Tópico 1: Lucile M. de Sanz Brito (A)

Com a evolução da computação e a crescente demanda por softwares para uso nas mais diversas aplicações em nosso dia a dia, surgiu a necessidade de definir novos modelos de processo para desenvolvimento de software. Isso se deve ao fato dos processos tradicionais primarem por etapas bem definidas e documentadas da concepção e entrega do software ao cliente, tornando o processo longo e demandando muito tempo até que o cliente tivesse em mãos uma versão utilizável do sistema. Em virtude disso, os profissionais da área de engenharia de software se reuniram para tentar resolver o problema e chegaram no que ficou conhecido como manifesto ágil, no qual afirmam que, em linhas gerais:

* software funcionando é mais importante do que documentação longa e completa
* satisfação do cliente é mais importante que contrato assinado
* comunicação com o cliente é mais importante que processos rígidos
* Adaptação a mudanças

1

Em resumo, essa manifestação não significa que não é necessário haver a documentação do software, a existência de contratos ou um processo bem definido, além de outras características marcantes no ciclo de vida de desenvolvimento de software tradicional, mas que essas questões são menos importantes que a entrega de um software funcional que atenda as necessidades do cliente e, consequentemente, sua satisfação.

Em consideração ao manifesto ágil foram propostas as metodologias ágeis. Nessas metodologias existe uma intenção clara de se obter um produto de software viável para o cliente no menor tempo possível. A comunicação constante com o cliente é um elemento fundamental para as metodologias ágeis.

Algumas características dos processo ágeis são:

* A entrega constante de versões utilizáveis do software;
* O alinhamento constante entre o cliente e as equipes de desenvolvimento;
* O tratamento de mudanças que surgem durante o processo de desenvolvimento;
* Validação constante com o cliente se o que

está sendo entregue corresponde ao que foi solicitado;

* A existência de parca documentação do software desenvolvido.

Embora a agilidade na entrega de incrementos utilizáveis do software ao cliente seja um dos principais pontos positivos dos processos ágeis, eles também possuem críticos. Por exemplo, em cenários nos quais os requisitos mudam com muita frequência pode ser difícil obter versões do software que atendam as demandas do cliente, existe uma demanda muito grande pela disponibilidade do cliente para interação com as equipes de desenvolvimento (e quando o cliente não dispõe desse tempo para dar atenção às equipes de desenvolvimento, o uso desses processos não é indicado) e a gestão das entregas pode ser um desafio para o gerente do projeto por, muitas vezes, haver entregas parciais de funcionalidades.

Para entender melhor o funcionamento dos processo ágeis vamos considerar como exemplo a programação extrema (XP).

A XP é uma das principais metodologias ágeis utilizadas nos dias de hoje, embora seja uma

das precursoras.

No XP, são usadas as histórias do usuário. Nelas o usuário descreve de maneira breve e direta os requisitos do sistema (as funcionalidades que ele espera encontrar no sistema). Essas histórias podem ser escritas sob uma perspectiva de tipos de usuário do software geralmente na forma de

Como <descrição do tipo de usuário> eu gostaria de <descrição da ação a ser executada>

Considere como exemplo um software utilizado para controle de estoque. Um requisito para esse sistema usando histórias de usuário poderia ser

Como estoquista eu gostaria de alterar a quantidade em estoque de um produto.

Neste exemplo o tipo de usuário "estoquista" tem interesse em realizar a ação de alteração da quantidade em estoque de um dado produto. Note que os detalhes sobre como essa ação será realizada pelo software não são apresentados, já que o propósito das histórias do usuário é somente apresentar as funcionalidades esperadas pelo usuário no sistema. Os detalhes passos para implementação de funcionalidade fica por conta das equipes de desen-

volvimento.

As histórias do usuário são geralmente escritas em cartões. Esses cartões são analisados pelas equipes de desenvolvimento para que haja um entendimento do que o usuário espera do sistema. Uma vez que haja esse entendimento, a equipe de desenvolvimento se reúne com o cliente para realizar a validação. Durante a validação, o usuário pode confirmar se o entendimento da equipe a respeito daquela funcionalidade está correto ou incorreto. Além disso, nessa fase também pode haver a negociação de funcionalidade com o cliente, caso o que foi pedido não possa ser implementado exatamente como descrito ou esteja em conflito com outra funcionalidade já implementada no sistema. Informações sobre o processo de validação ou outras informações consideradas importantes para a equipe de desenvolvimento podem ser anotadas no verso do cartão.

