Geiser Chalco Chullco (H)

① Processos Ágeis de Software

O processo de desenvolvimento de software é um conjunto de atividades/tarefas ~~efetuadas a serem efetuadas~~ que realizadas para a concepção, desenvolvimento e manutenção de software. Essas atividades/tarefas visam garantir que o software consiga garantir qualidade do produto - entendido qualidade como a satisfação dos stakeholders (partes interessadas) ~~envolvidos~~ no ~~Software~~ contexto onde será o software utilizado.

Para atingir essa qualidade, diversos ~~modelos de processo~~ modelos de processo de Software tem sido desenvolvidos desde o surgimento da Engenharia de Software. Os primeiros modelos foram os prescritivos, tais como o modelo cascata, modelo V, modelo espiral e o RUP (Rational Unified Process). Esses modelos são prescritivos porque antes da construção do software são explicitamente indicados em documentos o que o software irá fazer e como irá ser feito isso.

Modelos prescritivos segurem a qualidade de software segurando a qualidade da comunicação ~~entre~~ máxima possível entre os stakeholders ~~e a equipe~~ e entre a equipe de desenvolvedores,

No entanto, requeria muita documentação para sequenciar de forma explícita essa comunicação. Além disso, tinham origem e desenvolvimento na década onde softwares de grande porte e segurança eram desenvolvidos. Com a evolução dos computadores e redução de custos, começou sua popularização e precisou-se de softwares para stakeholders que muitas vezes não sabe o que deseja ou tem dificuldade em expressar suas necessidades. Surgiram equipes pequenas de desenvolvimento de software (equipes de 3-5, por exemplo), os quais não poderiam alocar ~~todos~~ seus membros nas atividades/tarefas requeridas pelos processos prescritivos sem ocasionar atrasos nas entregas. Além disso, ~~muitas~~ as mudanças tecnológicas, fazem com que alguns requisitos iniciais mudem totalmente o foco do projeto de software. Sendo que os processos prescritivos usualmente requerem de longo tempos para serem entregues.

Com esses problemas, que muitas vezes, fizeram que muitos projetos de software ficarem apenas no papel (documentação, especificação de requisitos, arquitetura), surgem os ~~projetos~~ processos ágeis de software, os quais tem foco no produto,

mais entregas, ~~se~~ fornecem para os stakeholders o mínimo ~~se~~ fundamental para resolver suas necessidades.

Processos Ágeis são fundamentados na entrega contínua do produto, o qual não é necessariamente o ~~produto~~ Software final, a entrega de ~~uma~~ protótipos que a cada entrega tornam-se o produto final. Isso é a entrega iterativa e incremental, em prazos curtos, e a constante feedback dos stakeholders. Com essa abordagem, a comunicação entre os desenvolvedores e stakeholders é mais constante.

Processos Ágeis não eliminam a documentação, mas a reduzem ao mínimo necessário para conseguir uma qualidade aceitável do Software. A validação e avaliação do Software é feita em cada entrega e pelos usuários finais.

Exemplos de Processos Ágeis são o XP (Extreme Programming) e **Scrum**. Além dos prazos fixos de entrega (15-21 dias), Entregáveis iterativos/incremental, o XP possui alguma ~~restrição~~ práticas que fizeram com que hoje não seja o modelo mais adotado no Brasil. Por exemplo,

- Pair Programming (programação em pares)
- Escritório do Product Owner (principal stakeholder) como parte da equipe de desenvolvimento
- Papo-em-pé diário

. O modelo Scrum é um pouco mais flexível com essas práticas. ✱4 (4)

Ambos modelos possuem a prática de uso de Kan-Ban para documentar requisitos e atividades, a entrega Iterativa/Incremental, ~~a diferença~~ entre outros.

Pela entrega frequente e retorno imediato do feedback é possível dar suporte rápido e frequente as mudanças de tecnologia ou novos requerimentos.

Por esse motivo, processos ágeis, tem-se, hoje, convertido no processo empregado por várias empresas de software. e o modelo mais adotado.

Casos de Uso e Requisitos de Software

Todo Projeto de software envolve com a elicitação de requisitos de software. Requisitos são as necessidades desejos, oportunidades e o que espera ser resolvido por um software quando é inserido no contexto dos stakeholders (pessoas interessadas ou afetadas pela informação que será processada pelo software). Identificar esses requisitos precisam de uma ferramenta que possibilite a comunicação clara, explícita e não ambígua entre os desenvolvedores do software e os stakeholders.

