses requisitos são apresentados pelos usuários de forma ambígua, o que exige alguma interpretação por parte do desenvolvedor. Além disso, permite uma abordagem mais sistemática do problema e testes mais precisos. Veja na Tabela 1 algumas métricas que podem ser usadas para especificar requisitos não-funcionais.

Tabela 1 - Métricas para requisitos não-funcionais

Propriedade	Medida
Velocidade	Transações processadas/segundo Tempo de resposta de usuário/evento Tempo de atualização de tela
Tamanho	Megabytes Número de chips de memória ROM
Facilidade de uso	Tempo de treinamento Número de *frames* de ajuda
Confiabilidade	Tempo médio para falha Probabilidade de indisponibilidade Taxa de ocorrência de falhas Disponibilidade
Robustez	Tempo de reinício após falha Percentual de eventos que causam falhas Probabilidade de corrupção de dados em caso de falha
Portabilidade	Percentual de declarações dependentes do sistema-alvo Número de sistemas-alvo

Fonte: (SOMMERVILE, 2011, P. 63)

Por vezes, a linha que separa requisitos funcionais de requisitos não-funcionais acaba por ser mais tênue que o esperado. Isso acontece pelo fato de que os requisitos estabelecem, no sistema como um todo, relações de interdependência, podendo conflitar ou interagir entre si. Por isso, segundo Sommervile (2011), é importante que, no documento de requisitos, os RNF sejam, de alguma forma, distinguidos dos RF, podendo ser colocados em uma seção a parte ou explicitamente destacados.

4 ETAPAS DA ENGENHARIA DE REQUISITOS

4.1 Elicitação de requisitos

Quando falamos de elicitar requisitos, nos referimos a uma "atividade com o propósito de identificar os requisitos de um sistema através de conversas, análises e consultas a documentos e pesquisas de mercado" (SOMMERVILE, 2011).

De certa forma, esse processo envolve tanto os funcionários de uma empresa, como também o usuário final, porque o sistema que será implementado pode afetá-los. Baseado nisso, para levantarmos os requisitos de um sistema, necessitamos antes de tudo analisar se esse sistema é útil para a empresa, através de um estudo inicial de viabilidade.

Logo após isso, os engenheiros de software conversarão com os clientes ou usuários finais para a obtenção de informações sobre os serviços que o sistema deve ter, restrições etc. Esses clientes em sua maioria são as pessoas que possuem influência direta ou indireta sobre aquele sistema, e podem ser chamados de *stakeholders*.

Há outras fontes de requisitos como o domínio da aplicação, ou outros sistemas. Garantindo até diferentes pontos de vista, e "mostrando possíveis requisitos em comum que podem estruturar a descoberta e a documentação dos requisitos de um sistema. "(SOMMERVILE, 2011).

Durante esse processo, o engenheiro de software Ian Sommervile cita algumas atividades que são fundamentais neste processo para que não haja futuros problemas, sendo elas:

• A descoberta dos requisitos (elicitação), onde há a maior interação entre os engenheiros e os *stakeholders* para justamente descobrir quais são os requisitos, seus níveis (requisitos de usuário ou de sistema) etc. Algumas das técnicas utilizadas são: as entrevistas; brainstorming e a prototipagem.

• A classificação e a organização desses requisitos, onde eles devem ser agrupados com outros requisitos relacionados para facilitar a identificação de possíveis erros.

Algo importante durante a aplicação dessas atividades, é evitar desvio do escopo do projeto. "Ao interagir com um *stakeholder* para levantar suas necessidades, ele provavelmente trará à tona todos os problemas que vivencia" (VASQUEZ; SIMÕES, 2016). Nem sempre um único projeto resolverá todos os problemas dele. Portanto, é importante sempre antes de tomar uma atitude, verificar se esse requisito compreende alguma necessidade do projeto.

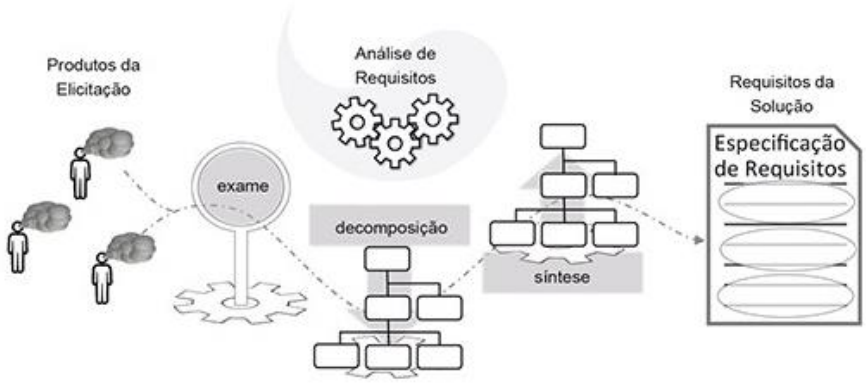
Devemos ter em mente também que nem sempre os *stakeholders* citarão todos os requisitos que necessitam, sendo assim chamados de requisitos implícitos. “Não se deve esperar que as partes interessadas digam tudo que é necessário voluntariamente.” (VASQUEZ; SIMÕES, 2016). Um bom analista já consegue perceber os requisitos necessários para tal sistema, antes de serem levantados pelos *stakeholders*. Já que “Conseguir identificar o que a parte interessada pensa que é óbvio e não precisa ser dito, ou antecipar um desejo que ela ainda não percebeu, são ações que evitam muito retrabalho à frente no projeto.” (VASQUEZ; SIMÕES, 2016).

4.2 Análise de Requisitos

Após obter esses requisitos, um processo de análise detalhada deve-se iniciar. Isso porque algumas inconsistências, omissões ou redundâncias podem acontecer. “Deve-se perceber quais são os requisitos necessários e que o usuário deseja.” (SOMMERVILE, 2011).

“Se a elicitação de requisitos descobre as peças do quebra-cabeças, então a análise de requisitos procura montá-lo. O objetivo da análise de requisitos é aumentar o entendimento atual da informação, completá-la e melhorá-la.” (VASQUEZ; SIMÕES, 2016). A seguir vemos na Figura 2 um resumo das habilidades básicas nessa etapa que é análise de requisitos.

Figura 2 - As habilidades básicas na análise de requisitos



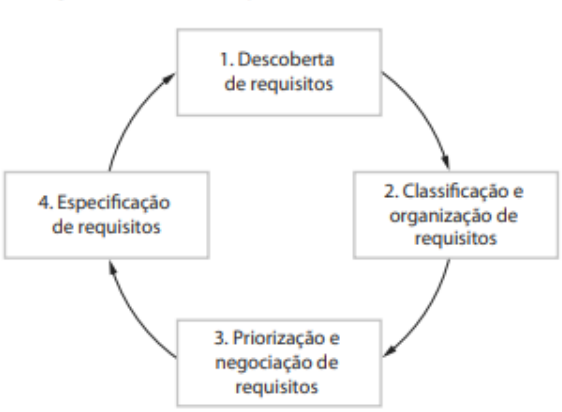
Fonte: Vazquez e Simões (2016)

Durante a análise dos requisitos obtidos anteriormente, podemos perceber que podem surgir conflito de requisitos entre usuários/clientes, e “Cabe ao engenheiro de software resolver esses conflitos sem comprometer a satisfação dos objetivos de cada um.” (SOMMERVILE, 2011). Os usuários e clientes devem discutir, priorizar e negociar esses requisitos até que

cheguem a um consenso sobre alterações e simplificações necessárias. Por fim, os requisitos mais críticos são analisados para que sejam atendidos prioritariamente.

Esses processos na engenharia de software podem comprometer o sistema a longo ou curto prazo se não forem bem executados. São etapas em que as coisas devem ser explícitas e realizadas no tempo em que for necessário, garantindo que nenhum tópico ou requisito fique de fora. No fim das contas, o que ocorre é que nem mesmo os *stakeholders* sabem o que querem, ou os requisitos podem sofrer mudanças por conta de saída ou entrada de novos stakeholders, ou a evolução da tecnologia num geral. Tornando a Elicitação e Análise de Requisitos um processo iterativo e constante, assim como podemos ver na Figura 3.

Figura 3 - O processo de elicitação e análise de requisitos



Fonte: Sommerville (2011)

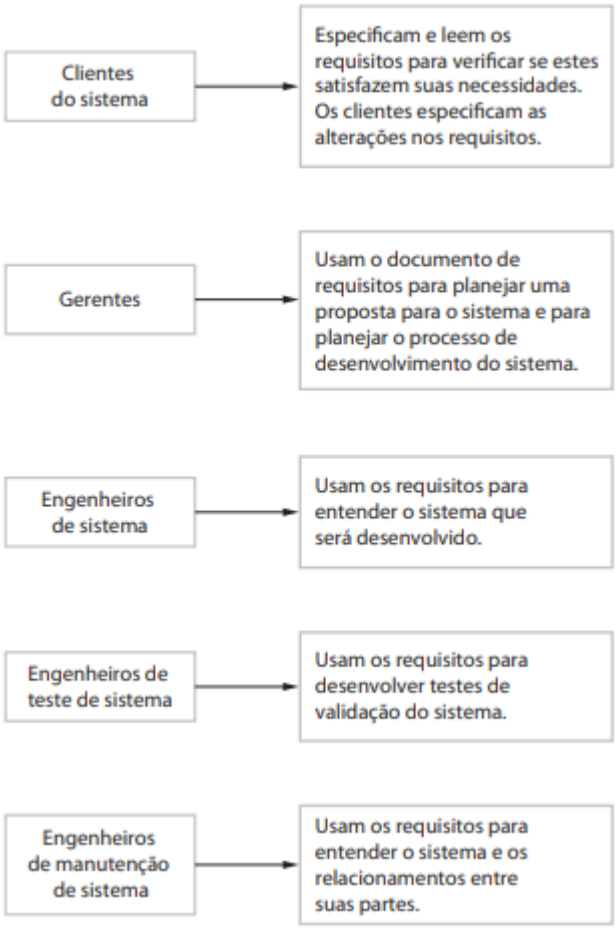
4.3 Documentação

A documentação de requisitos é a declaração oficial que os desenvolvedores devem implementar. Nela é incluída: requisitos de usuário e especificação detalhada dos requisitos do sistema, seja em uma única descrição ou apresentados em documentos separados (SOMMERVILLE, 2011).