Com base nas histórias do usuário e nas prioridades definidas para as histórias, o gerente do projeto seleciona as histórias que serão desenvolvidas e dá início à implementação de história. Na XP, o teste unitário

é desenvolvido antes da funcionalidade, empregando o conceito de desenvolvimento dirigido a teste (TDD). Então; a equipe de desenvolvimento primeiro pensa e implementa o teste unitário de funcionalidade (que ao ser executado pela primeira vez vai falhar, já que o código de funcionalidade ainda não foi implementado) para só depois implementar o código de funcionalidade.

Outro aspecto importante de XP é a programação em pares, tendo sempre uma dupla de programadores trabalhando juntos. Isso ajuda tanto no desenvolvimento/evolução dos programadores, já que uma boa prática é colocar um programador junior com um senior, quanto na revisão do código, já que enquanto um está programando o outro está revisando o código produzido. Além disso, a programação em pares ajuda a manter a equipe focada e uniformiza a maneira de codificar entre os membros da equipe.

Uma vez que a funcionalidade tenha sido desenvolvida, ela é validada com o cliente e aceita, caso esteja de acordo com o especificado, ou ajustada, caso o entendimento e implementação de funcionalidade não estejam corretos.

Essa validação constante após a implemen

tação ajudam a tornar os sistemas desenvolvidos com XP (e outros metodologias ágeis) mais amigáveis a mudanças.

Tópico 3:

Como afirmado no tópico anterior, requisitos de software são funcionalidades que devem ser providas pelo sistema e sob quais condições. As funcionalidades propriamente ditas são os chamados requisitos funcionais, enquanto que as condições para a disponibilização destas funcionalidades são os chamados requisitos não funcionais.

Existem na literatura diferentes formas de identificar, analisar, validar, documentar e gerenciar requisitos, sendo esses passos comuns a qualquer ciclo de vida de requisitos.

Na identificação de requisitos é comum que a equipe responsável pelo desenvolvimento do software em questão realize reunião com o cliente, a fim de entender melhor o problema a ser resolvido e a funcionalidades desejadas pelo cliente, além de leitura de documentos de referência sobre o negócio no qual o software irá operar e o acompanhamento de perto das atividades diárias no ambiente no qual o software irá operar. Essas são apenas algumas alternativas para as medidas que podem ser adotadas

5

pela equipe de desenvolvimento na fase de levantamento de requisitos

Uma vez que se tenha realizado o levantamento de requisitos é necessário usar uma ferramenta para representar esses requisitos de modo que sejam entendidos tanto pelas equipes de desenvolvimento quanto pelos clientes. Por isso, essa ferramenta tem que ser simples, fácil de usar e entender, já que desempenha um importante papel facilitador para a análise dos requisitos (garantia de que são completos, íntegros, não conflitantes, testáveis e consistentes) e, consequentemente, sua qualidade.

Os casos de uso são uma das alternativas viáveis para a representação de requisitos. Eles são usados para descrever como um usuário (geralmente uma pessoa, mas também pode ser um hardware ou outro software, por exemplo) interage com o sistema. É uma forma simples, sistemática e amplamente conhecida para representar essas interações.

No contexto de requisitos, os casos de uso podem ser usados para ilustrar como um dado requisito pode ser percebido no sistema,