Casos de Uso é a ferramenta pela qual são descritos os requisitos de software de forma explícita e clara (sem ambiguidades). Dois pontos essenciais definem um caso de Uso:

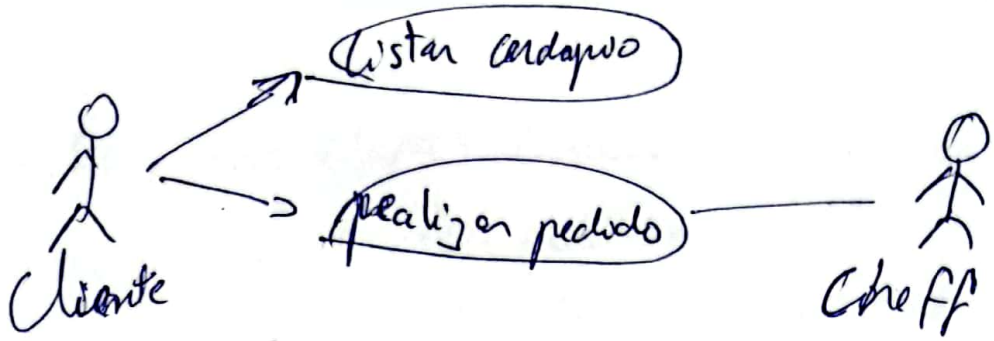
- ~~O objetivo~~ ~~O que é esperado do software~~
- O comportamento esperado do software (o objetivo)
- Os stakeholders que acessam esse comportamento (atores)

-

. Adicionalmente é recomendado especificar o porque se precisa de esse comportamento. Isso é frequentemente realizado para conseguir detalhar de forma adequada o melhor comportamento esperado e ajuda frequentemente na inovação.

Casos de uso são definidos pelos próprios desenvolvedores ou Engenheiro de Software após reunião com os stakeholders ou observação de oportunidades nos contextos/situações onde os stakeholders estão envolvidos (precisando do software)

Diferentes formas de representação de caso de uso tem sido propostas na literatura desde formulários, diagramas e cartões. Sendo a mais popular a de diagramas UML onde os caso de uso são representados como círculos com um nome que brevemente descreve o comportamento esperado do software e os stakeholders que precisam desse comportamento no lado esquerdo, enquanto os stakeholders que atendem alguma demanda do software são localizados na parte direita do diagrama. A seguir é apresentado um diagrama de caso de uso para realizar pedidos online de comida para um restaurante.



Para simplificar o exemplo, consideraremos que não existe pagamento online, apenas o pagamento será na entrega do produto. Vale-se que nesse exemplo há (02) dois casos de uso, o primeiro referido à listagem de pratos que o restaurante oferece - no qual é esperado que o software mostre para o cliente a lista de pratos disponíveis (objetivo do software) e sendo o cliente o ator do caso de uso.

No segundo caso de uso de nosso exemplo, pode-se observar que há dois stakeholders sendo atores do caso de uso "realizar pedido". O comportamento esperado do sistema é que ao "realizar pedido", o cheff (cozinheiro) do restaurante confirme o pedido (seja notificado) e inicie a elaboração do prato solicitado no pedido.

Caso de uso descreveria de forma explícita, o que é esperado do software (comportamento) e os stakeholder que ~~[illegible]~~ serão afetados/beneficiados com o comportamento (atores). Com essa ~~[illegible]~~ especificação é possível mitigar problemas

no desenvolvimento de software, em particular, em etapas iniciais, na sua concepção. A clareza dos diagramas UML para caso de uso tem feito dessa ferramenta a principal utilizada hoje no desenvolvimento de software, além de sua simplicidade e facilidade de uso.

(3) Contagem de Pontos de Função

Estimar o esforço necessário para um projeto de software, seu custo e o tempo requerido, além da equipe necessária para realizar seu desenvolvimento, é uma atividade/tarefa muito frequente e necessária em Engenharia de Software.

Durante décadas diferentes métodos e métricas de estimação tem sido desenvolvidos, entre eles KLC (Quantidade de linhas de código), complexidade do código, e Pontos de Função. Sendo essas últimas as consideradas mais relevantes e adoptadas, não somente no âmbito académico, mas também na indústria de Software.

Pontos de função são as métricas que definem as funcionalidades que um software possue para atender os requisitos dos stakeholders.

. Efetuar a contagem de Pontos de Função de uma Software existente, tanto quanto de um software que esta na fase de concepção ou elaboração, precisam de habilidades de decomposição do Software em funções básicas que são definidas pela práxis de desenvolvimento de software.