Apesar da grande importância inicial da documentação, os requisitos mudam tão rapidamente que já se tornam ultrapassados assim que termina de ser escrito. Portanto abordagens como *Extreme Programming* coletam requisitos de usuário de forma incremental, dessa maneira o usuário prioriza os requisitos para a implementação do próximo sistema (SOMMERVILLE, 2011).

O documento de requisitos tem um conjunto diversificado de usuários. Sommerville classifica-os de acordo com o conteúdo da Figura 4.

Figura 4 - Usuários de um documento de engenharia de requisitos



Fonte: Sommerville (2011)

Naturalmente, a informação incluída em um documento de requisitos depende do tipo de software a ser desenvolvido e da abordagem de desenvolvimento que está em uso. Se uma abordagem evolutiva é adotada para um produto de software (por exemplo), o documento de requisitos deixará de fora muitos dos capítulos detalhados sugeridos. O foco será sobre a definição de requisitos de usuário e os requisitos não funcionais de alto nível de sistema. Nesse caso, os projetistas e programadores usam seu julgamento para decidir como atender aos requisitos gerais de usuário para o sistema. A Tabela 2 a seguir, trata exatamente sobre os elementos da estrutura de um documento de requisitos.

Tabela 2 - A estrutura de um documento de requisitos

Capítulo	Descrição
Prefácio	Deve definir os possíveis leitores do documento e descrever seu histórico de versões, incluindo uma justificativa para a criação de uma nova versão e um resumo das mudanças feitas em cada versão.
Introdução	Deve descrever a necessidade para o sistema. Deve descrever brevemente as funções do sistema e explicar como ele vai funcionar com outros sistemas. Também deve descrever como o sistema atende aos objetivos globais de negócio ou estratégicos da organização que encomendou o software.
Glossário	Deve definir os termos técnicos usados no documento. Você não deve fazer suposições sobre a experiência ou o conhecimento do leitor.
Definição de requisitos de usuário	Deve descrever os serviços fornecidos ao usuário. Os requisitos não funcionais de sistema também devem ser descritos nessa seção. Essa descrição pode usar a linguagem natural, diagramas ou outras notações compreensíveis para os clientes. Normas de produto e processos que devem ser seguidos devem ser especificados.
Arquitetura do sistema	Deve apresentar uma visão geral em alto nível da arquitetura do sistema previsto, mostrando a distribuição de funções entre os módulos do sistema. Componentes de arquitetura que são reusados devem ser destacados.
Especificação de requisitos do sistema	Deve descrever em detalhes os requisitos funcionais e não funcionais. Se necessário, também podem ser adicionados mais detalhes aos requisitos não funcionais. Interfaces com outros sistemas podem ser definidas.
Modelos do sistema	Pode incluir modelos gráficos do sistema que mostram os relacionamentos entre os componentes do sistema, o sistema e seu ambiente. Exemplos de possíveis modelos são modelos de objetos, modelos de fluxo de dados ou modelos semânticos de dados.
Evolução do sistema	Deve descrever os pressupostos fundamentais em que o sistema se baseia, bem como quaisquer mudanças previstas, em decorrência da evolução de hardware, de mudanças nas necessidades do usuário etc. Essa seção é útil para projetistas de sistema, pois pode ajudá-los a evitar decisões capazes de restringir possíveis mudanças futuras no sistema.
Apêndices	Deve fornecer informações detalhadas e específicas relacionadas à aplicação em desenvolvimento, além de descrições de hardware e banco de dados, por exemplo. Os requisitos de hardware definem as configurações mínimas ideais para o sistema. Requisitos de banco de dados definem a organização lógica dos dados usados pelo sistema e os relacionamentos entre esses dados.
Índice	Vários índices podem ser incluídos no documento. Pode haver, além de um índice alfabético normal, um índice de diagramas, de funções, entre outros pertinentes.

Fonte: Sommerville (2011)

4.3.1 Especificação

É o processo de registrar os requisitos de usuário e de sistema em um documento de requisitos. Via de regra, os requisitos de usuário e de sistema descrevem requisitos funcionais e não-funcionais de forma clara, inequívoca, de fácil compreensão até para usuários que não possuam conhecimento técnico, completa e consistente. Todavia, dificilmente alcançamos tal resultado, dada a interpretação de cada *stakeholder* (SOMMERVILLE, 2011).

Os requisitos de sistema são uma extensão dos requisitos de usuário, usado como ponto de partida, especificando como os requisitos de usuário devem ser atendidos. Deve descrever apenas o comportamento externo do sistema e suas restrições operacionais (SOMMERVILLE, 2011). Para evitar ambiguidade e mal-entendidos, são seguidas certas diretrizes:

• Inventar formato-padrão e garantir que todos os requisitos sigam esse formato

• Usar linguagem consistente, distinguindo requisitos obrigatórios e desejáveis. Obrigatórios utilizando-se “deve”, desejáveis utilizando-se “pode”

• Destacar partes fundamentais (negrito, itálico ou cores)

• Não usar jargões, siglas e acrônimos. Evitar palavras como “arquitetura” e “módulo”

• Associar uma lógica a cada um dos requisitos de usuário se possível

4.4 Validação

Nessa etapa se verifica se os requisitos realmente atendem o desejo do cliente. Diferente da análise de requisitos, esse processo tem ênfase em encontrar problemas nos requisitos. A validação certifica que não há erros nos requisitos, visto que, ao consertar um requisito, a implementação inteira do sistema deve ser alterada, gerando altos custos de retrabalho desnecessários que poderiam ter sido ajustados antes do sistema já estar em operação. Dentre as verificações estão:

• Verificações de validade: identificar funções necessárias, adicionais ou diferentes

• Verificações de consistência: identificar restrições contraditórias ou descrições com mesma função

• Verificações de completude: identificar todas as definições de funções e restrições

• Verificações de realismo: identificar impossibilidade de funções e restrições (como orçamento e cronograma)

• Verificabilidade: escrever conjunto de testes demonstrando todos os requisitos atendidos ao cliente

4.4.1 Técnicas usadas para validação de requisitos

• Revisão de requisitos: os requisitos são analisados sistematicamente por uma equipe de revisores que verifica erros e inconsistências

• Prototipação: é analisado e testado um modelo executável do sistema pelos usuários finais e clientes

• Geração de casos teste: cada requisito passa por um teste para se verificar se ele será de fácil implementação ou se deverá ser reconsiderado

Esse processo é mais passivo à erros, dada sua complexidade de análise abstrata. Raramente é encontrado todos os erros, consequente é inevitável a necessidade de mudanças nos requisitos mesmo após se ajustar o documento de requisitos.

4.4.2 Estudo de viabilidade

Essa etapa ocorre após a especificação de requisitos de negócio e visa avaliar se o projeto é viável dos pontos de vista operacional, técnico, econômico e organizacional. A seguir, temos os tipos de viabilidade:

• Viabilidade organizacional: está relacionada a quanto a solução beneficia a organização.

• Verifica-se se haverá aderência ao uso da solução por parte dos usuários devido a cultura organizacional e a percepção dos envolvidos; se a solução está alinhada com os objetivos estratégicos da organização; se há compreensão e suporte da alta direção da organização em relação ao projeto, etc.

• Viabilidade operacional: está relacionada a quanto a solução se adéqua a organização; quais são os requisitos da solução; o que o cliente espera que o sistema faça;

• Viabilidade econômica: análise entre o custo de desenvolvimento e os benefícios após implementação do projeto (custo-benefício).

• Viabilidade técnica: atrelada ao suporte técnico que a organização oferecerá para o desenvolvimento do projeto; restrições da equipe ou da tecnologia; necessidade de se investir em pesquisas antes de realizar o projeto; etc.

• Viabilidade de cronograma: cruzamento entre as atividades levantadas e o tempo estimado para realizá-las; definição de marcos do projeto; impacto de atrasos.

• Outros: viabilidade legal, cultural, marketing, etc.

O próximo passo também se baseia em um documento. O documento de estudo de viabilidade deve conter:

• Introdução

• Finalidade: citar a viabilidade e sua função, além dos usuários

• Definições: explicar possíveis acrônimos e abreviações, caso não possua explicitar que não se aplica

• Objetivo: descrever de forma clara o objetivo do projeto

• Escopo: delimitar aspectos que serão encobertos, delimitando fronteira de atuação

• Diagnóstico: descrever situação atual para cliente, breve descrição das dificuldades, versões e desenvolvedores

• Requisitos: O documento de requisitos deve ser inserido nessa etapa

• Alternativas propostas: Listar alternativas propostas para os problemas encontrados no passo anterior

• Alternativas recomendadas: citar alternativas mais viáveis para solução

• Benefícios: citar benefícios tangíveis ou intangíveis para implementar ao sistema

• Custos: relacionar custos com maior precisão possível, junto da fonte desses custos

• Riscos: listar possíveis riscos da alternativa sugerida, além de ações preventivas

• Cronograma: apresentação das atividades com datas e recursos envolvidos

• Conclusão: evidenciar se o projeto é ou não viável, junto das razões que induziram esse resultado

4.5 Gerência de requisitos

Durante a criação de um projeto, diversas mudanças ocorrem na fase de desenvolvimento, teste e execução, com isso é natural o surgimento de novos requisitos e a alteração na nos requisitos já pré-existentes. Esse processo de constante mudança também ocorre com nós seres humanos, recebendo o nome de Evolução, o que não difere do termo utilizado para essa mudança nos sistemas, que tem como nome Evolução dos Sistemas. Tal evolução que se refere ao TIC – Tecnologia da Evolução e Comunicação – ocorreu de maneira rápida, devido à grande demanda de informações que recebemos e que acrescentamos em nossos projetos (NASCIMENTO, 2020).