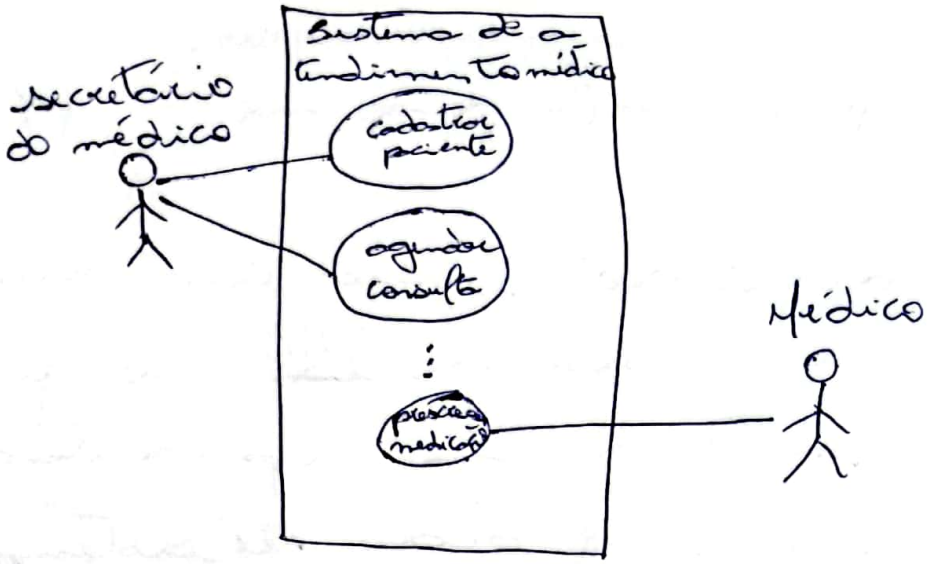
ou seja, de que forma o usuário terá acesso àquela funcionalidade no sistema.

Para ficar mais claro, vamos considerar o diagrama de caso de uso da UML, no qual os usuários do sistema são representados por e as funcionalidades do sistema por ○. Existe uma linha ligando um usuário a uma funcionalidade sempre que o usuário tiver acesso àquela funcionalidade no sistema.

Ainda para fins de exemplo, considere um sistema de atendimento médico, para o qual foram levantados os seguintes requisitos:

* O secretário do médico deve ser capaz de cadastrar pacientes;
* O secretário do médico deve ser capaz de agendar consulta;
* O médico deve ser capaz de prescrever uma medicação ao paciente.

Ao utilizar um diagrama de caso de uso para representar esse cenário, podemos perceber claramente aqueles requisitos que estão relacionados com cada tipo de usuário. Vejamos esse aspecto na figura a seguir



É importante destacar que o exemplo acima não contempla uma lista exaustiva dos requisitos possíveis para um sistema de atendimento médico, nem todos os usuários possíveis, servindo apenas para ilustrar a discussão.

Dito isso, quando a equipe de desenvolvimento utiliza os casos de uso no ciclo de vida dos requisitos, ela podem se valer de diferentes vantagens:

* Na etapa de identificação de requisitos a equipe já pode usar os casos de uso para estruturar os requisitos;

* Na etapa de análise dos requisitos a identificação de problemas sintáticos (representação incorreta) e semânticos (requisito incorreto) são facilitados pela representação visual provida pelo diagrama de caso de uso. Além disso, torna mais fácil para o usuário identificar como foi o seu entendimento para o requisito e validar se o entendimento está correto;

* Na validação (que pode ser realizada junto com a etapa de análise), os casos de uso representam uma forma não ambígua de representar os requisitos e ter a garantia de que aqueles conjunto de requisitos aprovados não serão interpretados de forma diferente no futuro;

* Na etapa de documentação, a própria utilização dos casos de uso e dos diagramas de caso de uso é uma forma de documentação dos requisitos, o que contribui para agilizar o processo de documentação e, consequentemente, a produção do sistema;

* Na etapa de gerenciamento dos ~~req~~ requisitos, torna-se fácil para pessoas que não participaram do processo de identificação dos requisitos iniciais e compreender os requisitos e realizar os ajustes apropriados.

De certa forma, o uso de casos de uso durante o ciclo de vida dos requisitos contribui para a melhoria da qualidade do software produzido pela visão do usuário, pois uma vez que se tenha requisitos corretos e validados pelo usuário, há uma chance maior de o software final atender as expectativas do usuário.

A discussão aqui apresentada está mais alinhada com o processo de desenvolvimento tradicional, baseado no modelo ~~de~~ em cascata, ~~[illegible]~~

example.

Tópico 9:

Devido a natureza subjetiva dos softwares, existem diferentes fatores que afetam as avaliações de qualidade e estimativas de projetos de software. Além disso, com a dinamicidade do mercado de desenvolvimento de software e a competitividade entre as empresas, obrigam que estas adotem alguma maneira viável para fazer suas estimativas de espaço necessário para conduzir um projeto de desenvolvimento. Sem essa medida, por exemplo, a empresa corre o risco de fazer uma definição de preço para o desenvolvimento de software superestimado e não conseguir fechar o negócio, ou fazer uma definição de preço subestimado e não conseguir cobrir os custos do desenvolvimento, ocorrendo prejuízo para empresa.