Entende-se por funções básicas as funções de:

- ~~relatórios de~~ saída: emissão de algum relatório de informação, seja, textual, gráfico, combinação das duas, é considerado um ponto de função de saída Cada relatório requerido pelos stakeholders. Relatórios de formatos diferentes são considerados pontos de função diferentes, mesmo que eles possuam a mesma informação. No entanto relatórios que possuam mais de uma parte ~~são~~ são considerados apenas como um único ponto de função. O que muda é a complexidade da função que será uma métrica utilizada no cálculo de custo/esforço/tempo necessário para o seu ajuste.
- entrada: dados de entrada para seu processamento são um ponto de função. O software é um artefato que processa dados/informações para gerar informações de valor agregado para os stakeholders no contexto que estão envolvidos. Nesse sentido,

Um ponto de função de entrada deve ser considerado como todos os dados solicitados ao stakeholder para processar eles na informação de valor agregado que deseja. Isso é o conjunto de dados, não cada dado, solicitado ao stakeholder para produzir alguma saída.

- Consultas: ~~Reportes~~ Relatórios do processamento de informação podem ser externos (saídas do software), sendo assim necessário uma filtragem por parte do stakeholder. Um software que fornece a possibilidade de filtrar os dados de saída, fornece uma nova função, denominada ponto de função de consulta. Cada suporte a consulta específica da informação de saída do software deve ser considerada um ponto de função.

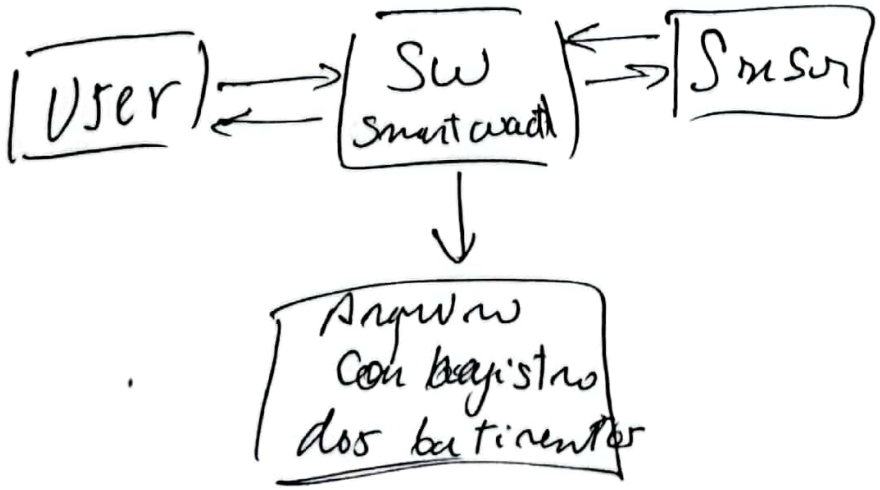
- ILF (Arquivo Lógico Interno) Internal logic file: Conceituado (~~criação, edição, manipulação~~) de um arquivo interno do software deve-ser contabilizado como um ponto de função. O processamento de dados que um software realiza pode precisar de persistência, para isso é necessário a manipulação de arquivos.

O manipulação de arquivos regem a criação, gerenciamento, atualização, assim como outras atividades que deverão ser programadas durante o desenvolvimento. Sendo assim, ~~todas elas consideradas~~

elas consideradas um ponto de função. (11)
Gerenciamento de cada arquivo é um ponto de função.

- Consultas Externas: O processamento de informação em um software frequentemente precessa de informação de um outro software ou sistema externo, seja por meio de uma API ou outro mecanismo. O processamento dessa informação externa no software precessa de transformação, adaptação e ainda frequentemente um esforço pela equipe de desenvolvimento para entender a API. Assim, cada mecanismo externo participante em software é considerado um ponto de função básico.

A seguir exemplificamos a contagem de pontos de função para o software ~~que consta~~ de Smartwatch, para simplificar o exemplo o Smartwatch apenas apresenta a função de batimentos cardíacos.



	PF
entradas	01
Saídas	01
ILF	01
External	01
	04

No exemplo, pode-se contar 04 pontos de função. Sendo o primeiro deles o de entrada quando o usuário solicita medir os batimentos cardíacos. O segundo é consulta ao sensor (consulta externa) e seu correspondente processamento de informações. O terceiro é a saída que é o relatório de batimentos cardíacos corrente do usuário. Finalmente, o quarto ponto de função é o cadastro da medida num arquivo que salva todos os registros do batimento cardíaco.

Pontos de função são uma métrica necessária para estimar custo, esforço e tempo no qual será executado um projeto de software. Ele é baseado na decomposição de software em funções básicas - pré estabelecidas pela praxis de Engenharia de Software.

Para o cálculo de custo o número de Pontos de função deverá ser ajustado de acordo com sua complexidade e multiplicado por horas homem para seu desenvolvimento.