Para a gerência de requisitos, é possível dar como definição algo que “[...] tem por objetivo gerenciar os requisitos dos produtos do projeto e dos seus componentes e garantir o alinhamento entre os requisitos, os planos do projeto e os produtos de trabalho do projeto.” (VAZQUEZ; SIMÕES, 2016, p.341 apud CMMI-DEV). Tendo essa análise pelo fato de que

os requisitos de um software são incompletos e uma vez que executado o programa, novos requisitos serão necessitados pelos *stakeholders* e novos usuários tendem a possuir diferentes necessidades e prioridades (SOMMERVILLE, 2011).

Composto de diversas etapas, é difícil determinar todos os aspectos fundamentais na gerência de requisitos, mas de antemão é necessário que para implementação desses aspectos haja uma concordância entre um conjunto de objetivos e políticas a se seguir. Como primeiro passo, é necessário o entendimento de todos os requisitos tendo um acordo entre ambas as partes (equipe desenvolvedora e fornecedores dos requisitos) como menciona Vazquez e Simões (2016). Após isso, a equipe deve se comprometer com projeto que foi acordado e documentado, seguindo uma série de etapas que devem se manter sempre atualizadas e que garantirão um processo dinâmico e eficaz.

Relembrando que cada autor possui suas próprias prioridades e aspectos que consideram fundamentais dentro da gerência e com o objetivo pré-estabelecido, os aspectos fundamentais que serão trabalhados a seguir com suas próprias características e ramificações, dentro da gerência de requisitos, são: controle de mudanças; gerência de configuração, priorização de requisitos; rastreabilidade e gestão de qualidade. Vale ressaltar que mesmo com todas as divergências dentro dos aspectos fundamentais, todos autores entornam seus conceitos envolta das práticas de manter, priorizar e rastrear seus requisitos, além de avaliar e aprovar mudanças (LAGUNA, 2016).

4.5.1 Responsável pela gerência de requisitos

Os cargos na gerência de requisitos podem se dividir em duas áreas, uma na parte de criação do plano de requisitos e outra na execução do plano. Segundo Vazquez e Simões (2016) a parte de elaboração fica por conta do gerente de projetos ou pela pessoa que está responsável pela metodologia de desenvolvimento de software da empresa, enquanto a execução é feita sem uma determinação muito rígida, geralmente realizada pela equipe de gerenciamento de projetos (com o analista de requisitos exercendo uma função de suporte a quase todas as etapas e áreas).

4.5.2 Controle de mudanças

O controle de mudanças, também chamado de gestão de mudanças, não se trata de um processo que irá realizar uma prevenção contra as mudanças de um projeto, mas sim a

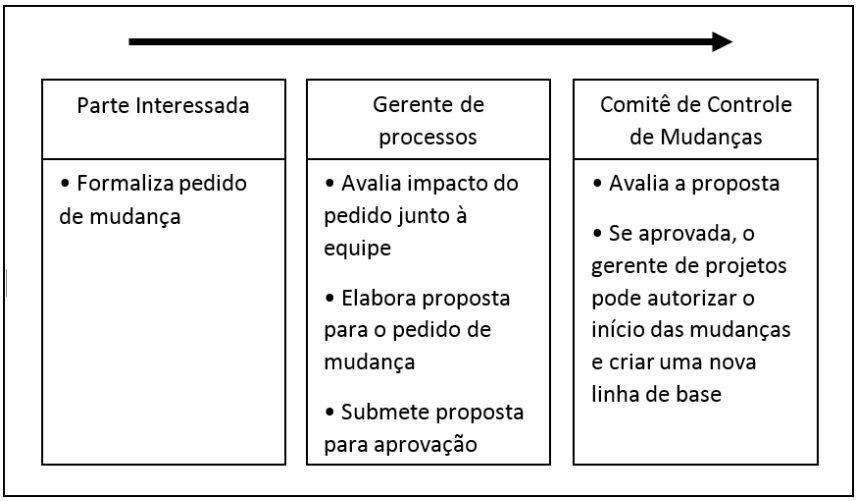
identificação, avaliação e gerenciamento de possíveis mudanças que irão surgir, podendo eventualmente serem aprovadas ou rejeitadas, fazendo-se uma análise de possíveis impactos sejam eles no custo, cronograma do projeto, qualidade, dentre outros (ENGHOLM, 2013).

Dentre os principais motivos para mudanças, temos: real complexidade do sistema de acordo com o desenvolvimento; requisitos mal desenvolvidos que terão que ser corrigidos e adaptados; adição de novas funcionalidades ao software; além de mudanças na tecnologia ou nas preferências dos clientes (SAYÃO; BREITMAN, 2007).

Todas as mudanças geram impactos, segundo Engholm (2013) essas mudanças impactam principalmente nos custos, prazos, quantidade de riscos, satisfação do cliente e alteração de outros requisitos, tornando a área extremamente importante no projeto como um todo, fazendo-a ser executada de maneira cuidadosa para que não haja impactos negativos.

Tais mudanças são solicitadas por diversas partes, para isso, não basta um acordo verbal com os responsáveis pelo projeto, é necessário que haja uma parte escrita e documentada que será avaliada e incrementada ao sistema que está sendo desenvolvido. Devido a esta etapa, existe uma figura criada que fica responsável pela avaliação, onde em projetos pequenos é composta por apenas uma pessoa e em projetos grandes por um grupo, recebe o nome de Comitê de Controle de Mudança (CCB, do inglês Change Control Board) (VAZQUEZ; SIMÕES, 2016, p.346 apud PMBOK Guide). A Figura 5 apresenta graficamente a situação do processo até chegar ao Comitê de Controle de Mudanças.

Figura 5 - Esquema até o Comitê de Controle de Mudanças



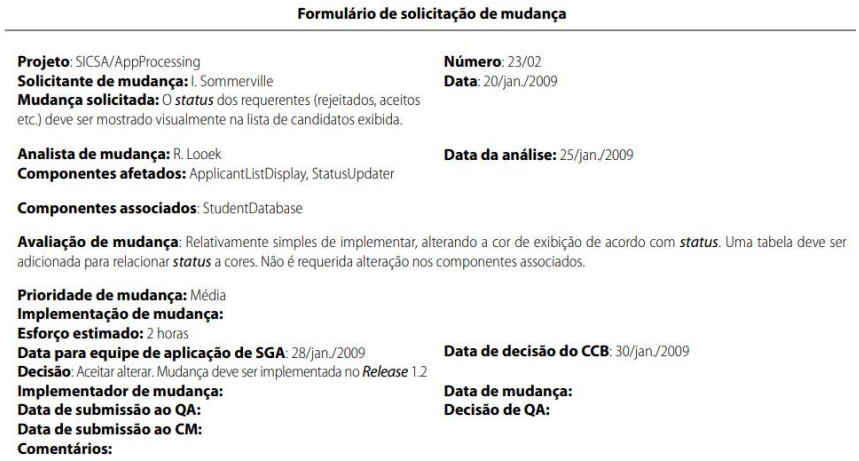
Fonte: autoria própria adaptada de (VAZQUEZ; SIMÕES, 2016, p.346)

Como descrito anteriormente, é necessário que haja uma solicitação de mudança por escrito (CR, do inglês *change request*), normalmente feito através de um preenchimento de

Formulário de Requisição de Mudança (CRF, do inglês *change request form*) e o envia para parte responsável pelo tratamento das mudanças como na Figura 5. Esse processo não pode ser visto com um obstáculo, mas sim um filtro que vai ajudar no controle e futuramente na rastreabilidade desses requisitos incrementados, facilitando a manipulação e armazenamento dos mesmos em um formato único e compartilhado (SAYÃO; BREITMAN, 2007).

A Figura 6 mostra um *template* básico de formulário a ser preenchido, que devem ter no mínimo "informações relativas ao tipo de mudanças, à pessoa responsável pela solicitação, data e órgão de origem, e uma boa descrição da mudança em si" (SAYÃO; BREITMAN, 2007, p.7).

Figura 6 - Formulário de Requisição de Mudança



Formulário de solicitação de mudança

Projeto: SICSA/AppProcessing
Número: 23/02
Solicitante de mudança: I. Sommerville
Data: 20/jan./2009
Mudança solicitada: O *status* dos requerentes (rejeitados, aceitos etc.) deve ser mostrado visualmente na lista de candidatos exibida.