Neste sentido, foram propostas diferentes métricas que visam realizar alguma estimativa com relação ao tamanho do software em questão, métricas essas que são definidas como métricas de produto (já que podem ser definidas em termos relacionados ao processo de produção do software e a organização). A contagem de pontos de função é uma dessas maneiras de estimar o tamanho de um software. 8

A contagem de pontos de função é focada na contagem de funcionalidades que são percebidas pelo usuário do software. A ideia é ter uma forma de estimar o número de funcionalidades disponibilizadas para o usuário sem se preocupar com detalhes de implementação ou linguagem de programação utilizada. Essas características, inclusive, contribuíram para que a contagem de pontos de função seja hoje uma das formas mais utilizadas por empresas (privadas, mas principalmente, públicas) para dimensionar o tamanho de software contratado para desenvolvimento.

O processo de contagem de ponto de função começa com a definição do escopo para a contagem e o tipo de contagem a ser realizada. No que se refere ao tipo de contagem, ele pode ser de todas as funcionalidades percebidas pelo usuário numa versão final do software, funcionalidades afetadas por um conjunto de mudanças a serem realizadas ou o as funcionalidades existentes em uma dada versão de software. No que se refere ao escopo da contagem, a definição pode ser de interesse realizar a contagem de pontos de função de um

software inteiro ou de partes de software.
Após essas definições é feita a contagem propriamente dita. Nesta etapa, são considerados os pontos de manipulação de dados e as transações efetuadas. Após a contagem, pode-se adotar um fator de ajuste para a contagem, que pode ser vista como uma constante de calibração da contagem. Com o uso do fator de ajuste e da contagem dos pontos de manipulação de dados e transações se chega ao valor final da contagem de pontos de função.

Quando são considerados os pontos de manipulação de dados, estamos interessados nos arquivos internos ou externos ao escopo da contagem de pontos de função definidos inicialmente que são manipulados. No caso das transações, a contagem se concentra nas consultas realizadas internamente no escopo da contagem e nas consultas realizadas para fora do escopo da contagem e de fora do escopo da contagem para dentro do escopo.

Os dois tipos de manipulação de dados e os três tipos de transações podem ser contados com base em uma tabela de referência

utilizado pelo método no qual existe uma classificação com 3 níveis (baixo, médio e alto) e o valor atribuído para cada item de contagem é definido com base nessa tabela.

Existem ainda diferentes aspectos que devem ser observados para os tipos de contagem discutidos anteriormente.

Como exemplo, considere que temos um software de atendimento médico sendo desenvolvido e queremos realizar a contagem de pontos de função para a funcionalidade cadastro de paciente para o seu desenvolvimento. Neste caso, a definição do tipo de contagem é uma contagem [illegible] das funcionalidades que devem compor a versão final entregue para o cliente e o escopo da contagem é parcial, cobrindo somente o cadastro de pacientes.

O fator de ajuste, pode ser definido como alguem adotado na literatura, ou pode-se optar por não utilizar o fator de ajuste. Neste exemplo, o fator de ajuste não será utilizado.

O passo seguinte é realizar a contagem dos pontos de manipulação de dados e transações e, com o auxílio da tabela de valores para ponto de função disponível na literatura.

Note que para utilizar a tabela é necessário realizar uma análise das operações de manipulação de dados e transações para obter da tabela o valor apropriado ao nível definido para cada operação em questão.

Como exemplo de arquivo de dados interno manipulado, temos aquele que armazena os dados dos pacientes e não há nenhum arquivo de dados externo manipulado nesse escopo. Como só há um arquivo interno, define-se o nível de dificuldade para realizar a atualização desse arquivo com os dados do novo paciente. Julgamos que após essa análise tenha se chegado ao nível médio. Então vamos até a tabela e pegamos o valor para a manipulação de arquivos interno na coluna de nível médio. Repetimos o processo identificando, classificando e buscando na tabela os valores para transações.

Ao final, a contagem de pontos de função será composta pelos valores obtidos da tabela.