Para resposta desta pergunta o conhecimento sobre o que é um requisito é extremamente importante, deste modo podemos dizer que é uma declaração de uma característica que o sistema deve ter ou deve fazer. Um bom requisito resume-se a uma afirmação que possa ser verificada, este também tem de representar as necessidades, as restrições e condições da característica. Como tal podemos dizer que um requisito deve ser SMART, isto é, específico, mesurável, realizável, relevante e temporalmente limitado.

Quando os clientes/*stakeholders* dão requisitos demasiado vagos dá oportunidade aos programadores implementar as coisas da maneira que eles pensam que seja melhor. Isto leva a uma falha, pois muitas vezes na implementação na vista do programador não era igual à visão que o cliente tinha sobre o projeto, visto que a vagueza do requisito, deu muita pouca orientação ao programador. Tendo isto em conta pode-se dizer que quando ambas as partes fazem o que lhes dá mais jeito o produto final muitas vezes não é aquilo que o cliente desejou, o que pode tornar a qualidade do produto muito má.

Existem boas práticas que façam com que os requisitos estejam a um nível adequado, ou seja, mais concretos/específicos, para que o produto saia de acordo com a visão do cliente e deste modo tenha uma qualidade muito superior.

De acordo com o *OpenUp* tais práticas traduzem-se em:

- Entrevistas
- Desenvolvimento da aplicação em conjunto(JAD)
- Questionários
- Análise de documentação
- Observação

Estas técnicas podem dar diferentes níveis de profundidade de conhecimento sobre o requisito e todas possuem diferentes custos. Tal como possuem diferentes custos, também possuem diferentes características, umas são melhores para conhecimento global do projeto como a analise documental, outras com mais profundidade a nível do conhecimento do projeto como as entrevistas. De todas estas técnicas a que pode dar melhores resultados será a JDA, isto porque implica que todos os elementos, clientes e programadores, se juntem para o design da aplicação o que leva a um desenvolvimento de uma visão coletiva sobre o que realmente o utilizador quer dando assim requisitos muito bem orientados para os programadores. O uso de uma técnica não implica que as outras sejam descartadas, muito pelo contrário, quantas mais técnicas forem utilizadas maior a probabilidade dos requisitos serem de maior qualidade. O que ajuda a criar também soluções de alta qualidade.

Existem também técnicas alternativas para recolher requisitos, tais como, Análise do domínio, em que se estuda as características de um domínio e se retira as partes reutilizáveis. Mapas de Conceitos, que se foca em representar ligações importantes entre os principais conceitos. *User Stories, Story Cards e Task Lists*, bastante acessíveis tanto em termos de custo como a nível de *updates*, e necessita de pouca tecnologia para ser implementado, esta técnica está muito associada com o desenvolvimento de software ágil e funciona bastante bem para capturar tanto requisitos funcionais como não funcionais.

Em suma, o processo de obtenção de requisitos, *requirement elicitantion* , é um processo colaborativo e analítico, que é o aspeto mais critico de um projeto de desenvolvimento de software e resume-se à identificação de necessidades e restrições dos vários *stakeholders* para um sistema de software.