Analista de mudança: R. Looek
Data da análise: 25/jan./2009
Componentes afetados: ApplicantListDisplay, StatusUpdater

Componentes associados: StudentDatabase

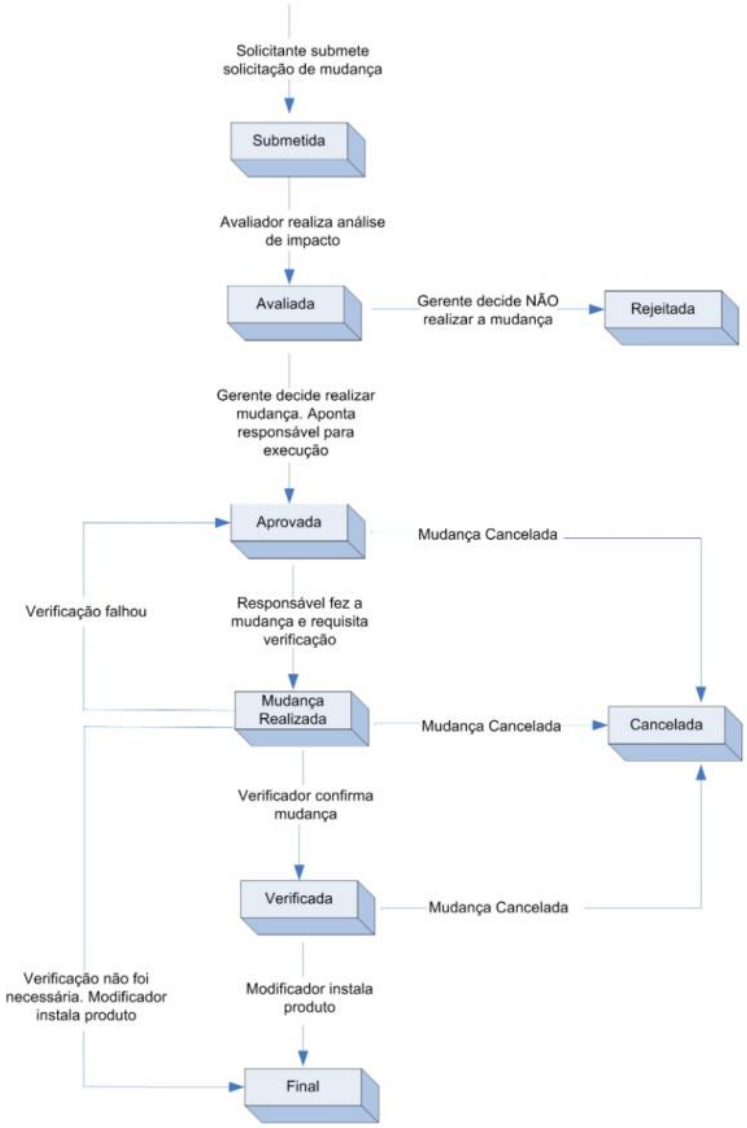
Avaliação de mudança: Relativamente simples de implementar, alterando a cor de exibição de acordo com ***status***. Uma tabela deve ser adicionada para relacionar ***status*** a cores. Não é requerida alteração nos componentes associados.

Prioridade de mudança: Média
Implementação de mudança:
Esforço estimado: 2 horas
Data para equipe de aplicação de SGA: 28/jan./2009
Data de decisão do CCB: 30/jan./2009
Decisão: Aceitar alterar. Mudança deve ser implementada no ***Release*** 1.2
Implementador de mudança:
Data de mudança:
Data de submissão ao QA:
Decisão de QA:
Data de submissão ao CM:
Comentários:

Fonte: Sommerville (2011)

O processo de mudança de requisitos deve ser bem elaborado e comunicativo, em todas suas etapas deve haver uma comunicação entre o responsável e sua equipe, além dos outros funcionários envolvidos no projeto. Vale ressaltar que não somente durante a mudança se há essa comunicação, após a finalização do projeto é essencial uma conversa entre todos envolvidos, novas funcionalidades trazem alterações para todos, inclusive de adaptabilidade ao que foi incrementado. Temos como exemplo a aprovação e execução do projeto "Pix", criado pelo Banco Central do Brasil, que foi amplamente divulgado e comunicado sua finalização e uso através das mídias, não só para a equipe interna, mas para todos os usuários. A Figura 7 representa um esquema de processo de requisição de mudança em requisito que foi proposto por Wiegers.

Figura 7 - Processo de requisição de mudança em requisito



Fonte: Sayão e Breitman (2007) apud Wiegers

4.5.3 Gerência de configuração

Gerência de configuração de Software, "(GCS) é um conjunto de atividades de apoio que permite a absorção ordenada das mudanças inerentes ao desenvolvimento de software, mantendo a integridade e a estabilidade durante a evolução do projeto." (DIAS, 2016). Dentro de sua grande abrangência ela engloba algumas áreas, a de Controle de Mudança já foi citada, mas as que serão desenvolvidas a seguir são a de Controle Versões e Controle de Configuração de Software.

4.5.3.1 Controle de versões

O Controle de Versões, ou gerenciamento de versões (VM, do inglês *version management*) é um processo que permite que todos os envolvidos no projeto consigam acessar o histórico de versões anteriores, fazendo recuperações, estudos sobre as mudanças, observar as informações e novidades incrementadas ao software, rever o levantamento inicial de requisitos, dentre outros (REVELO, 2019).

Os artefatos são mantidos em um repositório, o que é essencial para que todos os desenvolvedores trabalhem na mesma versão, sem que haja discrepância entre a própria equipe, onde poderia haver novos pedidos de mudanças de requisitos sobre algo já trabalhado ou até mesmo disponibilizar uma versão para uso incompleta, o que acarretaria em um longo processo de mudança.

Além disso, a produtividade é aumentada consideravelmente, onde um mesmo trabalho pode ser dividido entre equipes para que o prazo seja atendido o mais rápido possível, principalmente pelas altas pressões feitas pelos clientes, por isso pode ocorrer dois modelos de controle de versão: Centralizado (SCVc) e Distribuído (SCVd). O modelo centralizado constitui-se de um único repositório central, onde cada integrante trabalha com uma cópia do arquivo (REVELO, 2019). Já no modelo distribuído, cada desenvolvedor possui seu próprio repositório, tendo a comunicação entre repositórios feita através de revisões por outra pessoa (REVELO, 2019).

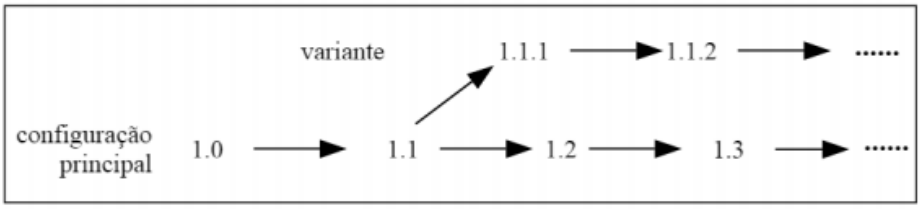
Para realizar esse controle, normalmente há disponível diversas ferramentas de versionamento de software que vão além de arquivos espalhados em uma pasta, dentre os mais usados é possível citar: Subversion, Git, Mercurial, RedMine, Microsoft Visual SourceSafe, dentre outros.

4.5.3.2 Controle de configuração de software

O Controle de Configuração de Software faz menção ao processo de "manter, sob estrito acompanhamento, o conjunto de artefatos relacionadas a uma determinada configuração do sistema" (SAYÃO; BREITMAN, 2007, p.10). Estando ciente que a configuração "é o estado do conjunto de itens que formam o sistema em um determinado momento" (REVELO, 2019) um bom exemplo se trata das funções betas disponibilizadas de alguns aplicativos, onde há uma linha principal que será lançada com mais funcionalidades e a beta que serve apenas de teste, nessas versões limitadas, o próprio fabricante realiza restrições nas funcionalidades e conjuntos de requisitos (SAYÃO; BREITMAN, 2007).

Essa restrição é exatamente o papel do controle de Configuração de software, manipulando e controlando todas as mudanças que estão sendo realizadas em um software. A Figura 8 retrata um processo simples de versões sendo trabalhadas e alteradas através de revisões, lembrando que todas carregam uma porção dos requisitos iniciais, tendo o fato de que saíram de um mesmo princípio.

Figura 8 - Revisões e Variantes



Fonte: Sayão e Breitman (2007)

4.5.4 Priorização de requisitos

Até esse ponto é possível perceber a importância que os requisitos possuem no desenvolvimento de um software, no entanto é imprescindível o processo de priorização de requisitos, isso se dá pela limitação de recursos disponíveis a um projeto, sejam esses recursos financeiros, quantidade de funcionários, data para entrega do projeto, dentre outros.

A limitação de recursos é a barreira principal de que se faz necessária a priorização, por isso é necessário decidir o que deverá ser feito antes e com mais urgência para que o software seja funcional. Tentar priorizar todos os requisitos acaba em desperdício total ou parcial do projeto, onde na tentativa de se realizar tudo, acaba-se um dos recursos e o que está pronto não tem objetividade alguma, caso os recursos fossem ilimitados não haveria necessidade de priorização (VENTURA, 2015).

Segundo Vazquez e Simões (2016) “Priorizar significa atribuir um valor de importância relativa entre os requisitos. O objetivo é maximizar o valor entregue pelo projeto, fazendo com que as coisas mais importantes sejam tratadas em primeiro lugar”.

De acordo com Duan et al. (2009) é necessário atenção e cuidado na seleção de requisitos a serem desenvolvidos, a observação de quais deles terão maior impacto para a satisfação de usuários e clientes.

4.5.4.1 Quem realiza a função?

A função está designada ao gerente de projeto, mas o analista de requisitos, como em outras, também está presente nessa equipe. É claro que a decisão de o que deve ser mais importante tem que vir do cliente, pois ele está solicitando o software, mas cabe aos envolvidos na tarefa de priorização, moldar a ideia e intenção do cliente, explicando ou informando as consequências de uma escolha (VAZQUEZ; Simões, 2016).

Suponhamos que o cliente prefira e deseje que o requisito A seja priorizado, porém existe uma implicação onde para que A seja executado, o requisito B precisa estar pronto, logo o requisito B será priorizado no projeto, assim conseguindo realizar a função A que o cliente deseja e que terá uma priorização necessariamente inferior.

4.5.4.2 Métodos

Existem diversos métodos de priorização de requisitos, de acordo com Lehtola e Kauppinen (2004) eles se dividem em duas categorias, os que são baseados em valores atribuídos aos requisitos e os de negociação.

Os métodos baseados em valores atribuídos aos requisitos, permitem a priorização de maneira absoluta ou relativa. Quando absoluta, os valores irão ser atribuídos a cada requisito, porém sem levar em conta os restantes. Já quando é relativa, a atribuição é feita aos requisitos havendo uma comparação aos demais (CORDEIRO, 2010, apud LETHOLA; KAUPPINEN).

Os métodos de negociação “baseiam-se na ideia de que a prioridade de um requisito pode ser determinada através do consenso entre os diferentes *stakeholders*” (CORDEIRO, 2010, apud LETHOLA; KAUPPINEN).

