(Tópico 1) (I) Bruno Costa (I)

No ano de 1969, em uma conferência da Organização do Tratado do Atlântico Norte (OTAN), um grupo de pesquisadores e especialistas se reuniram para dialogar sobre questões importantes acerca do desenvolvimento de softwares da época. Naquele período, problemas recorrentes em sistemas computacionais motivaram a observância da, então, atividade de codificação sob o prisma da engenharia. Como resultado do encontro, a sistematização destas atividades em etapas bem definidas estabeleceu um incipiente processo, racional e previsível, de desenvolvimento de softwares. O processo prescreve atividades a serem executadas, insumos de entrada, artefatos de saída, assim como responsabilidades a serem cumpridas. Após melhorias e modificações, o processo em sua versão mais estável, ficou conhecido a partir da analogia a uma cachoeira ou cascata onde, para cada etapa a ser realizada no processo, a etapa imediatamente anterior deveria estar concluída, fazendo alusão figurativa ao comportamento da água. O modelo do processo foi amplamente utilizado pela indústria de softwares, até que novos problemas levaram a sua descontinuação no fim da década de 1990.

Em meados de 2001, em um novo encontro de pesquisadores, desenvolvedores, professores e entusiastas da área de desenvolvimento de softwares, o modelo prescritivo de desenvolvimento em cascata foi duramente criticado. Nas argumentações, os participantes, recordando a voz de muitas comunidades, empresas e grupos de pesquisa, identificaram os principais problemas deste modelo, os quais resultavam em falhas, atrasos e outras dificuldades: (i) intolerância à mudanças de requisitos; (ii) rigidez processual; (iii) demora na entrega de softwares funcionais; (iv) impossibilidade de validação intermediária de funcionalidades com o resultado final; (v) geração excessiva de documentação. Como resposta a estes problemas, um documento foi gerado, contendo princípios a serem seguidos no desenvolvimento de softwares, visando a qualidade do produto final. Tal documento ficou conhecido como "Manifesto Ágil".

A partir do Manifesto uma nova perspectiva se abriu para os processos de desenvolvimento de softwares. Emergiram, assim, iniciativas que buscavam estruturar as diversas atividades do desenvolvimento de sistemas computacionais à luz dos princípios contidos no documento.

II I

Os Processos ágeis de Software reconhecem a mudança inevitável de requisitos, promovendo a interação frequente com o usuário final a fim de apresentar-lhe componentes funcionais de software e receber o seu feedback. Apesar da identificação das etapas básicas, os processos não pretendem que elas sejam aplicadas de maneira rígida, mas, possível de adaptação, seguindo as especificidades das equipes e usuários. O software, assim, passou a ser desenvolvido de forma mais iterativa e incremental.

Um exemplo representativo de modelo de processo ágil é o SCRUM. O modelo estabelece três papéis principais em um time de desenvolvimento, a saber, o Scrum Master, responsável por orquestrar a equipe de desenvolvedores, removendo obstáculos para o cumprimento das tarefas; o Product Owner (PO), quem detém a visão do produto a ser desenvolvido e é responsável por priorizar as tarefas, e; os desenvolvedores, quem especifica e implementa os componentes do sistema. Um software construído à luz do SCRUM é entregue em períodos curtos, de 1 a 4 semanas, chamados Sprints. No início de um Sprint, o PO prioriza um conjunto de tarefas a serem desenvolvidas. Tais tarefas estão listadas no chamado "Product Backlog", o qual é criado no início do projeto e contém todas as funcionalidades desejadas para o produto de software.

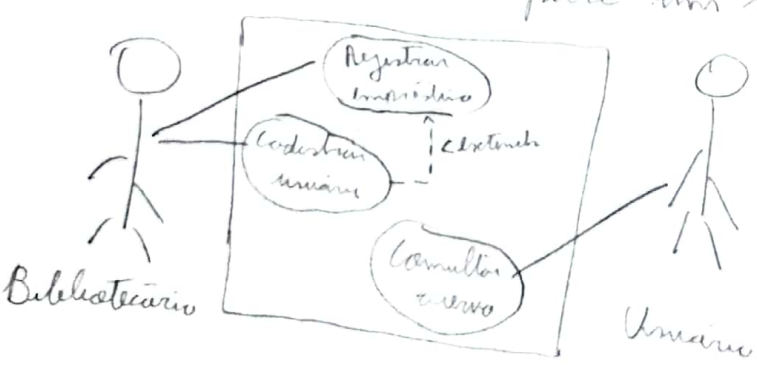
Atualmente, a utilização de modelos de processos ágeis de software é amplamente popularizada. No gerenciamento do processo, métricas podem ser aplicadas, tais como Planning Poker e Duodtime, fornecendo subsídios para análises e estimativas. No entanto, apesar de sua utilização global, novos problemas e desafios são continuamente apresentados aos processos ágeis de software, como a emergência de questões relacionadas à "infraestrutura como código" (DevOps) e arquitetura de software dinâmica, os quais indicam a necessidade, por exemplo, de novas iniciativas de pesquisa e desenvolvimento.