Existem diversos métodos de priorização de sistemas, a Tabela 3 mostra resumidamente alguns dos mais usados.

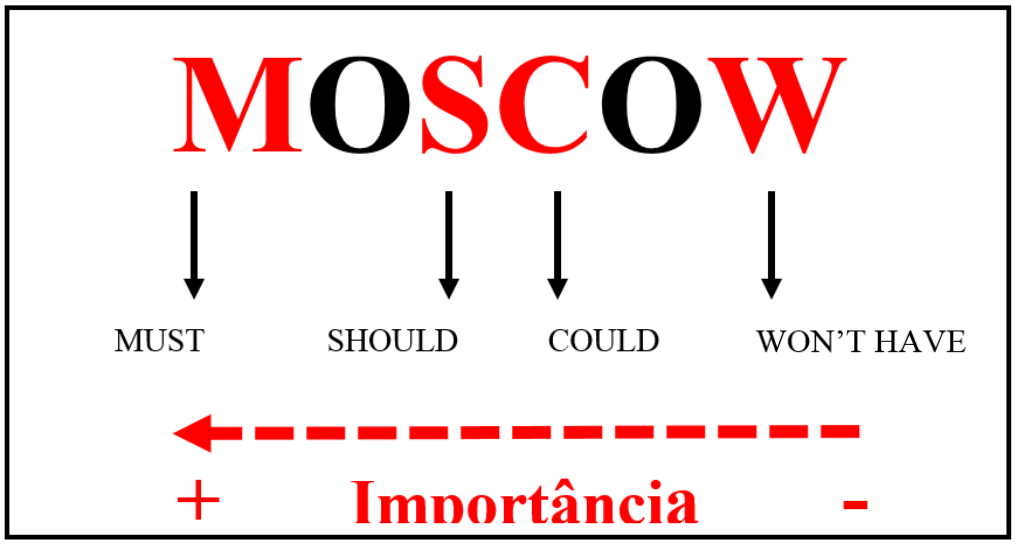
Tabela 3 - Métodos de Priorização

MÉTODO	DESCRICAO	TIPO DE PRIORIZAÇÃO
Árvore de busca binária	Algoritmo utilizado para a busca de informações. Consiste em selecionar um dos requisitos de uma lista e colocá-lo no nó raiz, a partir daí cada requisito restante na lista deve ser comparado com cada nó da árvore em relação a importância. Ao final obtém-se a árvore de requisitos priorizados.	Relativa
Atribuição numérica	Utiliza uma escala de 1 a 5, os *stakeholders* devem identificar a qual nível da escala cada requisito corresponde.	Absoluta
Método dos 100 pontos	A cada *stakeholder* são fornecidos 100 pontos que devem ser utilizados em favor dos requisitos mais importantes com base na percepção de cada um.	Relativa
Triagem de requisitos	Consiste em um processo de várias etapas que inclui estabelecer prioridades relativas para os requisitos, estimar recursos necessários para cada requisito e selecionar um subconjunto de requisitos para otimizar a probabilidade de sucesso do projeto.	Relativa
AHP (Analytic Hierarchy Process - Método de Análise Hierárquica)	Método de apoio à decisão utilizado em situações nas quais múltiplos objetivos estão presentes. O método utiliza a comparação par a par para calcular o valor relativo de cada item, neste caso requisito.	Relativa

Fonte: Cordeiro (2010) adaptado de Allen et al. (2008)

Todos os métodos tem seus prós e contras, apesar disso, cada um tem uma maior usabilidade de acordo com o projeto. O método MOSCOW, que apesar de não estar presente na tabela, é um dos mais utilizados e possui uma linguagem simples e funcional. A Figura 9 destaca a significação da sigla que será trabalhada a seguir.

Figura 9 - MOSCOW



Fonte: autoria própria

Neste método, atribuímos o requisito a um dos 4 valores possíveis (*MUST, SHOULD, COULD, WOULD*) e cada um possui um significado diferente.

MUST HAVE (deve ter), refere-se aos requisitos essenciais e que são obrigatórios de se ter no software. Caso esse requisito não seja contemplado, o software não estará cumprindo seu papel, pois não terá valor para o cliente (VAZQUEZ; SIMÕES, 2016).

SHOULD HAVE (deveria ter), engloba os requisitos que são importantes, mas não essenciais de um ponto de vista estratégico. Mesmo que não essencial, ele possui um alto grau de prioridade, podendo afetar a satisfação do cliente (em um grau muito alto), porém ele pode ser esperado um pouco mais para ser trabalhado e o projeto continuará viável, mesmo que sem todas suas funcionalidades (COSTA, 2018)

COULD HAVE (poderia ter), requisitos que são desejáveis, mas não necessários, podem melhorar a expectativa e satisfação dos usuários e clientes com algum esforço, mas são executados somente quando sobra algum tempo e recurso financeiro do projeto, sendo os mais fáceis de serem cortados do escopo inicial (VAZQUEZ; SIMÕES, 2016).

WON'T HAVE THIS TIME (não terá por agora), são os itens menos críticos e que estão de comum acordo de que não serão executados, mesmo que futuramente eles possam ser considerados e usados em outros projetos ou em outra versão, pois futuramente sua importância pode ser revista (VAZQUEZ; SIMÕES, 2016).

4.5.4.3 Critérios de priorização

Toda prioridade pode mudar de acordo com o tempo, por isso é sempre preciso fazer uma revisão e nova priorização (repriorização). A Tabela 4 mostra alguns critérios comuns para priorização.

Tabela 4 - Critérios de Priorização

Critérios de priorização	Como priorizar os requisitos?	Quando utilizar?
Valor de negócio (benefício)	Com base na análise de custo-benefício.	Projetos incrementais ou que possuem limitações orçamentárias.
Risco (técnico ou de negócio)	Com base nos maiores riscos de falha para o projeto.	Maximizar a probabilidade de sucesso do projeto.
Custo	Com base nos custos de implementar os requisitos.	As partes interessadas podem mudar prioridades depois que entendem os custos.
Perdas	Com base nas perdas ocasionadas por não implementar o requisito.	Quando políticas e regulamentações são impostas à organização ou risco de perda de oportunidade para concorrência.
Dependência com outros requisitos	Com base na dependência que requisitos com alto valor agregado possuem de requisitos com baixo valor agregado.	Projetos incrementais ou que possuem limitações orçamentárias.
Estabilidade	Com base no consenso (mais úteis ou valiosos) obtido pelas partes interessadas.	Quando há conflitos ou indefinição de requisitos de partes interessadas.
Sensibilidade temporal	Com base na sensibilidade de tempo.	Quando há janelas de oportunidade para o negócio que devem ser aproveitadas (sazonalidade ou situações específicas – por exemplo, legais).

Fonte: Vazquez e Simões (2016)

4.5.5 Rastreabilidade

Sendo de extrema importância e considerada como a principal atividade da gerência de requisitos, a rastreabilidade está intrinsicamente ligada as necessidades de um bom software, tendo papel essencial em sua produção. Basicamente, a rastreabilidade é uma área que busca identificar e realizar vínculos (elos) entre requisitos, rastreando assim a dependência entre os requisitos, os artefatos derivados (modelos, documentos, código fonte, sequências de testes ou executáveis) e também o ponto de origem e idealização, quanto a implementação do ponto referencial (*baseline*) que está a ser analisado (DIAS, SOUZA e MACEDO, 2018)

4.5.5.1 Benefícios

A rastreabilidade possui diversas funcionalidades e benefícios que facilitam o trabalho dos gerentes e desenvolvedores, dentre elas podemos citar: Presumir variações e alterações no projeto, seja por tempo ou recurso financeiro; resolução de requisitos em conflitos; correção de defeitos; desvendar inconsistências e lacunas nos sistemas; ajuda na gestão de riscos, onde requisitos com muitas dependências se tornam perigosos por conta de sua essencialidade;

visualizar a alocação de requisitos a componentes de software e também possíveis *updates*; realizar a cobertura de testes; garantir uma contínua concordância entre os requisitos dos interessados no sistema; recuso de componentes; dentre outros (SAYÃO e LEITE, 2005).

4.5.5.2 Classificação para a rastreabilidade

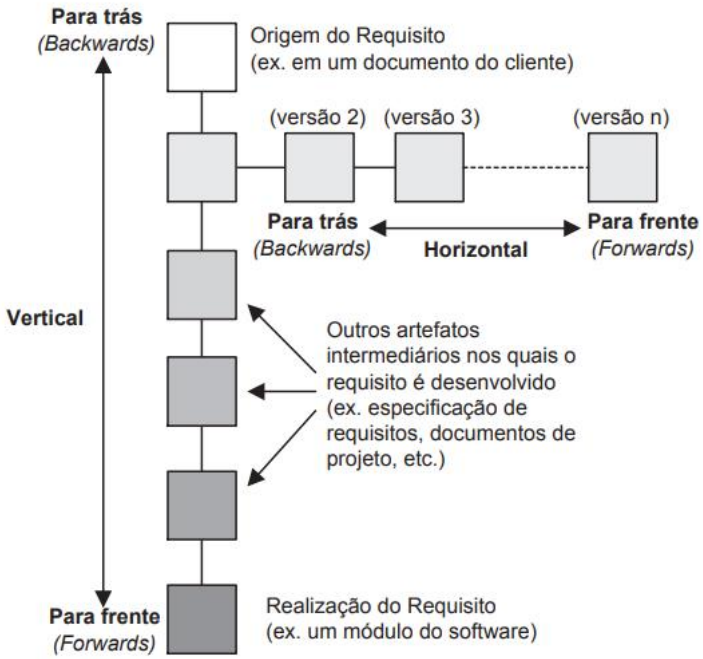
A capacidade de se rastrear um requisito até o produto final e suas etapas até chegar a esse ponto é definida como rastrear para frente (*forwards*), e a de rastrear um requisito refinado/ produto até sua origem é definida como rastrear para trás (*backwards*). Essas capacidades são conhecidas como rastreabilidade bidirecional, não sendo opcionais, mas sim essenciais para qualquer tipo de rastreabilidade, para que assim seja possível cumprir seus objetivos. Caso não tenha nenhuma das capacidades citadas, o processo de rastreabilidade será falho (DAVIS, 1993).

Dado isso, há 2 tipos de rastreabilidades: Rastreabilidade Horizontal e Vertical, e Pré e Pós Rastreabilidade.

4.5.5.3 Rastreabilidade horizontal e vertical

A rastreabilidade horizontal consiste em ajustar vínculos entre diferentes versões dos requisitos ou artefatos em certa fase do ciclo de vida do projeto (sendo também possível uma relação entre requisitos e artefatos de mesmo nível). Já a rastreabilidade vertical relaciona os requisitos ou artefatos produzidos nas distintas fases ao longo do ciclo de vida do projeto (VAZQUEZ; SIMÕES, 2016). A Figura 10, mostra de maneira simples e eficaz esse tipo de rastreabilidade.

Figura 10 - Rastreabilidade Horizontal e Vertical

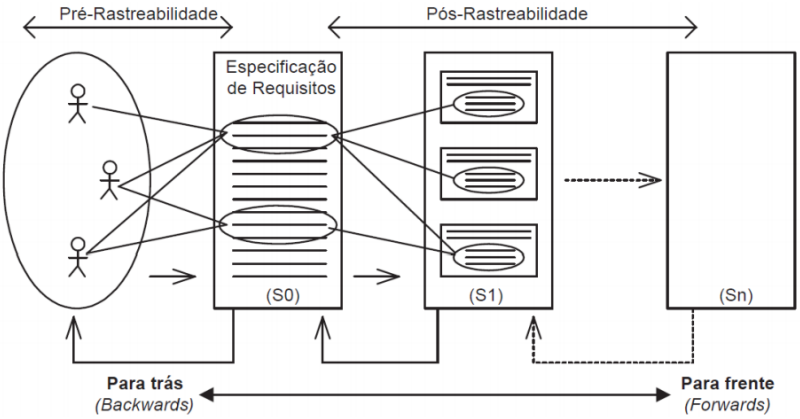


Fonte: Genvigir (2009) adaptado de Gotel (1995)

4.5.5.4 Pré e pós-rastreabilidade

A pré e pós-rastreabilidade estão relacionados ao ciclo de vida dos requisitos tendo como referência a especificação de requisitos. Quando retratado a pré-rastreabilidade, ela está indexada ao ciclo de vida no período que está antes da inclusão de requisitos na especificação de requisitos, conseguindo assim identificar a origem de um requisito e das fontes de requisitos (padrões organizacionais, clientes, usuários, normas, por exemplo) e artefatos gerados em outras fases do início do programa. Já a pós-rastreabilidade está concentrada no ciclo de vida dos requisitos após a especificação deles, permitindo identificar quais componentes do software implementam um determinado requisito e também saber quais requisitos estão associados a um componente do software a partir de um ponto referencial (*baseline*) (VAZQUEZ; SIMÕES, 2016). A Figura 11 possui um esquema gráfico que exemplifica a definição.

Figura 11 - Pré e Pós Rastreabilidade



Fonte: Genvigir (2009) adaptado de Gotel (1995)

4.5.5.5 Metamodelos e tipos de elos

"De acordo com Genvigir, vários autores fazem o uso de modelos com base nas informações de um determinado domínio que desejam representar de forma gráfica ou textual." (DIAS, SOUZA e MACEDO, 2018). Dado esta definição, cada autor caracteriza e classifica diferentes tipos de elos dentro de seus metamodelos, de acordo com suas necessidades e conceituações. A grande gama de autores que criaram seus próprios metamodelos, geraram diversos tipos de classificações, a Tabela 5 mostra algumas delas.

Tabela 5 - Tipos de elos

Autor	Grupos	Tipos de elo
Ramesh e Jarke (2001)	Relacionado ao produto	Satisfação
		Dependência
	Relacionado ao processo	Evolução
		Rationales
Toranzo et al. (2002)		Satisfação
		Recurso
		Responsabilidade
		Representação
		Alocação
		Agregação
Pohl (1996)	Condição	Restrições
		Pré-condições
	Documentos	Exemplos
		Propósito
		Caso de teste
		Comentários
		Segundo Plano
	Abstração	Refinado
		Generalizado
	Evolução	Elaborado
		Formalizado
		Baseado em
		Satisfação
		Substituído
	Conteúdo	Similar
		Comparação
		Contradição
		Conflito
De Lucia et al. (2005)		Dependência
		Composição
		Indireto
Spanoudakis et al. (2004)		Sobreposição
		Execução_Requerida
		Característica_Requerida
		Parcialmente_Realizável
Aizenbud-Reshef et al. (2006)		Imposto
		Inferido
		Manual
	Computacional	Derivação
		Análise

Fonte: Genvigir (2009)

4.5.5.6 Matriz de rastreabilidade

A matriz é uma das ferramentas mais populares da rastreabilidade, facilitando uma visão documentada entre requisitos e objetos, permitindo cruzar os dados sendo rastreados e marcar os pontos de intersecção entre eles. Com uma matriz bem elaborada, fica mais fácil identificar os impactos que futuras alterações trarão. Geralmente é necessário que uma matriz possua: ID do requisito, nome do requisito, prioridade e situação do requisito, descrição do requisito e código de teste, dentre outros que dialogam com a necessidade de cada projeto. A Figura 13 traz uma boa referência de matriz composta em formato de tabela/planilha que é mais comum em projetos menores, quando grandes, há a necessidade de softwares específicos (ESPINHA, 2020).

Figura 13 - Matriz de Rastreabilidade

Matriz de rastreabilidade dos requisitos								
Nome do projeto:								
Centro de custo:								
Descrição do projeto:								
ID	ID associado:	Descrição dos requisitos	Necessidades do negócio, suas oportunidades, metas e objetivos	Objetivos do projeto	Entregas de EAP	Design de produto	Desenvolvimento do produto	Casos de teste
001	1.0							
	1.1							
	1.2							
	1.2.1							
002	2.0							
	2.1							
	2.1.1							
003	3.0							
	3.1							
	3.2							
004	4.0							
005	5.0							

Fonte: A Guide to the Project Management Body of Knowledge (PMBOK Guide), 5th Edition, 2014

4.5.6 Gerência da qualidade de requisitos

O objetivo desse processo consiste em garantir uma base composta por bons requisitos (SAYÃO; BREITMAN, 2007). Essa classificação de um requisito em um requisito bom possui diversas interpretações, das quais podem ser citados os critérios da Tabela 06:

Tabela 6 – Critérios de avaliação da qualidade de requisitos

Necessidade	O sistema é capaz de atingir seus objetivos sem este requisito? Caso afirmativo este é um requisito desnecessário
Verificável	É possível verificar que este requisito está sendo atendido pelo sistema?
Atingível	O requisito pode ser atendido pelo sistema que está sendo desenvolvido?
Livre de Ambiguidades	O requisito possui mais de uma interpretação possível?
Completo	O documento de especificação do sistema contém todos os requisitos?
Consistente	Todos os requisitos podem ser atendedidos sem que se entre em conflito uns com os outros?
Rastreável	A origem dos requisitos é conhecida? O requisito pode ser referenciado e localizado no sistema?
Alocação	O requisito pode ser alocado a um elemento ou componente do sistema?
Concisão	O requisito está descrito de forma simples e concisa?
Livre de Implementação	O requisito descreve o QUE deve ser feito sem influências de possíveis implementações?
Identificador único	Cada requisito possui um identificador único que permita com que possamos referenciá-lo de forma única e precisa?
Correção	O requisito contém toda a informação necessária que permita sua implementação?
Priorizável	O requisito é passível de ser priorizado frente aos outros requisitos?

Fonte: Sayão e Breitman (2007) apud Young (2001) e Wiegers (1999)

De acordo com o CMM, o *Capability Maturity Model*, do inglês Modelo de Maturidade em Capacitação [19--], criado pelo *Software Engeneering Institute* da Universidade Carnegie Mellon, a gerência da qualidade de requisitos corresponde a uma revisão dos requisitos antes de adicioná-los ao projeto. As atividades dessa área se resumem em: identificar requisitos incompletos ou ausentes; determinar se os requisitos são precisos, consistentes e verificáveis; revisar requisitos com problemas potenciais; e negociar compromissos com grupos envolvidos no projeto (CMM [19--] apud SAYÃO; BREITMAN, 2007).

4.5.6.1 Problemas encontrados durante gerência de qualidade

Grande parte dos requisitos é escrita em linguagem natural, o que causa uma falta de formalidade, que, por sua vez, resulta em uma série de problemas que devem ser corrigidos durante a gerência de qualidade (SAYÃO; BREITMAN, 2007).

Dentre esses problemas os mais comuns, segundo Sayão e Breitman (2007), tem-se a ambiguidade (falta de clareza ou duplo sentido em expressões levando, a interpretações incorretas do que realmente era necessário), requisitos incompletos (requisitos onde isentam parte da informação necessária para compreensão), requisitos múltiplos (requisito que englobam vários requisitos ao invés de um requisito específico, o que acaba prejudicando a

priorização e gerência de mudanças.), requisitos com alternativas (requisitos que se aplicam a situações gerais, sem contar qual deve ser a ação do sistema em alguma situação particular ou específica), requisitos mal escritos (podem causar mal-entendidos por serem mal redigidos, sendo o mais comum o excesso de informação) e requisitos inverificáveis (requisito que não se sabe ao certo se o sistema é capaz de atender).

4.5.6.2 Processos de verificações da qualidade de requisitos

No processo de garantia da qualidade de software deve ocorrer uma revisão periódica dos requisitos do projeto (SAYÃO; BREITMAN, 2007). Durante este processo, são atribuídas tarefas às gerências sênior, do projeto e de qualidade (CMM, [19--] apud SAYÃO; BREITMAN, 2007).