(TÓPICO III)

Como estabelecido na literatura de Engenharia de Software, por autores como Roger Pressman e Ian Sommerville, um sistema de software é concebido, modelado, para atender ou cumprir os seus requisitos. Requisitos de software são expressões, textuais ou simbólicas, que enunciam verbos, adjetivos ou advérbios acerca de um sistema computacional, ou seja, ações, características ou circunstâncias requeridas para um software. Em classificação estabelecida, requisitos podem ser de ordem funcional ou não-funcional. Um requisito funcional expressa funcionalidade (ação) esperada para o software em questão, geralmente associada ao objetivo do contexto no qual o software está inserido. Um requisito não-funcional representa características ou circunstâncias desejadas para o software, considerando o seu desenvolvimento (ex. manutenibilidade) ou uso (ex. usabilidade).

Ao expressar um requisito, espera-se que seja possível sua validação, garantindo que o software o atende, de maneira completa ou parcial. Casos de uso oferecem mecanismos que permitem a especificação, análise e validação de requisitos. Um caso de uso, portanto, é uma especificação, textual ou simbólica, de um requisito de software, passível de análise e validação.

Existem diversos modelos de estruturação de casos de uso. Em comum, um caso de uso deve informar os perfis dos usuários do software, ações que estes usuários devem executar no sistema e resultados esperados a partir da execução destas ações. A UML, por exemplo, provê um modelo e diagrama para a especificação de casos de uso. O perfil do usuário (chamado "ator") é identificado pelo símbolo  ; funcionalidades ou ações esperadas (chamadas, propriamente, Casos de Uso) são representadas por uma elipse; a identificação de relação entre um ator e o caso de uso é expressa por uma linha simples. Como exemplo, o diagrama a seguir representa três casos de uso para um sistema hipotético de gestão de bibliotecas



Os casos de uso "Registrar Empréstimo" e "Cadastrar Usuário" tem por ator principal a Bibliotecária. Por outro lado, o caso de uso "Consultar Livro" tem por ator principal o Usuário. No exemplo também foi especificada uma relação de extensão entre os casos de uso "Registrar Empréstimo" e "Cadastrar Usuário". Para fins de expressividade, esta relação indica que, ao cadastrar o empréstimo, uma bibliotecária pode cadastrar também um usuário (caso este não esteja cadastrado, por exemplo).

Complementar ao diagrama de casos de uso, um formulário contendo a descrição textual dos casos de uso pode ser elaborado bem, pré e pós condições, circunstâncias e características relacionadas a cada caso de uso permitiriam a validação mais precisa das funcionalidades esperadas para o software.

(TÓPICO IX)

No universo do desenvolvimento de softwares, dentre os diversos desafios inerentes à área, a estimativa de esforço, prazo, qualidade etc. figura entre os principais riscos de um projeto. Projetar uma demanda de forma muito imprecisa, pode acarretar em prejuízo financeiro e fracasso no projeto. Com o objetivo de mitigar esta questão, as métricas de softwares oferecem meios para uma análise quantitativa de sistemas. Sob o prisma do código, é possível medir sua complexidade, indicando centros de gargalo; seu acoplamento, indicando riscos de falhas acidentais em modificações pontuais. Considerando o desenvolvimento métricas como COCOMO e COCOMO II permitem estimar o esforço para o desenvolvimento de funcionalidades dentro de um período de tempo. No ponto de vista do usuário, é possível medir ou estimar o esforço no desenvolvimento de uma funcionalidade por meio de Pontos de Função.

A Contagem de Pontos de Função considera aspectos internos e externos com relação às funcionalidades do sistema. Para isto, o esforço é quantificado para componentes de software e dados e sua interação com dispositivos externos. Na técnica, tais elementos recebem os nomes de "Arquivos Lógicos Internos (ALI) e "Arquivos de Interface Externa" (AIE), sendo estes, elementos centrais na contagem. No fluxo de aplicação da técnica, inicialmente deve-se definir os elementos relacionados à contagem e as funcionalidades a serem medidas. Em seguida, reúne-se a documentação necessária e procede-se à contagem. Como resultado, obtem-se tabelas indicando um valor numérico que representa o esforço a ser empreendido no desenvolvimento das funcionalidades.

A técnica de Contagem de Pontos de Função é amplamente utilizada por empresas. No Brasil, por exemplo, ela é utilizada em processos licitatórios para artefatos de desenvolvimento de software, permitindo uma avaliação mais precisa do esforço. Internacionalmente, dispõe de uma unidade certificadora, a qual credita profissionais na aplicação da técnica.