De acordo com Sayão e Breitman (2007), o CMM [19--] define que deve ser realizado pela gerência sênior a revisão periódica todas as atividades de gerência dos requisitos de software; a gerência do projeto se encarrega da revisão de todas as atividades da gerência de requisitos do software periodicamente e quando se dá ocorrência de eventos; enquanto o grupo de qualidade de software (GQS) revisa e audita as atividades e artefatos usados no gerenciamento de requisitos alocados, reportando os resultados obtidos.

Ao grupo de qualidade de software também são designadas verificações. No mínimo, verificações dos requisitos de software, dos planos de software, dos artefatos e produtos de trabalho e das atividades da gerência de requisitos devem ser aplicadas (CMM [19--] apud SAYÃO; BREITMAN, 2007).

4.5.6.3 Inspeções

Diversas técnicas para a realização dessas verificações podem ser encontradas, dependendo da formalidade necessária e do autor selecionado para a análise, sendo a mais conhecida e aplicada a inspeção, que utiliza de técnicas de leitura, como *Ad-hoc*, a baseada em checklists e a baseada em perspectivas (SAYÃO; BREITMAN, 2007). Nesta fração deste texto, a inspeção e suas técnicas de leitura serão apresentadas.

A inspeção corresponde a uma das técnicas mais bem aceitas por desenvolvedores, que envolve a leitura de um artefato a fim de localizar erros ou defeitos através de um critério pré-estabelecido, podendo ser aplicada a praticamente todos os tipos de artefato (FAGAN, 1986

apud SAYÃO; BREITMAN, 2007). Ocorre em diversas etapas e possui diferentes agentes atuando em cada engrenagem desse processo.

4.5.6.4 A função dos participantes das inspeções

Podem ser encontrados no processo diferentes agentes que realizam funções distintas na localização de defeitos e erros nos artefatos de requisitos.

Sayão e Breitman (2007) organizam esses papéis desempenhados pelos participantes do projeto em: organizadores, responsáveis pela organização e condução do processo; autor do artefato, que possui uma visão completa do artefato antes da inspeção ser aplicada; inspetor, que analisa os artefatos através de técnicas de leitura; secretário, que recebem defeitos encontrados pelos inspetores durante a inspeção e os solidifica em um documento; moderador, estes recebem o documento dos secretários, além de conduzirem a reunião de inspeção; e relatores, que apresentam os defeitos encontrados aos autores.

A reunião na qual o relator apresenta ao autor do artefato os defeitos coletados conta com a presença de um moderador, que exerce papel de mediação, e de um secretário, que registra esse encontro (SAYÃO; BREITMAN, 2007)

4.5.6.5 Etapas do processo de inspeção

O processo da inspeção decorre em etapas, que serão descritas a seguir nesta fração desse texto.

A primeira etapa corresponde ao planejamento, no qual uma pessoa da organização é responsável pela condução e organização da inspeção. Participantes são selecionados e recebem suas funções e o material que será analisado (SAYÃO; BREITMAN, 2007).

A segunda, corresponde a visão geral, que representa o início da reunião de inspeção, onde o autor do artefato o apresenta para os participantes, medida importante para os artefatos de requisitos, enquanto a terceira corresponde a inspeção em si, realizada por inspetores através de uma técnica de leitura de artefatos pré-definida, a fim de detectar os defeitos e erros deste artefato (SAYÃO; BREITMAN, 2007).

Tem-se assim, a coleção, onde os defeitos são reunidos em um documento, podendo haver uma revisão desses e uma afirmação se correspondem aos que necessitam ser corrigidos, passando para a próxima etapa, a correção, momento no qual esses artefatos voltam aos desenvolvedores para serem corrigidos (SAYÃO; BREITMAN, 2007).

Por fim, tem-se o acompanhamento, fração do processo em que são acompanhadas e verificadas as correções dos defeitos ou erros evidenciados na etapa da inspeção e, caso o número de defeitos que persistirem forem significativos, uma nova inspeção pode ser necessária (SAYÃO; BREITMAN, 2007). Na Figura 14 será possível observar a sequências de etapas da inspeção

Figura 14 – Esquema com as etapas da inspeção

Fonte: Sayão e Breitman (2007)

4.5.6.6 Técnicas de leitura

Utilizadas pelos inspetores na etapa da inspeção na análise dos artefatos, podem ser citadas as técnicas de leitura *Ad-hoc*, a Baseada em *checklists* e a Baseada em perspectivas.

A primeira depende dos conhecimentos do inspetor envolvido no processo, não havendo estrutura técnica que especifique quais informações devem ser verificadas e como a identificação de defeitos deve decorrer nessas informações, com os critérios vindos mais exclusivamente do inspetor, podendo verificar a clareza, completude, consistência, funcionalidade, detalhamento, entre outros aspectos (SAYÃO; BRETIMAN, 2007).

Na Baseada em *checklists* tem-se "o conjunto de inspetores utiliza uma mesma lista para a leitura e análise do artefato" (SAYÃO; BREITMAN, 2007, p. 24). Nessa lista, relaciona-se os itens a serem verificados e seus defeitos, portanto, para evitar com que os defeitos não caracterizados na lista não passem despercebidos, tem-se algumas regras básicas para sua construção: uma página deve ser o suficiente para as questões a serem resolvidas; essas questões precisam ser objetivas e precisas; e os atributos de qualidade devem estar presentes no artefato de maneira explícita, com as respostas para as questões levantadas devendo verificar se esse atributo de qualidade se faz presente no artefato (SAYÃO; BREITMAN, 2007).

Já a Baseada em perspectivas (PBR, ou *Perspective Based Reading*, do inglês Leitura Baseada em Perspectivas), é a técnica em que se pode "considerar as diferentes perspectivas (visões) dos atores do processo" (PORTER, 1995; SHULL, 2000 apud SAYÃO; BREITMAN, 2007, p. 25). São usadas listas diferentes, uma vez que são consideradas as diferentes visões dos diversos agentes (usuário final, projetista, programador, por exemplo), por isso, inspetores

adquirem o papel de criar um modelo seguindo uma dessas visões e analisar os requisitos conforme esse modelo, com uma lista de questões para serem levantadas (SAYÃO; BREITMAN, 2007).

Segundo Sayão e Breitman (2007), aos requisitos, essas duas últimas técnicas podem ser importantes para verificar se a informação é consistente, se os requisitos não-funcionais são explicitados, se há presença de ambiguidade ou, até mesmo, se possui requisitos incompletos, com alternativas, múltiplos, mal escritos ou incompletos. Além disso, é dito que a terceira ainda é capaz de localizar defeitos não identificados em nenhuma das duas primeiras, como um requisito em uma seção errada, por exemplo.

5 CONSIDERAÇÕES FINAIS

A partir do aumento da necessidade que se tem de mais sistemas integrando os meios digitais e a sociedade, cria-se diversas possibilidades de softwares a serem desenvolvidos, dependendo dos requisitos e funções em que cada conceito se baseia. Portanto, a obtenção e organização desses requisitos em um projeto, papel da Engenharia de Requisitos, torna-se essencial para evitar falhas ou resultados insatisfatórios às partes interessadas.

Nesse contexto, foi-se concluído o objetivo geral deste trabalho, que correspondia a apresentar os conceitos da Engenharia de Requisitos, objeto muito estudado e que já foi conceituado por diferentes especialistas. A pesquisa apresentou o significado dessa engenharia, bem como definiu e organizou os tipos de requisitos. Atrelados a definição da Engenharia de Requisitos, tem-se os objetivos específicos, que correspondem ao seu papel na Engenharia de Software e a sua divisão em etapas e subtemas. Para tais propostas, foram analisados pesquisas e materiais da área, a fim de organizar e estabelecer uma ordem para as etapas do processo e, desse modo, relacioná-lo com o desenvolvimento de softwares.

Devido aos diversos modos em que pode ser vista, analisada e reproduzida, a Engenharia de Requisitos pode ser fragmentada e suas etapas podem ser classificadas de diferentes maneiras, dependendo da interpretação de quem a estuda. Para tal questão, essa pesquisa analisou e organizou essas etapas a fim de atender a um processo linear da Engenharia de Requisitos.

Com o objetivo de se ter um software que apresente melhores resultados, ou seja, atendendo aos pedidos do cliente com um melhor custo-benefício e qualidade, surge a necessidade de, como estabelecido por Sommerville (2011) entender os requisitos e coletá-los, documentá-los e mantê-los ordenadamente e sistematicamente. A partir desse momento, pode-se afirmar que a Engenharia de Requisitos tem como objetivo ter desenvolvedores e clientes com a mesma visão do projeto. Neste contexto, realizou-se a apresentação de requisito baseada nas definições do IEEE (1990) e Sommerville (2011).

A Engenharia de Requisitos pode ser dividida em etapas, como a elicitação, análise, validação, documentação e gerência de requisitos. Em linhas gerais, após a construção dessa pesquisa, pode-se dizer que o processo se inicia com a interação dos engenheiros com os *stakeholders*, durante a elicitação, onde ocorre a obtenção dos requisitos do sistema.

Como a primeira etapa decorre de uma reunião, as demandas obtidas devem ser organizadas, o que ocorre justamente durante a análise de requisitos. Segundo Vasquez (2016),

a análise ajuda na compreensão destes, uma vez que acabam entrando em conflito entre os demandados pelos clientes ou pelos usuários, por exemplo. Assim, o próximo passo é a documentação, etapa na qual é realizada uma declaração oficial que deve ser implementada pelos desenvolvedores. Nesse momento, entra a especificação desses requisitos, momento no qual são classificados, por exemplo, em funcionais e não-funcionais seguindo uma série de diretrizes e padrões.

Com os requisitos obtidos, analisados e, nesse ponto, documentados, entra a validação, onde é verificado se eles atendem aos desejos dos clientes e se possuem problemas antes do sistema entrar em operação. Nessa etapa também ocorre o estudo de viabilidade, que analisa, a partir de diferentes pontos de vista, se o projeto é viável, estando presente também na análise da necessidade desse sistema para a empresa, que ocorre ainda antes do processo de elicitação.

Feitos todos esses processos, durante o desenvolvimento de um projeto, mudanças podem ocorrer, fazendo surgir a necessidade de alteração ou adição de novos requisitos, sendo necessário um gerenciamento de requisitos, como dito por Vasquez e Simões (2016 apud CMMI-DEV). Nessa etapa ocorre o controle de mudanças, a gerência de configuração, a priorização de requisitos, a rastreabilidade e a gerência de qualidade de requisitos. Esses procedimentos da gerência de requisitos envolvem, como já desenvolvido anteriormente seguindo as ideias de Sayão e Breitman (2007), Engholm (2013), Vasquez e Simões (2016), Revelo (2019), et. al., um acompanhamento dos requisitos, das mudanças que podem ocorrer, seus impactos e controle delas no projeto, da origem e de pontos de interligação entre requisitos, além de garantir que o projeto tenha requisitos precisos e concisos.

Dadas essas informações, pode-se dizer que o processo de desenvolvimento de um software demanda de um profundo entendimento das necessidades do meio em que ele será inserido. A Engenharia de Requisitos decorre nesse processo exercendo o papel de padronizar, organizar e sistematizar a obtenção de pedidos de demandas a serem cumpridas pelo sistema que será desenvolvido. Sua importância é intrínseca para o entendimento dos requisitos por todas as partes envolvidas no projeto.

Como demonstrado pelo relatório CHAOS Report, do The Standish Group (2015), mencionado no início deste trabalho, são poucos os projetos de software que são concluídos com sucesso. Muitos desses sistemas acabam falhando por problemas em custo e prazo, dados que podem ser mais bem previstos, até mesmo evitados, com a aplicação das técnicas da Engenharia de Requisitos.

Conclui-se evidenciando que as etapas dessa engenharia não anulam todas as chances de erros de desenvolvimento, porém, permitem que haja uma base melhor estruturada e que

todas as partes envolvidas no projeto tenham um padrão de registro de requisitos em mãos, diminuindo as múltiplas interpretações que podem decorrer e, consequentemente, também diminuindo as chances de erros que, até serem corrigidos, terão aumentos nas questões financeiras, técnicas e relacionadas ao tempo, como apontadas no relatório do The Standish Group (2015). Com isso, pode-se dizer que quando há uma análise e uma classificação dos pedidos a serem atendidos de acordo com o que as partes interessadas no produto desejam, uma padronização e uma organização dos procedimentos a serem efetuados durante a produção de um software, tem-se menores probabilidades de ocorrerem crises e, consequentemente, frações mais expressivas de produtos com resultados satisfatórios.

É importante destacar que ao longo do aprofundamento deste trabalho, diversos modos de se organizar os processos da engenharia surgiam, devido à amplitude de interpretações possibilitada por ela. Com isso, alguns nomes estavam presentes em algumas passagens e, apesar de terem sua importância justificada por determinado autor, acabavam não entrando para as etapas na visão de outro profissional.

Portanto, pode-se dizer que há várias outras maneiras de se explorar os conceitos da Engenharia de Requisitos. Ao basear-se nas análises de outros autores para o desenvolvimento de uma pesquisa, outras possibilidades de interpretação das etapas do processo são abertas. Além disso, pode-se destacar a possibilidade de um maior aprofundamento e um estudo separado e ainda mais especificado sobre as particularidades de cada subtema aqui apresentado. Apesar deste texto não incluir esse amplo leque de opções, ele abre espaço para futuras dissertações e pesquisas acerca das definições e de um aprofundamento das etapas do tema.

REFERÊNCIAS

CORDEIRO, A. G. **PRIORIZAÇÃO DE REQUISITOS E AVALIAÇÃO DA QUALIDADE DE SOFTWARE SEGUNDO A PERCEPÇÃO DOS USUÁRIOS**. UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO. CAMPOS DOS GOYTACAZES - RJ, p. 96. 2010.

COSTA, F. Técnica MoSCoW na Priorização dos Requisitos. **Site Campus**, 2018. Disponivel em: <https://sitecampus.com.br/tecnica-moscow-na-priorizacao-dos-requisitos/>. Acesso em: 15 jun. 2021.

DIAS, C.; SOUZA, A. C.; MACEDO, M. **AADSP GERENCIA DE REQUISITOS**. São Paulo: Editora Ixtlan, 2018.

DIAS, F. O que é Gerência de Configuração de Software? **Blog Pronus**, 2016. Disponivel em: <https://blog.pronus.io/posts/controle-de-versao/o-que-eh-gerencia-de-configuracao-de-software/>. Acesso em: 14 jun. 2021.

DUAN, C. . L. P. . C.-H. J. E. A. In: _____ **Towards automated requirements prioritization and triage. Requirements Eng 14**. [S.l.]: [s.n.], 2009. p. 78-79.

ENGHOLM, H. Gerenciamento e Controle de Mudanças. **Ti Especialistas Brasil**, 2013. Disponivel em: <https://www.tiespecialistas.com.br/gerenciamento-controle-mudancas/>. Acesso em: 13 jun. 2021.

ESPINHA, R. G. Matriz de Rastreabilidade de Requisitos: saiba como gerenciar as mudanças no escopo. **Artia**, 2020. Disponivel em: <https://artia.com/blog/matriz-de-rastreabilidade/>. Acesso em: 19 jun. 2021.

LAGUNA, F. Gerenciamento de requisitos sem mistério. **Youtube**, 2016. Disponivel em: <https://www.youtube.com/watch?v=jajQyzOpLaE>. Acesso em: 11 jun. 2021.

NASCIMENTO, L. A. D. Softwares, Sistemas e Aplicações: a evolução no mundo das soluções. **https:** //administradores.com.br/artigos/softwares-sistemas-e-aplica%C3%A7%C3%B5es-a-evolu%C3%A7%C3%A3o-no-mundo-das-solu%C3%A7%C3%B5es, 2020. Disponivel em: <https://administradores.com.br/artigos/softwares-sistemas-e-aplica%C3%A7%C3%B5es-a-evolu%C3%A7%C3%A3o-no-mundo-das-solu%C3%A7%C3%B5es>. Acesso em: 07 jun. 2021.

REVELO. O que é controle de versão de software e como usar no seu projeto. **Revelo**, 2019. Disponivel em: <https://blog.revelo.com.br/controle-versao-software-como-usar/>. Acesso em: 14 jun. 2021.

SAYÃO, M.; BREITMAN, K. K. **Gerência de Requisitos**. PUC-RIO. Rio de Janeiro, p. 31. 2007.

SAYÃO, M.; LEITE, J. C. S. D. P. **Rastreabilidade de Requisitos**. PUC-RIO. Rio de Janeiro, p. 18. 2005.

SOMMERVILLE, I. **Engenharia de Software**. 9ª. ed. [S.l.]: Pearson, 2011.

VAZQUEZ, C. E.; SIMÕES, G. S. **Engenharia de Requisitos:** software orientado ao negócio. 1. ed. [S.l.]: Brasport, v. 1, 2016.

VENTURA, P. Priorização de Requisitos. **Até o momento**, 2015. Disponivel em: <https://www.ateomomento.com.br/priorizacao-de-requisitos/>. Acesso em: 15 jun. 2021.

Lehtola, L.; Kauppinen, M. (2004) Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Projects. Proceedings of the European Software Process Improvement Conference (EuroSPI 2004), p. 161-170.

Duan, C.; et al. (2009) Towards automated requirements prioritization and triage. Requirements Engineering 14:73–89.

DAVIS, A. M. Software requirements: objects, functions and states. New Jersey: Prentice–Hall, 1993. 39, 40, 42

TAVEIRO, Fabio Tozetto. **Análise do impacto de um requisito não funcional relacionado a usabilidade**. Monografia (Curso de Pós-graduação - Especialização em Gestão da Tecnologia da Informação) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Campus São Paulo. São Paulo, p.103. 2016

WOJEWODA, S.; HASTLE, S. Standish Group 2015 Cahos Report – Q&A with Jennifer Lynch. **InfoQ** [online], 04 out. 2015. Disponível em: <https://www.infoq.com/articles/standish-chaos-2015/>. Acesso em: 04 jun. 2021

DIDIER, A. C. V. B. **WRE-Process: Um processo de Engenharia de Requisitos Baseado no RUP.** 2003. 246 f. Dissertação (Mestrado em Ciências da Computação) – Universidade Federal de Pernambuco, 2003. Disponível em: <https://repositorio.ufpe.br/bitstream/123456789/2477/1/arquivo4654_1.pdf>. Acesso em: 13 jun. 2021.

PINOTTI, Mateus Monteiro. **Diretrizes para guiar o processo de Engenharia de Requisitos a partir de Modelos Organizacionais desenvolvidos em EKD.** 2004. Dissertação (Mestrado em Ciências da Computação) – Instituto de Ciências Matemáticas e de Computação – ICM-USP, 2004.

FILHO, A M. S. Artigo de Engenharia de Software 3 – Requisitos Não Funcionais. **Devmedia**, 2008. Disponível em: <https://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525>. Acesso em: 21 jun. 2021.

CAMPOS R. Requisitos de sistema. **Youtube**, 2020. Disponível em: <https://www.youtube.com/watch?v=e8221YxMnQc&list=PLm2YCT58S2dBGDDgJasQV_Yyfnd7Y0Byx&index=8&t=1s>. Acesso em: 21 jun. 2021.

CAETANO G. Tipos de Requisitos // Engenharia de Software #03. **Youtube**, 2020. Disponível em: <https://www.youtube.com/watch?v=kTusEEsdTwY>. Acesso em: 21 jun. 2021.